UNIVERSIDAD NACIONAL DEL LITORAL

DOCTORADO EN INGENIERIA

Algoritmos de alta performance en Unidades de Procesamiento Gráfico (GPU) aplicados a la Dinámica de Fluidos Computacional

Santiago Daniel Costarelli

FICH
FACULTAD DE INGENIERIA
Y CIENCIAS HIDRICAS

INTEC
INSTITUTO DE DESARROLLO TECNOLOGICO
PARA LA INDUSTRIA QUIMICA

Tesis de Doctorado **2017**

UNIVERSIDAD NACIONAL DEL LITORAL
Facultad de Ingeniería y Ciencias Hídricas

Algoritmos de alta performance en Unidades de Procesamiento Gráfico (GPU) aplicados a la Dinámica de Fluidos Computacional

Santiago Daniel Costarelli

Tesis remitida al Comité Académico del Doctorado
como parte de los requisitos para la obtención
del grado de
DOCTOR EN INGENIERIA
Mención MECANICA COMPUTACIONAL
de la
UNIVERSIDAD NACIONAL DEL LITORAL

2017

Secretaría de Posgrado, Facultad de Ingeniería y Ciencias Hídricas, Ciudad Universitaria, Paraje "El Pozo",
S3000, Santa Fe, Argentina.

UNIVERSIDAD NACIONAL DEL LITORAL
Facultad de Ingeniería y Ciencias Hídricas

Algoritmos de alta performance en Unidades de Procesamiento Gráfico (GPU) aplicados a la Dinámica de Fluidos Computacional

Santiago Daniel Costarelli

Lugar de Trabajo:
CIMEC
Centro de Investigación de Métodos Computacionales
Facultad de Ingeniería y Ciencias Hídricas
Universidad Nacional del Litoral

Director:
Mario Alberto Storti, UNL/CONICET

Co-director:
Rodrigo Rafael Paz, UNL/CONICET

Jurado Evaluador:
Dr. Diego Campana, IBB UNER/CONICET
Dr. Carlos Sacco, IUA/FI
Dr. Alejandro Clause, UNC/CNEA
Dr. Norberto Nigro, UNL/CONICET

2017

ACTA DE EVALUACIÓN DE TESIS DE DOCTORADO

En la sede de la Facultad de Ingeniería y Ciencias Hídricas de la Universidad Nacional del Litoral, a los seis días del mes de abril del año dos mil dieciocho, se reúnen los miembros del Jurado designado para la evaluación de la Tesis de Doctorado en Ingeniería titulada ***"Algoritmos de alta performance en Unidades de Procesamiento Gráfico (GPU) aplicados a la Dinámica de Fluidos Computacional"***, desarrollada por el Ing. Santiago Daniel COSTARELLI, DNI Nº 33.548.246. Ellos son: el Dr. Diego Campana, el Dr. Carlos Sacco y el Dr. Norberto Nigro.

Luego de escuchar la Defensa Pública y de evaluar la Tesis, el Jurado resuelve:

aprobar la defensa de la tesis sin observaciones o modificaciones dejando constancia de una muy buena exposición oral. con una calificación de SOBRESALIENTE (10)

Sin más, se da por finalizado el Acto Académico con la firma de los miembros del Jurado al pie de la presente. ----------

Dr. Diego Campana	Dr. Carlos Sacco	Dr. Norberto Nigro

Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas

Secretaría de Posgrado

Ciudad Universitaria
C.C. 217
Ruta Nacional Nº 168 - Km. 472,4
(3000) Santa Fe
Tel: (54) (0342) 4575 229
Fax: (54) (0342) 4575 224
E-mail: posgrado@fich.unl.edu.ar

UNIVERSIDAD NACIONAL DEL LITORAL
Facultad de Ingeniería y Ciencias Hídricas

Santa Fe, 6 de abril de 2018.

Como miembros del Jurado Evaluador de la Tesis de Doctorado en Ingeniería titulada ***"Algoritmos de alta performance en Unidades de Procesamiento Gráfico (GPU) aplicados a la Dinámica de Fluidos Computacional"***, desarrollada por el Ing. Santiago Daniel COSTARELLI, en el marco de la Mención "Mecánica Computacional", certificamos que hemos evaluado la Tesis y recomendamos que sea aceptada como parte de los requisitos para la obtención del título de Doctor en Ingeniería.

La aprobación final de esta disertación estará condicionada a la presentación de dos copias encuadernadas de la versión final de la Tesis ante el Comité Académico del Doctorado en Ingeniería.

Dr. Diego Campana	Dr. Carlos Sacco	Dr. Norberto Nigro

Santa Fe, 6 de abril de 2018.

Certifico haber leído la Tesis, preparada bajo mi dirección en el marco de la Mención "Mecánica Computacional" y recomiendo que sea aceptada como parte de los requisitos para la obtención del título de Doctor en Ingeniería.

Dr. Rodrigo Paz Codirector de Tesis	Dr. Mario Storti Director de Tesis

Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas

Secretaría de Posgrado

Ciudad Universitaria
C.C. 217
Ruta Nacional Nº 168 - Km. 472,4
(3000) Santa Fe
Tel: (54) (0342) 4575 229
Fax: (54) (0342) 4575 224
E-mail: posgrado@fich.unl.edu.ar

Declaración legal del autor

Esta tesis ha sido remitida al departamento de posgrado de la Facultad de Ingeniería y Ciencias Hídricas como parte de los requisitos para la obtención del grado académico Doctor en Ingeniería mención Mecánica Computacional de la Universidad Nacional del Litoral y ha sido depositada en la Biblioteca de la Facultad de Ingeniería y Ciencias Hídricas para que esté a disposición de sus lectores bajo las condiciones estipuladas por el reglamento de la mencionada Biblioteca.

Citaciones breves de esta tesis son permitidas sin la necesidad de un permiso especial, en la suposición de que la fuente sea correctamente citada. Solicitudes de permiso para la citación extendida o para la reproducción parcial o total de ese manuscrito serán concebidos por el portador legal del derecho de propiedad intelectual de la obra.

Algunas de las partes del presente trabajo fueron publicadas en revistas tales como: Computer & Fluids, Cluster Computing y Computer Methods in Applied Mechanics and Engineering.

Santiago Daniel Costarelli

Agradecimientos

Esta tesis ha recibido apoyo financiero principalmente del Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), la Universidad Nacional del Litoral (UNL) y la Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT). Otros contribuyentes fueron el Consejo de Investigación Europea a través del Proyecto Avanzado: ERC-2009-AdG "Técnicas de Mecánica Computacional de Tiempo Real para problemas de Multi-Fluidos" y el "Programa de Becas de Formación en el exterior en Ciencia y Tecnología (BEC.AR) y Campus France".

El trabajo de investigación fue principalmente llevado a cabo en el Centro de Investigación de Métodos Computacionales (CIMEC) ubicado en Santa Fe, Argentina, instituto dependiente de la UNL y el CONICET.

Me gustaría agradecer a mi director, Mario Storti, por su guía y confianza a lo largo de estos años. Además de ello, este trabajo no podría haber sido realizado sin el soporte y la cálida compañía de las personas trabajando en CIMEC.

Un agradecimiento especial es también dado a toda la comunidad de software libre que me ha provisto de la enorme cantidad de herramientas que uso a diario.

Finalmente, pero no menos importante, esta tesis está dedicada a mis familiares y amigos que me han apoyado en todo momento.

Índice general

Índice de figuras

Índice de cuadros

Resumen

Una gran cantidad de problemas requieren el uso de computadoras para obtener su solución. Como ejemplo, las ecuaciones de Navier-Stokes son unas de las más sofisticadas, todavía no tan bien entendidas, ecuaciones acopladas, transientes y no lineales.

Tanto la academia como la industria requieren resolver problemas más grandes y complejos a medida que el tiempo pasa, y la electrónica y la computación son quienes les proveen herramientas cada vez más poderosas para alcanzar estos objetivos. Sin embargo, existen ciertas variables extremadamente importantes que deben ser consideradas pudiéndose destacar el tiempo de cálculo y los costos asociados. El primero se relaciona con el tipo de flujo que está siendo estudiado mientras que el segundo consiste en el costo del hardware, el tiempo es dinero y este siempre es escaso.

NVIDIA introdujo en 2007 una arquitectura que prometía ser de bajo costo y eficiente, su nombre era CUDA, y su poder de cómputo era entregado por las Unidades de Procesamiento Gráfico (GPU). De esta manera se hacía incapié en que una gran variedad de problemas podían ser atacados haciendo uso de estas unidades de cálculo extremadamente eficientes. Este es el punto de partida de la presente tesis, que trata de combinar los métodos más amigables para una GPU y así obtener soluciones a problemas de academia e industria de una forma relativamente económica.

En esta tesis se han estudiado flujos internos y externos, a bajo y medio número de Reynolds, con interacciones entre el flujo y las estructuras o con acople térmico. El método de Volúmenes Finitos en grillas Cartesianas fue escogido como discretización espacial usando esquemas colocados estabilizados por medio de la interpolación de Rhie-Chow. Además, la familia de métodos del tipo SIMPLE fue adaptada a efectos de introducir cuerpos y condiciones de contorno haciendo uso de métodos de fronteras embebidas.

Finalmente el método propuesto es aplicado a un problema de interacción fluido-estructura y es validado con resultados experimentales. Además, se presentan resultados parciales de un problema industrial que incluye acople térmico.

Resume

Many problems require the use of computers in order to obtain a solution. As an example, the Navier Stokes equations are one of the most sophisticated, not yet well understood, transient, non-linear, coupled equations.

Academy and industry need to solve bigger and more challenging problems as time goes by and electronic and computer advances provides them with more powerful tools to reach their goals. Nevertheless, there are some important variables that must be considered. Among the most important ones are computational time and costs. The former due to the type of flow being studied and the latter consisting on hardware costs, time is money and money is almost always scarce.

NVIDIA introduced in 2007 an architecture that promised to be cheap and fast, its name being CUDA, whereas the computing power is provided by the Graphic Processing Unit. This way they claimed that the problems introduced previously could be tackled using these extremely efficient computing units. This is the departure point of this thesis, that tries to combine the most GPU friendly methods to get academy and industry solutions and to study if it is possible to do so in the cheapest way.

In the present thesis both internal and external flows were studied, at low and medium Reynolds number, with interactions among the flow and the structures or even with thermal coupling. The Finite Volume Method in Cartesian grids has been chosen as spatial discretization with colocated schemes stabilized by the Rhie-Chow interpolation. Furthermore, the family of SIMPLE methods has been adapted to account for embedeed bodies and boundary conditions by using immersed boundary methods.

Finally the proposed method is applied to a fluid-structure problem being validated with experimental results. Moreover, some partial results of an industrial problem with thermal coupling is presented.

Capítulo 1

Introducción

1.1. Motivación

Las unidades de procesamiento gráfico (GPU) son coprocesadores de cálculo utilizadas principalmente para ayudar al procesador principal (CPU) en la tarea de renderizar visualizaciones computacionales. Durante las últimas décadas, estas han evolucionado a sistemas cada vez más complejos conteniendo cientos, sino miles, de unidades de procesamiento, una gran cantidad de memoria y un poder de cálculo del orden de los teraflops.

A lo largo de los últimos años, las GPU han ganado popularidad entre científicos e ingenieros que hacían uso de ténicas de Computación de Alta Performance (HPC) para resolver sus problemas cotidianos, dado que resultaban ser un dispositivo de cálculo barato pero poderoso [MCPN08, CCLW11]. Como resultado, esto motivó a los fabricantes de GPU a desarrollar una tarjeta gráfica de propósito general (GPGPU) orientada al mercado HPC.

Sin embargo, la resolución de problemas de Mecánica de Fluidos Computacional (CFD) en las GPGPU requieren, en general, de algoritmos especialmente adaptados para este tipo de arquitecturas. Los algoritmos que caen dentro de la categoría conocida como Autómatas Celulares (CA) son los que mejor se adaptan a las GPGPU. Por ejemplo, métodos explícitos del tipo Volúmenes Finitos (FVM) o de Elementos Finitos (FEM), en conjunto con técnicas de fronteras embebidas (IBM) para representar sólidos, pueden ser usados en mallas Cartesianas.

1.2. Trabajos previos

El tema de la presente tesis doctoral ha sido escogido luego de la publicación de un trabajo previo [SPD+13] donde la resolución de las ecuaciones de Navier-Stokes, usando un método de Gradientes Conjugados (CG) precondicionado con Transformadas Rápidas de Fourier (FFT), había sido desarrollado. El estudio hacía foco principalmente en los aspectos teóricos del precondicionamiento FFT aplicado a la resolución de la ecuación de Poisson para la presión.

En efecto, se demostró que el número de condición del sistema con cuerpos precondicionado (de la discretización de la ecuación de Poisson para la presión) resulta independiente del refinamiento de la malla cuando se usa este tipo de precondicionadores. Esto se muestra en la Figura (1.1), donde el número de condición para tres configuraciones diferentes, con y sin precondicionamiento, se grafican en términos del tamaño del problema.

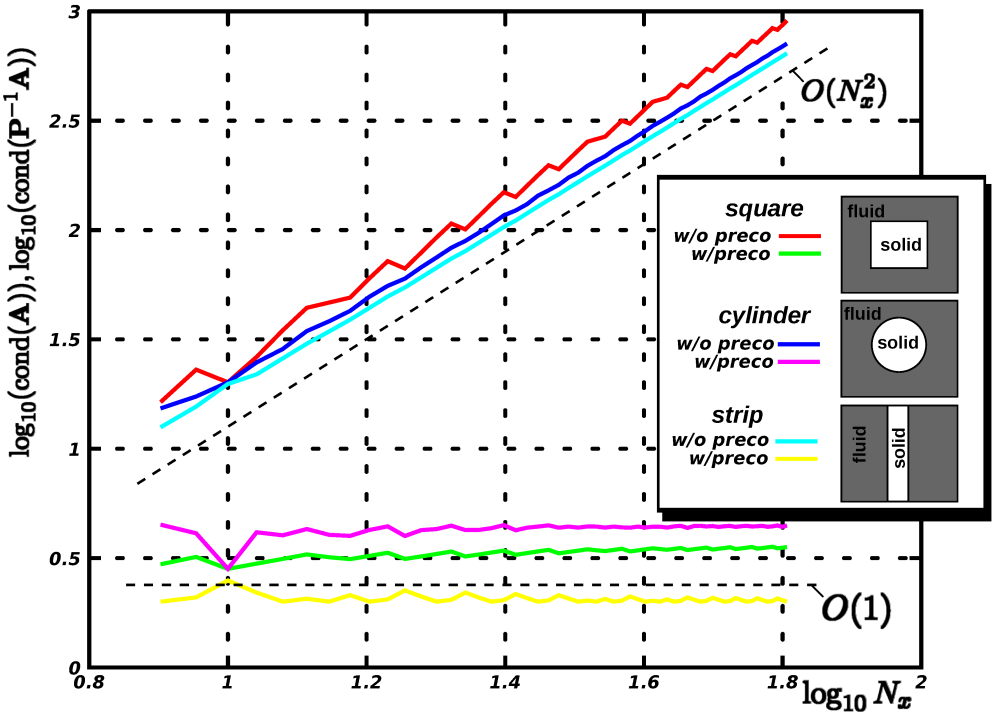


Figura 1.1: Número de condición vs tamaño del problema. El número de condición del sistema precondicionado es independiente del tamaño del problema.

Además, la FFT (y también el CG) presentaban implementaciones extremadamente eficientes en la arquitectura CUDA. En particular, las Operaciones de Punto Flotante por Segundo (flops) crecen a medida que las nuevas GPGPU salen al mercado y, más aún,

su rendimiento crece a medida que lo hace el tamaño del problema. En la Figura (1.2) se muestran los flops obtenidos por una NVIDIA GTX 580 en simple (SP) y doble precisión (DP).

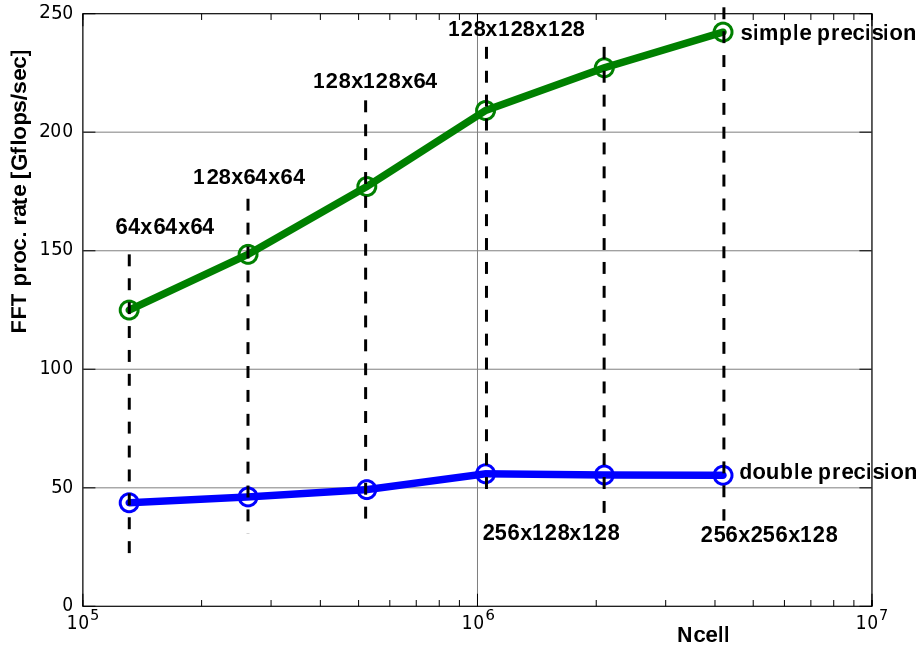


Figura 1.2: Flops obtenidos en el cálculo de la FFT por una NVIDIA GTX 580.

Existen sin embargo una serie de limitaciones que merecen ser mencionadas para realizar un correcto uso de la técnica aquí propuesta. Para empezar, es aplicable sólo para mallas cartesianas de paso constante por dirección. Además, como se hace uso de FFT se consideran implícitamente condiciones de contorno del tipo periódicas. Para introducir condiciones de contorno del tipo Dirichlet, por ejemplo, es necesario introducir en el dominio una región ficticia donde se fije el valor deseado. Esto introduce un costo adicional no sólo por el hecho de tener que considerar en el cálculo regiones que no son de interés, sino que también el número de condición se degrada (aunque levemente) y se requieren de esta forma más iteraciones para resolver el sistema.

Finalmente, este trabajo sirvió como motivación para atacar una serie de deficiencias que se habían encontrado en la implementación utilizada. A tales efectos, se consideró necesario estudiar en profundidad técnicas para incrementar el paso de tiempo, para mejorar la representación de cuerpos embebidos sin introducir costos adicionales en la etapa de la resolución de la ecuación de Poisson (haciendo uso de la técnica de FFT), para incrementar la performance general de la implementación, entre otros.

1.3. Mejoras a los trabajos previos y más estudios

1.3.1. Incrementando el paso de tiempo

Inicialmente todos los esfuerzos se aplicaron a mejorar el rendimiento de una implementación de primer orden. El objetivo principal era el de estudiar en que grado era posible explotar el poder de cómputo de la arquitectura CUDA para ser utilizada como un dispositivo de cálculo en *tiempo real* [SPD+13, CSP+14] sin perder de vista la física del problema, entendiendo como *tiempo real* a la posibilidad de calcular a velocidadades iguales o mayores a la que transcurre el tiempo físico de un problema determinado.

Durante esos años muchos intentos fueron llevados a cabo, algunos con más éxitos que otros, donde uno de los más prometedores (además del método que será descripto en la presente tesis) es el PFEM [GRD+16], también parcialmente desarrollado en CIMEC.

Como una primera mejora se propuso incrementar el paso de tiempo dado que la etapa de resolución de las ecuaciones de cantidad de movimiento resultaba ser la más costosa. Como resultado, el método BFECC [SC91, KLLR07] fue adoptado. Este método promovía la idea de que era posible incrementar la velocidad de cálculo usando una combinación de advección de primer orden en conjunto con una serie de etapas a aplicarse para reducir los errores disipativos. Algunos de los detalles inherentes de este método en particular se presentarán a continuación.

Considere por el momento un campo escalar F que esta siendo advectado por el campo de velocidad $\mathbf{u}$, esto es:

$$\frac{D_{(m)}F}{Dt} = \frac{\partial F}{\partial t} + \mathbf{u} \cdot \nabla F = 0 \tag{1.1}$$

donde $D_{(m)}(\cdot)/Dt$ hace referencia a la derivada material, esto es siguiendo una partícula de fluido. Aunque el método de características pueda ser explicado de manera abstracta (sin discretización espacial), para fijar ideas una simple grilla Cartesiana con paso espacial constante h será utilizado.

El mecanismo para resolver este tipo de ecuaciones sigue el trabajo presentado por [SC91]. Considerando la Ecuación (1.1) en 1D (su extensión a más dimensiones es trivial) y aplicando la regla de la cadena se obtiene:

$$\frac{\partial F}{\partial t} + \frac{dx}{dt}\frac{\partial F}{\partial x} = 0 \tag{1.2}$$

donde:

$$\frac{dx}{dt} = u(x;t) \tag{1.3}$$

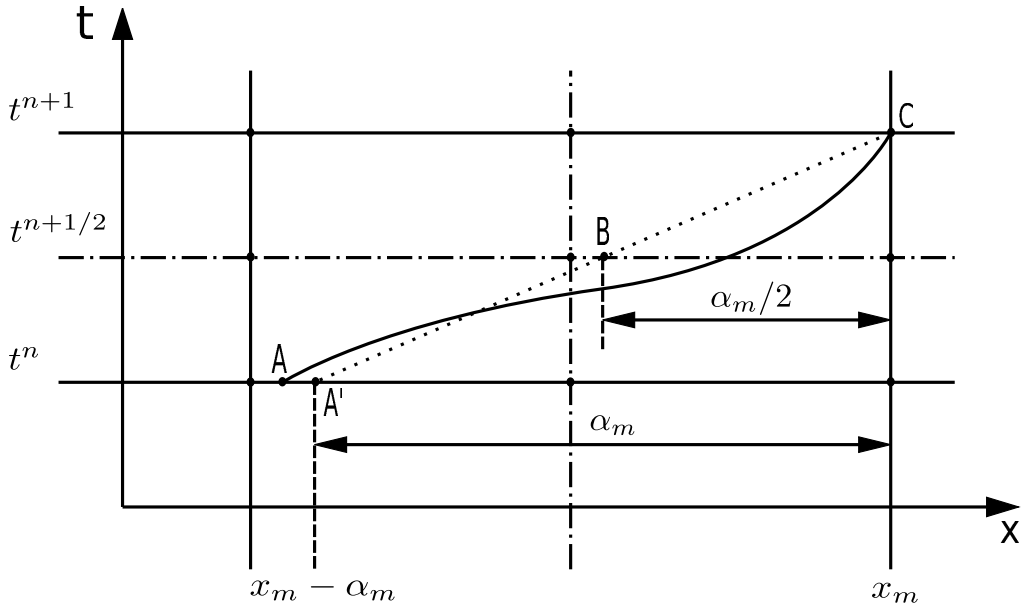


Figura 1.3: La trayectoria real, que conecta los puntos A y C, se muestra en trazo continuo. La trayectoria aproximada, $\overline{A'C}$, mostrada a trazo discontinuo, se calcula utilizando como predictor de primer orden al campo de velocidades.

es una ecuación diferencial ordinaria (ODE) para la posición de la partícula. Una vez que las trayectorias de las partículas sean determinadas la solución de la Ecuación (1.2) es obtenida simplemente por extrapolación considerando que el valor de F es constante a lo largo de la trayectoria [BC92]. Esta última ecuación relaciona, en particular, los puntos A y C de la Figura (1.3). El objetivo es rastrear la posición de la partícula pasando por el punto C al tiempo $t + \Delta t$ al punto definido como A al tiempo t. Esto debería ser realizado siguiendo la trayectoria de trazo continuo, pero en el caso discreto esto es aproximado usando el campo de velocidades como un predictor de primer orden de la posición previa, curva a trazo discontinuo, alcanzando el punto A'. Un análisis de las propiedades de estabilidad del esquema de advección semi lagrangiano [BM82] muestra que es posible integrar la Ecuación (1.3) para CFL mayores que uno, donde $\text{CFL} = U_{\max}\Delta t/h$ y $U_{\max}$ es la velocidad máxima del campo de velocidades.

Habiendo introducido la formulación semi Lagrangiana, es tiempo de incorporar algunas mejoras propuestas por [KLLR07]. Se define entonces el operador $L(.,.)$ como:

$$F^{n+1} = L(\mathbf{u}, F^n) \tag{1.4}$$

En el caso continuo una operación hacia adelante $L(-\mathbf{u}, \cdot)$ compensa exactamente una operación hacia atrás $L(\mathbf{u}, \cdot)$, como lo muestra la Figura (1.4), en otras palabras el mismo campo se obtiene al aplicar, de a pares, las operaciones de adelante y atrás (notar que esto

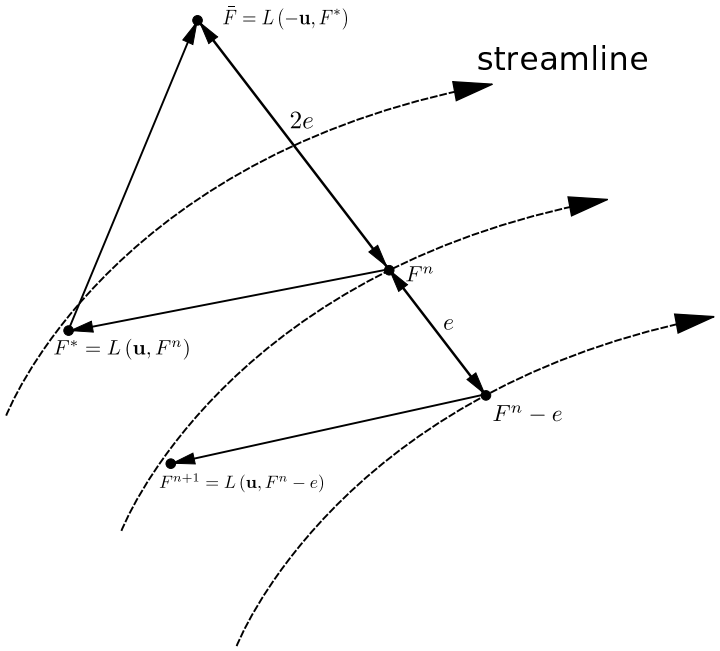


Figura 1.4: Diagrama esquemático del funcionamiento del BFECC usando $L(.,.)$ como el operador de advección para el campo escalar F.

será válido sólo si no existe difusión). Por supuesto esto no será válido cuando se tengan operadores discretos debido a errores numéricos. Considerando entonces que la operación hacia atrás introduce un error, e, puramente disipativo, luego la operación hacia adelante introducirá el mismo error y en efecto luego de dos pasadas, una adelante y otra hacia atrás, un campo $\bar{F} = F^n + 2e$ será obtenido. De esta forma, una expresión explícita para e puede ser hallada fácilmente, esto es:

$$e = -\frac{1}{2}\left(F^n - \bar{F}\right) \tag{1.5}$$

Este error puede ser sustraido (etapa llamada *corrección y compensación del error*) de F^n que luego será finalmente advectado (operación hacia adelante). La combinación de BFECC con la integración temporal previamente estudiada para el operador lagrangiano resulta de segundo orden, tanto en tiempo como en espacio [SFK+08].

Resumiendo, el BFECC se define como:

$$\begin{aligned} F^* &= L(\mathbf{u}, F^n) \\ \bar{F} &= L(-\mathbf{u}, F^*) \\ F^{**} &= F^n + \left(F^n - \bar{F}\right)/2 \\ F^{n+1} &= L(\mathbf{u}, F^{**}) \end{aligned}$$

Finalmente, como mecanismo de resolución de las ecuaciones de Navier-Stokes se utilizaban métodos del tipo Pasos Fraccionados [CSP+14], donde existían una serie de etapas que eran aplicadas sucesivamente. En particular, se tenian etapas de advección (haciendo uso de BFECC), de difusión (explícita, aunque el Fourier, Fo $= \nu \Delta t / h^2$, del problema nunca fue una limitante para los problemas que se han resuelto a lo largo de este trabajo), de resolución de la ecuación de Poisson para la presión y finalmente de corrección del campo de velocidades con la presión previamente obtenida.

Este método, aunque matemáticamente simple, contiene patrones de acceso a memoria que degradan el rendimiento de la GPGPU. Sin embargo, y a modo de resumen, pudo concluirse que aunque la implementación en CUDA resultó ineficiente (considerando todos los beneficios que podría brindar), su implementación satisfactoriamente incrementó su performance [CSP+14].

1.3.2. Progreso hacia una mejor representación de superficies

Hoy en día existen un gran variedad de técnicas disponibles para mejorar la representación de superficies embebidas. Sin embargo, sólo un grupo selecto presenta implementaciones amigables para la arquitectura CUDA. Una de las opciones más prometedoras fue la propuesta en [FVOMY00]. En este trabajo se estudian diversos problemas, entre los que podemos destacar el flujo externo pasando por una esfera y un problema relacionado al flujo que se desarrolla dentro de una configuración pistón-cilindro cuando este se encuentra en pleno regimen. En particular, se desarrollan diversas técnicas de representación de superficies embebidas (de primer y segundo orden), entre las cuales se puede destacar la que se ha utilizado como guía a lo largo de la presente tesis, y se hace un análisis exhaustivo de ellas.

Los detalles de la implementación propuesta serán desarrollados en el Capítulo (2). A efectos de introducir la técnica de IB, se introducirá una modificación al lazo SIMPLE (Semi Implicit Method for Pressure Linked Equations) [VM07]. Como resultado, se dispondrá de una implementación eficiente para la arquitectura CUDA (considerando que este tipo de técnicas IB presentan las ventajas de los métodos del tipo CA) y además se podrá utilizar la estrategia de resolver la ecuación de Poisson con el precondicionamiento FFT, puesto que las condiciones de contorno, tanto en el dominio como en el sólido, serán satisfechas eficientemente al considerar sólo fases de fluidos, es decir, en aquellas regiones del dominio donde se tengan que satisfascer condiciones de contorno existirán flujos parásitos que

ayudarán a satisfacer dichas condiciones.

Finalmente, los resultados obtenidos serán discutidos en el Capítulo (3), donde se hará un análisis detallado de implementaciones con esquemas de primer y segundo orden para el caso de una boya sumergida en un tanque cúbico relleno con agua sujeto a desplazamientos armónicos impuestos por una mesa vibradora y se validarán los resultados numéricos con experimentales.

En este punto, y debido al método de IB escogido, se optó por dejar de lado al BFECC. En particular, dicho método requería resolver un mayor número de veces las ecuaciones de cantidad de movimiento y, considerando que el cálculo del BFECC no era eficiente, degradaba totalmente el rendimiento del método propuesto. De esta forma se optó por utilizar métodos tradicionales de Disminución Total de Variación (Total Variation Diminishing, o TVD).

1.3.3. Aplicación a un caso industrial

Finalmente, el método propuesto estaba listo para ser aplicado a un problema industrial. Una descripción completa del problema será presentado en el Capítulo (4). El objetivo entonces es el de mejorar la transferencia térmica dentro de transformadores industriales haciendo uso de los denominados *Generadores de Vórtices*. Es así como se estudiaron una variedad de formulaciones que incluían acople térmico. Luego de un análisis detallado de las condiciones operativas, se escogió por implementar la aproximación de Boussinesq. Su simplicidad y ventajas (sin perder de vista sus desventajas) estaban en línea con los dos principales objetivos de la presente tesis, performance y precisión.

Capítulo 2

Aproximación numérica

2.1. El problema continuo

Las ecuaciones de Navier-Stokes describen la dinámica de los fluidos viscosos. Se obtienen aplicando la segunda ley de Newton al movimiento del fluido. Un gran número de supuestos se aplican para obtener las ecuaciones finales, siendo una de las más importantes el hecho que las tensiones en el fluido son la suma de efectos difusivos, o viscosos (proporcionales al gradiente de velocidad), y efectos de presión.

Las ecuaciones de Navier-Stokes para un fluido isotérmico Newtoniano son:

$$\begin{aligned} \rho\left(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}\right) + \nabla p - \nabla\cdot\left(2\mu\epsilon\left(\mathbf{u}\right)\right) = \rho\mathbf{f} \quad &\text{en } \Omega \\ \nabla\cdot\mathbf{u} = 0 \quad &\text{en } \Omega \end{aligned} \tag{2.1}$$

para el tiempo $t \in [0;\infty)$, donde Ω es el dominio completo, $\mathbf{u}$ es el campo de velocidades, p es el campo de presión, $\mathbf{f}$ es la fuerza de cuerpo por unidad de masa, ρ es la densidad del fluido, μ es la viscosidad dinámica y $\epsilon\left(\mathbf{u}\right) = 1/2\left(\nabla\mathbf{u} + \nabla^T\mathbf{u}\right)$ es el tensor velocidad de deformación.

Las condiciones de contorno requeridas para resolver las ecuaciones de Navier-Stokes van a depender del problema en particular estudiado, por lo tanto van a ser explícitamente especificadas cuando sea necesario.

Suponga que $\mathbf{f}$ es un campo de fuerzas gravitatorio, esto es $\mathbf{f} = \mathbf{g}$, donde $\mathbf{g}$ representa la gravedad. Si se considera que la gravedad y la densidad son constantes, el término $\rho\mathbf{g}$ puede ser escrito como $\nabla\left(\rho\mathbf{g}\cdot\mathbf{r}\right)$, donde $\mathbf{r}$ es el vector posición y se define como $\mathbf{r} = x_i\hat{\mathbf{r}}_i$ (asumiendo la convención de Einstein). La gravedad actúa usualmente en la dirección negativa de z, esto es $\mathbf{g}\cdot\mathbf{r} = g_z z$, donde $g_z < 0$. Luego $\rho g_z z$ es la presión hidrostrática y es conveniente (desde el punto de vista numérico) definir $\tilde{p} = p - \rho g_z z$ como la altura

gravimétrica (head) y usar esta nueva variable en reemplazo de la presión. Esto no es un problema en fluidos incompresibles puesto que sólo los gradientes de la presión son requeridos.

En fluidos con densidad variable el término $\rho\mathbf{g}$ puede ser expresado como $\rho_0\mathbf{g} + (\rho - \rho_0)\,\mathbf{g}$, donde ρ_0 es una densidad de referencia. La aproximación de Boussinesq establece que si la variación de densidad no es tan grande esta puede ser tratada como constante en los términos temporales y convectivos, y tratarla como variable sólo en el término gravitatorio.

Se asume entonces que la densidad varía linealmente con la temperatura. Si la altura gravimétrica es utilizada en lugar de la presión en sí, se puede expresar el término gravitatorio como:

$$(\rho - \rho_0)\, g_i = -\rho_0 g_i \beta\,(T - T_0) \tag{2.2}$$

donde β es el coeficiente de expansión volumétrica, T es el campo de temperatura y T_0 es una temperatura de referencia. En efecto, ρ_0 es la densidad del fluido a temperatura T_0. Esta aproximación será valida sólo si la variación de densidad en la configuración real difiere de la empleada en el modelo numérico dentro del rango del $1 - 10\,\%$.

La ecuación de conservación de energía, que permite relacionar a $\mathbf{u}$ y T, puede ser reducida a su forma más simple al considerar cierta serie de hipótesis [FP12]. En particular, la expresión resultante puede ser escrita como:

$$\rho c_p \left(\frac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T\right) = \kappa \Delta T \tag{2.3}$$

donde c_p es el coeficiente de calor específico a presión constante y κ es la conductividad térmica. Vale la pena mencionar que se considerará tanto a c_p como κ constantes.

De esta manera el sistema final de ecuaciones, combinando la Ecuación (2.1) y la Ecuación (2.3) es:

$$\begin{aligned} \rho\left(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}\right) + \nabla p - \nabla\cdot\left(2\mu\epsilon\left(\mathbf{u}\right)\right) &= -\rho_0 g_i \beta\,(T - T_0) && \text{en } \Omega \\ \nabla\cdot\mathbf{u} &= 0 && \text{en } \Omega \\ \rho c_p \left(\frac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T\right) &= \kappa\Delta T && \text{en } \Omega \end{aligned} \tag{2.4}$$

donde, a efectos de simplicidad en la escritura, el tilde sobre el campo de presiones fue eliminado.

2.2. Discretización por Volúmenes Finitos

Considere entonces una versión reducida de la Ecuación (2.4), esto es:

$$\begin{aligned}
\frac{\partial(\rho u)}{\partial t}+\frac{\partial(\rho u u)}{\partial x}+\frac{\partial(\rho u v)}{\partial y} &= -\frac{\partial p}{\partial x}+\frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial u}{\partial y}\right)\\
\frac{\partial(\rho v)}{\partial t}+\frac{\partial(\rho v u)}{\partial x}+\frac{\partial(\rho v v)}{\partial y} &= -\frac{\partial p}{\partial y}+\frac{\partial}{\partial x}\left(\mu\frac{\partial v}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial v}{\partial y}\right)\\
\frac{\partial(\rho u)}{\partial x}+\frac{\partial(\rho v)}{\partial y} &= 0
\end{aligned} \tag{2.5}$$

Cada término de la Ecuación (2.5) va a ser integrado en una ventana de tiempo de longitud Δt dentro de un volumen de control de volumen $\Delta\Omega$. El conjunto completo de los términos de la ecuación discretizada de cantidad de movimiento en la dirección x será derivada, las discretizaciones en otras direcciones son obtenidas mediante un proceso similar [VM07].

Considere entonces primero el término transiente de la Ecuación (2.5), esto es:

$$\int_{\mathrm{CV}}\int_{t}^{t+\Delta t}\frac{\partial(\rho u)}{\partial t}\,dt\,dV \tag{2.6}$$

luego haciendo uso del teorema fundamental del cálculo se obtiene:

$$\int_{\mathrm{CV}}\left[(\rho u)-(\rho u)^{n}\right]\,dV \tag{2.7}$$

donde el superíndice n hace referencia a una evaluación a tiempo t y la falta se superíndice hace referencia a una evaluación a tiempo $t+\Delta t$.

Aplicando la regla de valor medio con el objetivo de evaluar la Ecuación (2.7), la ecuación se lee ahora:

$$\left[(\rho u)_{P}-(\rho u)_{P}^{n}\right]\Delta\Omega \tag{2.8}$$

La integración temporal será descartada de aquí en adelante y la discusión respecto a la evaluación temporal va a ser introducida cuando se traten las fronteras embebidas y el lazo del método SIMPLE.

El término (linealizado) de convección de la ecuación de cantidad de movimiento estudiada será discretizado a segundo orden espacial haciendo uso de esquemas TVD. Una extensa revisión del tema puede encontrarse en [YWH85]. La expresión resultante puede ser expresada como la suma de dos términos:

$$\int_{\mathrm{CV}}\left[\frac{\partial(\rho u u)}{\partial x}+\frac{\partial(\rho u v)}{\partial y}\right]\,dV=\sum_{nb}a_{nb}^{C}u_{nb}+\left(S_{u}^{dc}\right)_{P} \tag{2.9}$$

donde $nb = span\{E, W, N, S, P\}$ y los coeficientes de convección del primer término estan dados por:

$$\begin{aligned} a_E^C &= \min\left(\dot{m}_e^n, 0\right) \\ a_W^C &= \min\left(\dot{m}_w^n, 0\right) \\ a_N^C &= \min\left(\dot{m}_n^n, 0\right) \\ a_S^C &= \min\left(\dot{m}_s^n, 0\right) \\ a_P^C &= -\left(a_E^C + a_W^C + a_N^C + a_S^C\right) + \underline{\sum_f \dot{m}_f} \end{aligned} \tag{2.10}$$

donde vale la pena mencionar que, en algún momento del algoritmo propuesto el término subrayado en el coefficiente P puede ser diferente de cero. Esto es así por el tipo de tratamiento de las fronteras embebidas y las condiciones de contorno. El término va a anularse a medida que el lazo de fronteras embebidas converja, tema que será desarrollado en la Sección (2.4).

Si bien es cierto que los esquemas TVD van a incrementar la longitud del *stencil* utilizado para realizar los cálculos (en comparación con un método del tipo *upwind* por ejemplo), la performance de la implementación en CUDA no se degradará demasiado considerando que seguirá existintiendo una cierta localidad en el acceso a la memoria y esto siempre será particularmente beneficioso.

El segundo término de la Ecuación (2.9), el término de corrección diferida (deferred), será:

$$\left(S_u^{dc}\right)_P = \frac{1}{2}\left[\max\left(\dot{m}_e^n, 0\right)\ \Psi\left(r_e^+\right) - \min\left(\dot{m}_e^n, 0\right)\ \Psi\left(r_e^-\right)\right]\left(u_E - u_P\right) + \tag{2.11}$$

$$\frac{1}{2}\left[\max\left(\dot{m}_w^n, 0\right)\ \Psi\left(r_w^-\right) - \min\left(\dot{m}_w^n, 0\right)\ \Psi\left(r_w^+\right)\right]\left(u_W - u_P\right) + \tag{2.12}$$

$$\frac{1}{2}\left[\max\left(\dot{m}_n^n, 0\right)\ \Psi\left(r_n^+\right) - \min\left(\dot{m}_n^n, 0\right)\ \Psi\left(r_w^-\right)\right]\left(v_N - v_P\right) + \tag{2.13}$$

$$\frac{1}{2}\left[\max\left(\dot{m}_s^n, 0\right)\ \Psi\left(r_s^-\right) - \min\left(\dot{m}_s^n, 0\right)\ \Psi\left(r_w^+\right)\right]\left(v_S - v_P\right) \tag{2.14}$$

donde $\Psi(r)$ es la función delimitadora y su parámetro r se define como el cociente entre los gradientes aguas arriba y abajo de la variable dependiente. Por ejemplo:

$$r_e^+ = \frac{u_P - u_W}{u_E - u_P} \tag{2.15}$$

mientras que:

$$r_e^- = \frac{u_E - u_{EE}}{u_P - u_E} \tag{2.16}$$

donde el superíndice indica la dirección del flujo.

El mismo proceso será aplicado al término de difusión. Entonces:

$$\int_{\mathrm{CV}} \left[\frac{\partial}{\partial x} \left(\mu \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(\mu \frac{\partial u}{\partial y} \right) \right] \, dV \tag{2.17}$$

y usando el teorema de Gauss:

$$\left(\mu A \frac{\partial u}{\partial x} \right)_e - \left(\mu A \frac{\partial u}{\partial x} \right)_w + \left(\mu A \frac{\partial u}{\partial y} \right)_n - \left(\mu A \frac{\partial u}{\partial y} \right)_s \tag{2.18}$$

A efectos de obtener un error de discretizacion de segundo orden un esquema centrado será escogido para el término difusivo. De esta forma, y luego de algunos arreglos algebraicos, la expresión final puede ser escrita como:

$$\sum_{nb} a_{nb}^D u_{nb} \tag{2.19}$$

donde:

$$\begin{aligned} a_E^D &= \frac{(\mu A)_e}{\Delta x_{EP}} \\ a_W^D &= \frac{(\mu A)_w}{\Delta x_{WP}} \\ a_N^D &= \frac{(\mu A)_n}{\Delta x_{SP}} \\ a_S^D &= \frac{(\mu A)_s}{\Delta x_{NP}} \\ a_P^D &= -\left(a_E^D + a_W^D + a_N^D + a_S^D \right) \end{aligned} \tag{2.20}$$

Finalmente, un esquema de segundo orden será usado también para el término gradiente de presión. De esta manera:

$$\begin{aligned} \int_{\mathrm{CV}} \frac{\partial p}{\partial x} \, dV = (Ap)_e - (Ap)_w &= A_e \left(\frac{p_E + p_P}{2} \right) - A_w \left(\frac{p_P + p_W}{2} \right) \\ &= \left[\frac{A_e}{2} \right] p_E + \left[\frac{A_e}{2} - \frac{A_w}{2} \right] p_P - \left[\frac{A_w}{2} \right] p_W \end{aligned} \tag{2.21}$$

Asumiendo grillas Cartesianas con paso de malla constante por dirección, las ecuaciones derivadas previamente serán simplificadas. El tratamiento de las fronteras embebidas será explicado a continuación y posteriormente se presentará la versión final del método propuesto.

2.3. Método de Fronteras Embebidas

El principal objetivo del método de Fronteras Embebidas es evitar la tarea altamente costosa de la regeneración de mallas debido al desplazamiento de geometrías complejas o

de fronteras móviles. Una gran variedad de estos métodos han sido propuestos a lo largo de la últimas décadas. Un extenso desarrollo respecto a las diversas variantes existentes puede ser encontrado en [MI05], donde se pueden destacar aquellos métodos donde las fronteras embedidas se introducen explícitamente en los operadores discretos (modificando su construcción [TF03, MKLU05]), usando fuerzas de cuerpo [Pes73] y hasta haciendo uso de nodos fantasmas (ghost nodes) extendiendo el dominio del problema dentro del sólido cuya frontera se está incorporando [MY97, FVOMY00]. Al ser de interés sólo aquellas formulaciones que presenten una implementación eficiente en términos de la arquitectura CUDA, se ha optado por elegir aquellas combinando los nodos fantasmas y fuerzas de cuerpo.

A tal fin, en esta tesis se optó por seguir el trabajo de [MY97]. Las condiciones de contorno en la velocidad y presión son garantizadas mediante la extrapolación de las velocidades del fluido dentro del sólido. De esta manera, y sin importar el hecho de que un flujo parásito vaya a desarrollarse dentro del sólido, es él quien va a garantizar que las condiciones de contorno se cumplan.

Una característica a destacar del método escogido es que no se necesitan condiciones de contorno en la presión. Las condiciones de contorno en la velocidad sí son prescriptas, o calculadas, usando los nodos fantasmas (calculando una fuerza compensatoria), resultando una operación muy eficiente en la arquitectura CUDA. El método propuesto es usado en el dominio completo, fluido y sólido. Finalmente, como los operadores discretos se mantienen sin modificaciones las propiedades de convergencia del método no cambian y, más aún, el método es particularmente fácil de agregar en caso que se tuviera alguna implementación previa de primer orden.

Los cuerpos, o fronteras embebidas, son descriptas haciendo uso de técnicas del tipo Curvas de Nivel (Level-Set, LS) [OF06] donde ψ hace referencia a la función de LS con la convención de que $\psi = 0$ representa la superficie del cuerpo y $\psi < 0$ ($\psi > 0$) identifica a la región del sólido (fluido).

Considerando la situación mostrada en la Figura (2.1), los puntos negros (blancos) representan los centros de celda que pertenecen a la región del sólido (fluido). Los valores de velocidad en los puntos del fluido son conocidos y son utilizados para extrapolar propiedades en los puntos del sólido, garantizando de esta forma las condiciones de contorno en la superficie del cuerpo. Tomando como ejemplo el punto del sólido S, y

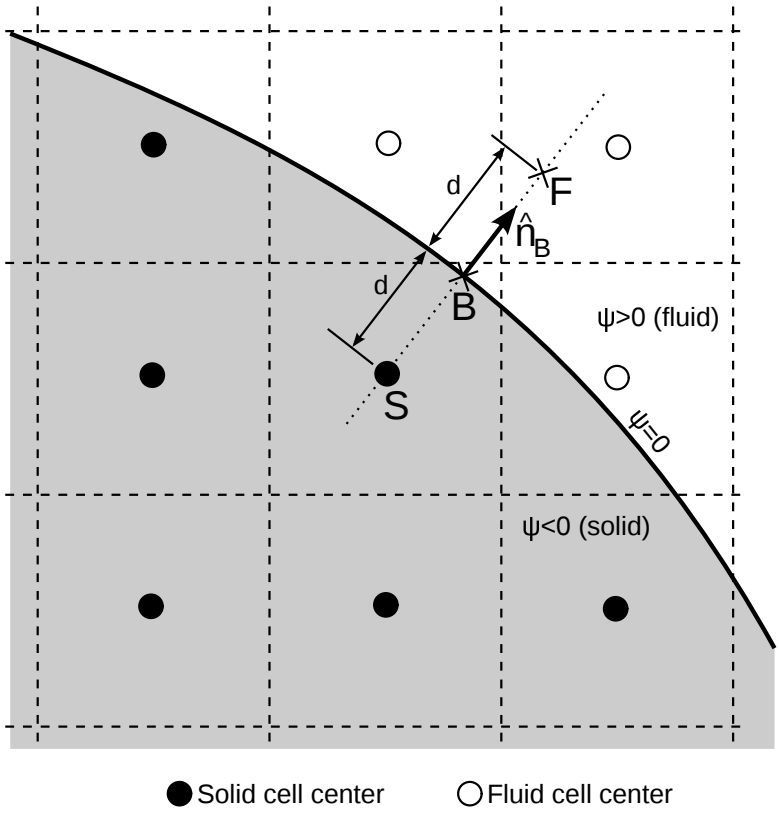


Figura 2.1: Ejemplo de operador de proyección.

siendo B el punto en la superficie del cuerpo más cercano a S, se tiene que:

$$\mathbf{x_S} = \mathbf{x_B} - \mathbf{d} \cdot \mathbf{n_B} \tag{2.22}$$

donde $\mathbf{d}$ es la distancia entre B y S, y $\mathbf{n_B}$ es la normal a la superficie del cuerpo en el punto B. Dicha normal puede calcularse como:

$$\mathbf{n_B} = \left(\frac{\nabla \psi}{\|\nabla \psi\|} \right)_B \approx \left(\frac{\nabla \psi}{\|\nabla \psi\|} \right)_S \tag{2.23}$$

Sea F el punto espejado de S con respecto a la normal en la superficie del cuerpo, esto es F descansa sobre la proyección de la normal que pasa a través de S a una distancia d de la frontera, entonces:

$$\mathbf{x_F} = \mathbf{x_B} + \mathbf{d} \cdot \mathbf{n_S} \tag{2.24}$$

El valor del campo de velocidades $\mathbf{u_F}$ en $\mathbf{x_F}$ se obtiene extrapolando el valor del campo de velocidades entre los centros de celdas más próximos (indicados como puntos en la Figura (2.1)). Partiendo de este valor de campo de velocidades $\mathbf{u_F}$ un valor extrapolado del campo de velocidades $\mathbf{u_S^f}$ en el punto $\mathbf{x_S}$ se calcula con la condición de satisfacer la condición de contorno $\mathbf{u_B}$ en $\mathbf{x_B}$. Note que este valor del campo de velocidades $\mathbf{u_S^f}$ dentro del sólido es ficticio (parásito). Una simple extrapolación lineal permite obtener la siguiente

relación:

$$\mathbf{u_S^f} = 2\mathbf{u_B} - \mathbf{u_F}. \tag{2.25}$$

El cálculo de la función de LS generalmente requiere de mucho tiempo, sin embargo existen muchos casos en los cuales se dispone de una representación matemática reduciendo así costos computacionales. Para geometrías complejas se debe tener especial atención al método de cálculo de dicha función con el objetivo de no deteriorar la eficiencia del método subyacente. Con este objetivo, algoritmos que combinan las soluciones de ecuaciones diferenciales parciales con algoritmos geométricos, pueden ser utilizados [EFFM02].

2.4. El método SIMPLE

Considerando que los superíndices $*$ y l hacen referencia a la iteración actual y a la iteración externa (relajada) del lazo de SIMPLE (los distintos tipos de iteraciones serán estudiados en la Sección (2.7)), mientras que el superíndice n hará referencia a la solución del paso de tiempo anterior, las ecuaciones discretas de Navier-Stokes explícitas relajadas pueden ser escritas como [Pas11]:

$$u_P^* = \alpha_u \underline{\left[u_P^n + \frac{\Delta t}{\rho \Delta V} \sum_{nb} A_{nb}^u u_{nb}^* - \frac{\Delta t}{\rho} \left(\left. \frac{\partial p}{\partial x} \right|_P^l - S_P^u \right) \right]} + (1 - \alpha_u)\, u_P^l + \frac{\Delta t}{\rho \Delta V} (1 - \alpha_u)\, A_{P|P}^u \left(u_P^l - u_P^* \right) \tag{2.26}$$

donde:

$$A_{nb}^u = a_{nb}^D - a_{nb}^C, \quad \left. \frac{\partial \mathrm{p}}{\partial \mathrm{x}} \right|_{\mathrm{P}}^{\mathrm{l}} = \frac{\mathrm{p_E} - \mathrm{p_W}}{2\Delta \mathrm{x}}, \text{ and } \quad \mathrm{S_P^u} = \left(\mathrm{S_u^{ibm}} \right)_{\mathrm{P}} + \left(\mathrm{S_u^{force}} \right)_{\mathrm{P}} - \left(\mathrm{S_u^{dc}} \right)_{\mathrm{P}} \tag{2.27}$$

siendo α_u el factor de relajación de la velocidad, $A_{P|P}^u$ el coeficiente central del nodo P ($A_{P|E}^u$ representaría entonces el coeficiente central del nodo E) y S_P^u es el término fuente que engloba los efectos de las fronteras embebidas, las fuerzas de cuerpo y las correcciones diferidas.

El término subrayado en la Ecuación (2.26) representa la clásica solución de Navier-Stokes en estado estacionario. Sin embargo son necesarias ciertas modificaciones a la anterior ecuación para independizarla de Δt y α_u para finalmente converger a la solución de estado estacionario cuando sea necesario.

Manipulando algebraicamente a la Ecuación (2.26) se obtiene:

$$\begin{aligned} A^u_{P|P} u^*_P = \sum_{j|P} A^u_j u^*_j - \Delta V \left(\left.\frac{\partial p}{\partial x}\right|^l_P - S^u_P \right) \\ + \frac{(1-\alpha_u)}{\alpha_u} \tilde{A}^u_{P|P} \left(u^l_P - u^*_P \right) \\ + \frac{\rho \Delta V}{\Delta t} \left(u^n_P - u^*_P \right) \end{aligned} \tag{2.28}$$

donde puede observarse con facilidad la solución de estado estacionario en la primer línea, la independencia del parámetro de relajación en la segunda y finalmente la independencia del paso de tiempo en la última línea, cuando se considera $u^*_P = u^l_P = u^n_P$. Más aún, existen dos grandes modificaciones respecto a la Ecuación (2.26). Primero, la sumatoria es desarrollada sobre los nodos adyacentes al nodo P y como resultado dicho nodo no estará presente de aquí en más en la sumatoria. Segundo, se incorporó un nuevo parámetro definido como $\tilde{A}^u_{P|P} = A^u_{P|P} + \rho \Delta V / \Delta t$ a efectos de mejorar la legibilidad de la ecuación resultante.

Se presenta entonces la Ecuación (2.26) como fue finalmente implementada, esto es:

$$\begin{aligned} \left[1 + (1-\alpha_u)\, \delta_P^{-1} \right] u^*_P = \alpha_u \left[u^n_P + \frac{\Delta t}{\rho \Delta V} \sum_{nb} A^u_{nb} u^*_{nb} - \frac{\Delta t}{\rho} \left(\left.\frac{\partial p}{\partial x}\right|^l_P - S^u_P \right) \right] \\ + (1-\alpha_u) \left(1 + \delta_P^{-1} \right) u^l_P \end{aligned} \tag{2.29}$$

donde $\delta_P = \rho \Delta V / \Delta t A^u_{P|P}$. Es importante destacar que, aunque se utilizan esquemas explícitos, al momento u^*_P aparece en ambos lados de la Ecuación (2.29). El método para evaluar explícitamente esta ecuación se desarrollará a continuación.

El método de fronteras embebidas escogido pertenece a la clase de métodos denominados de Forzamiento Directo (Direct Forcing), en el cual una fuerza es introducida en las ecuaciones de cantidad de movimiento de Navier-Stokes para satisfacer los requerimientos de las condiciones de contorno del campo de velocidades.

Considere entonces la ecuación explícita de cantidad de movimiento, esto es:

$$\begin{aligned} \left[1 + (1-\alpha_u)\, \delta_P^{-1} \right] u^{*,k+1}_P = \alpha_u \left[u^n_P + \frac{\Delta t}{\rho \Delta V} \sum_{nb} A^u_{nb} u^{*,k}_{nb} - \frac{\Delta t}{\rho} \left(\left.\frac{\partial p}{\partial x}\right|^l_P - \tilde{S}^u_P \right) + \frac{\Delta t}{\rho \Delta V} f^k_P \right] \\ + (1-\alpha_u) \left(1 + \delta_P^{-1} \right) u^l_P \end{aligned} \tag{2.30}$$

donde $u^{*,k}_P \to u^*_P$, la condición inicial $u^{*,0}_P = u^l_P$ y $\tilde{S}^u_P$ engloba ahora los términos de fuerzas de cuerpo y correcciones diferidas. La idea es que la fuerza actualizada, f^{k+1}_P, será obtenida como:

$$f^{k+1}_P = f^k_P + \frac{\rho \Delta V}{\Delta t} \frac{\left[1 + (1-\alpha_u)\, \delta_P^{-1} \right]}{\alpha_u} \left(\bar{u}^{*,k+1}_P - u^{*,k+1}_P \right) \tag{2.31}$$

donde $\vec{u}_P^{k+1}$ es el campo de velocidades obtenido luego de aplicar el procedimiento de interpolación desarrollado en la sección previa. La iteración será efectuada hasta que alguna métrica sea satisfecha, por ejemplo, si ϵ_{rtol}^{ibm} es la tolerancia relativa, luego:

$$\frac{\left\|\mathbf{f}^{k+1}-\mathbf{f}^{k}\right\|_1}{\left\|\mathbf{f}^{1}\right\|_1} \leq \epsilon_{rtol}^{ibm} \tag{2.32}$$

considerando que $\mathbf{f^0} = \mathbf{0}$.

En este punto es importante remarcar que la Ecuación (2.30) es en realidad una versión reducida de una ecuación aún más fuerte en la cual existen dos iteraciones. En la primera el campo de velocidades $u_{nb}^{*,k}$ está fijo y la fuerza del método de fronteras embebidas es iterada hasta converger. Luego, en una segunda iteración, este valor del campo de velocidades es reintroducido en la misma ecuación para generar una nueva secuencia de campos de velocidades y fuerzas que van a converger eventualmente hasta el campo de velocidades esperado. Como la cantidad de esta iteraciones *internas* ha resultado no ser importante en la solución final (al menos no en las pruebas que se presentan en los capítulos posteriores), sólo una iteración por defecto es calculada, pudiendo modificarse la cantidad de iteraciones si fuera necesario (incrementando así el costo por iteración).

A efectos de obtener una ecuacion de corrección para la presión algunos conceptos van a ser introducidos a continuación. Primero que todo, el operador de interpolación utilizado para obtener el valor de una variable en las caras, y representado como $\overline{*|_i}^e$ para cualquier tipo de propiedad $*$ y aplicada en la cara este por ejemplo, estará definido formalmente como el valor medio entre las propiedades en los nodos P y E respectivamente. Esto es así puesto que en esta tesis se utilizaron grillas Cartesianas con paso espacial constante por dirección.

Las ecuaciones de cantidad de movimiento (en la dirección x, por ejemplo) en una cara, la este por ejemplo, pueden ser reescritas entonces como:

$$\begin{aligned}(1+\delta_e)\,u_e^* = \overline{(1+\delta_i)\,u_i^*}^e + \alpha_u \Delta t \left[\overline{\frac{\delta_i}{\rho}\left(\left.\frac{\partial p}{\partial x}\right|_i^l - S_i^u\right)}^e - \frac{\delta_e}{\rho}\left(\left.\frac{\partial p}{\partial x}\right|_e^l - S_e^u\right)\right] \\ + (1-\alpha_u)\left[(1+\delta_e)\,u_e^l - \overline{(1+\delta_i)\,u_i^l}^e\right] + \alpha_u\left[\delta_e u_e^n - \overline{\delta_i u_i^n}^e\right]\end{aligned} \tag{2.33}$$

El método SIMPLE iterará en cada paso de tiempo hasta que los campos de velocidad y presión satisfagan:

$$\begin{aligned}(1+\delta_e)\,u_e^{n+1} = \overline{(1+\delta_i)\,u_i^{n+1}}^e + \alpha_u \Delta t \left[\overline{\frac{\delta_i}{\rho}\left(\left.\frac{\partial p}{\partial x}\right|_i^{n+1} - S_i^u\right)}^e - \frac{\delta_e}{\rho}\left(\left.\frac{\partial p}{\partial x}\right|_e^{n+1} - S_e^u\right)\right] \\ + (1-\alpha_u)\left[(1+\delta_e)\,u_e^{n+1} - \overline{(1+\delta_i)\,u_i^{n+1}}^e\right] + \alpha_u\left[\delta_e u_e^n - \overline{\delta_i u_i^n}^e\right]\end{aligned} \tag{2.34}$$

La misma ecuación es válida en todas las caras dentro de una celda. En particular, en una cara oeste la expresión resulta:

$$(1+\delta_w)\,u_w^{n+1} = \overline{(1+\delta_i)\,u_i^{n+1}}^{w} + \alpha_u \Delta t \left[\overline{\frac{\delta_i}{\rho}\left(\left.\frac{\partial p}{\partial x}\right|_i^{n+1} - S_i^u \right)}^{w} - \frac{\delta_w}{\rho}\left(\left.\frac{\partial p}{\partial x}\right|_w^{n+1} - S_w^u \right) \right] + (1-\alpha_u)\left[(1+\delta_w)\,u_w^{n+1} - \overline{(1+\delta_i)\,u_i^{n+1}}^{w} \right] + \alpha_u \left[\delta_e u_w^n - \overline{\delta_i u_i^n}^{w} \right] \tag{2.35}$$

Vale la pena indicar en este punto existen muchos flujos, como en aquellos donde existe flotación, en los cuales el gradiente de presión y el término fuente tienen que estar en balance [Men12, Cho99]. En la formulación presente esta restricción se impone explícitamente, como puede ser visto en la Ecuación (2.34) y la Ecuación (2.35).

Ahora se introducirá el concepto de *corrección* [VM07]. SIMPLE establece que tanto la velocidad como la presión pueden ser escritos como la suma de un campo propuesto y una corrección. La corrección va a tender a cero conforme la iteración externa de SIMPLE converja. De esta forma, el campo propuesto convergerá a la solución final eventualmente.

En aras de simplicidad, considere que SIMPLE converje en sólo una iteración, esto es que $u^{n+1} = u^* + u'$, para el caso de la componente de velocidad en la dirección x. La ecuación de la corrección para el nodo P será entonces:

$$A_{P|P}^u u_P' = \sum_{j|P} A_j^u u_j' - \Delta V \left.\frac{\partial p}{\partial x}\right|_P' - \frac{(1-\alpha_u)}{\alpha_u} \tilde{A}_{P|P}^u u_P' - \frac{\rho \Delta V}{\Delta t} u_P' \tag{2.36}$$

y usando la aproximación de SIMPLE, en la cual $\sum_{j|P} A_j^u u_j' / \hat{A}_{P|P}^u \approx 0$, donde $\hat{A}_{P|P}^u = A_{P|P}^u / \alpha_u$, la Ecuación (2.36) puede ser reecrita como:

$$(1+\delta_P)\,u_P' = -\frac{\Delta V}{\hat{A}_{P|P}^u} \left.\frac{\partial p}{\partial x}\right|_P' = -\alpha_u \Delta t \frac{\delta_P}{\rho} \left.\frac{\partial p}{\partial x}\right|_P' \tag{2.37}$$

que es la ecuación de corrección necesaria para SIMPLE. Una expresión similar sería obtenida si se hubiera aplicado el mismo procedimiento a alguna cara de una determinada celda. Por ejemplo, la expresión de corrección para una cara este sería:

$$(1+\delta_e)\,u_e' = -\frac{\Delta V}{\hat{A}_{P|e}^u} \left.\frac{\partial p}{\partial x}\right|_e' = -\alpha_u \Delta t \frac{\delta_e}{\rho} \left.\frac{\partial p}{\partial x}\right|_e' \tag{2.38}$$

Entonces, al menos por el momento, la ecuación de cantidad de movimiento propuesta y su corrección presentan la misma estructura formal que la ecuación real de cantidad de movimiento aplicada al nodo P.

La ecuación de continuidad es la encargada de relacionar la Ecuación (2.34) y la Ecuación (2.35). La masa no puede ser creada ni destruida dentro de una celda (volumen de

control), por lo tanto en el caso de flujos incompresibles la sumatoria de los flujos másicos en una celda deben ser nulos. Para que esto sea así la misma consideración de un campo propuesto y una corrección es aplicada esta vez al campo de presión. Sin embargo, y con el objectivo de obtener un stencil compacto para la ecuación de Poisson para la presión y de esta forma poder resolver el sistema haciendo uso del método CG+FFT que se había desarrollado previamente, un factor de relajación para la presión deberá ser introducido.

Para demostrar este paso, suponga entonces una situación simplificada de un balance de masa 1D, es decir que considere sólo la Ecuación (2.34) y la Ecuación (2.35). Además se hará uso de la Ecuación (2.38) y su contraparte para la cara oeste. De esta forma se puede obtener una expresión como la siguiente:

$$\begin{aligned} 0 = \Re_u^* + \Re_u^n + \frac{\Delta t}{\rho}\Bigg\{ & \frac{\delta_w}{(1+\delta_w)}\left(\left.\frac{\partial p}{\partial x}\right|_w^{n+1} - S_w^u\right) - \frac{\delta_e}{(1+\delta_e)}\left(\left.\frac{\partial p}{\partial x}\right|_e^{n+1} - S_e^u\right) \\ & + \left[\frac{\delta_E}{2\,(1+\delta_e)}\left(\left.\frac{\partial p}{\partial x}\right|_E^{n+1} - S_E^u\right) - \alpha_u \frac{\delta_E}{2\,(1+\delta_e)}\left.\frac{\partial p}{\partial x}\right|_E'\right] \\ & + \left[\delta_P\left(\frac{1}{2\,(1+\delta_e)} - \frac{1}{2\,(1+\delta_w)}\right)\left.\frac{\partial p}{\partial x}\right|_P^{n+1} - \alpha_u \delta_P\left(\frac{1}{2\,(1+\delta_e)} - \frac{1}{2\,(1+\delta_w)}\right)\left.\frac{\partial p}{\partial x}\right|_P'\right] \\ & + \left[\frac{\delta_W}{2\,(1+\delta_w)}\left(\left.\frac{\partial p}{\partial x}\right|_W^{n+1} - S_W^u\right) - \alpha_u \frac{\delta_W}{2\,(1+\delta_w)}\left.\frac{\partial p}{\partial x}\right|_W'\right]\Bigg\} \end{aligned} \tag{2.39}$$

donde la contribución por el campo de velocidad actual, $\Re_u^*$, queda definida como:

$$\Re_u^* = \frac{(1+\delta_E)}{2\,(1+\delta_e)}u_E^* + \left[\frac{(1+\delta_P)}{2\,(1+\delta_e)} - \frac{(1+\delta_P)}{2\,(1+\delta_w)}\right]u_P^* - \frac{(1+\delta_W)}{2\,(1+\delta_w)}u_W^* \tag{2.40}$$

y la contribución por el campo de velocidades de la iteración previa, $\Re_u^n$, se define como:

$$\begin{aligned} \Re_u^n = & \underline{\frac{\delta_e}{(1+\delta_e)}u_e^n - \frac{\delta_w}{(1+\delta_w)}u_w^n} - \frac{\delta_E}{2\,(1+\delta_e)}u_E^n \\ & + \left[\frac{\delta_P}{2\,(1+\delta_w)} - \frac{\delta_P}{2\,(1+\delta_e)}\right]u_P^n + \frac{\delta_W}{2\,(1+\delta_w)}u_W^n \end{aligned} \tag{2.41}$$

Vale la pena mencionar en este punto que la contribución subrayada en la Ecuación (2.41) es el único término que necesita ser almacenado en cada iteración, además de los nuevos campos, por supuesto, pero los términos restantes pueden ser leidos y calculados con información disponible. Note que este término es importante para obtener el correcto estado estacionario de la ecuación (en caso que fuera necesario).

De esta forma considerando la Ecuación (2.39) resulta evidente que el factor de relajación para la presión tiene que ser igual al factor de relajación del campo de velocidades para obtener un stencil compacto, esto es $p^{n+1} = p^* + \alpha_u p'$.

La ecuación de Poisson para la presión puede ser escrita entonces como:

$$\frac{\alpha_u \Delta t \delta_e}{\rho\left(1+\delta_e\right)} \left.\frac{\partial p}{\partial x}\right|_e^{'} - \frac{\alpha_u \Delta t \delta_w}{\rho\left(1+\delta_w\right)} \left.\frac{\partial p}{\partial x}\right|_w^{'} = \Re_p^* + \Re_u^* + \Re_u^n \tag{2.42}$$

donde el miembro derecho (RHS) de la ecuación contiene la contribución de tres términos bien definidos. En particular, sólo resta uno por definir, el término de contribución del campo de presión previo, $\Re_p^*$, que se define de la siguiente manera:

$$\begin{aligned}\frac{\rho}{\Delta t}\Re_p^* =& \frac{\delta_w}{\left(1+\delta_w\right)}\left(\left.\frac{\partial p}{\partial x}\right|_w^* - S_w^u\right) - \frac{\delta_W}{2\left(1+\delta_w\right)}\left(\left.\frac{\partial p}{\partial x}\right|_W^* - S_W^u\right) \\ &- \frac{\delta_e}{\left(1+\delta_e\right)}\left(\left.\frac{\partial p}{\partial x}\right|_e^* - S_e^u\right) + \frac{\delta_E}{2\left(1+\delta_e\right)}\left(\left.\frac{\partial p}{\partial x}\right|_E^* - S_E^u\right) \\ &+ \left[\frac{\delta_P}{2\left(1+\delta_e\right)} - \frac{\delta_P}{2\left(1+\delta_w\right)}\right]\left(\left.\frac{\partial p}{\partial x}\right|_P^* - S_P^u\right)\end{aligned} \tag{2.43}$$

Aunque pueda parecer que esta ecuación de Poisson para la presión sea dependiente de α_u o Δt, o que no provee la correcta solución de estado estacionario, debe recordarse que fue derivada directamente de ecuaciones que sí cumplían con estas condiciones y por ende esta ecuación también las cumplirá.

Finalmente los residuos de la ecuación de cantidad de movimiento en la dirección x, se definen como:

$$\begin{aligned}{\Re_{\mathbf{u}}}^{l+1} = \Bigg\| &\left[1+\left(1-\alpha_u\right)\delta_P^{-1}\right] u_P^{l+1} \\ &-\left\{\alpha_u\left[u_P^n + \frac{\Delta t}{\rho \Delta V}\sum_{nb} A_{nb}^u u_{nb}^{l+1} - \frac{\Delta t}{\rho}\left(\left.\frac{\partial p}{\partial x}\right|_P^{l+1} - S_P^u\right)\right] + \left(1-\alpha_u\right)\left(1+\delta_P^{-1}\right)u_P^l\right\}\Bigg\|_1\end{aligned} \tag{2.44}$$

La iteración del método va a continuar hasta que se satisfaga una o más métricas. En particular, si ϵ_{atol} y ϵ_{rtol} fueran las tolerancias absolutas y relativas, entonces:

$${\Re_{\mathbf{u}}}^{l+1} < \epsilon_{atol} \quad \text{or} \quad \frac{{\Re_{\mathbf{u}}}^{l+1}}{{\Re_{\mathbf{u}}}^{1}} < \epsilon_{rtol} \tag{2.45}$$

donde ${\Re_{\mathbf{u}}}^{1}$ es el primer residuo. Como este residuo puede ser grande (y no representativo) algunos otros autores prefieren usar el residuo de la segunda iteración, la tercera, u alguna otra medida representativa.

2.5. Resolución de la ecuación de Poisson para la presión

Una característica interesante del método propuesto es que hace uso de Transformadas Rápidas de Fourier en el cálculo del precondicionador de la ecuación de Poisson para

la presión. La FFT es un algoritmo muy estudiado con una excelente implementación en CUDA provista por NVIDIA.

De esta manera, considere entonces el sistema lineal a resolver, esto es:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.46}$$

donde $\mathbf{x}$ es la corrección de la presión estudiada anteriormente.

La Transformada Discreta de Fourier (DFT) actuando sobre un vector $\mathbf{x}$ será definida como $\tilde{\mathbf{x}} = \mathbf{O}\mathbf{x} = \text{fft}(\mathbf{x})$, mientras que su aplicación inversa, la Transformada Discreta de Fourier Inversa (IDFT), se define como $\mathbf{x} = \mathbf{O}^T\tilde{\mathbf{x}} = \text{ifft}(\tilde{\mathbf{x}})$. Como puede observarse, la Transformada de Fourier es una transformación ortogonal, esto significa que $\mathbf{O}^{-1} = \mathbf{O}^T$.

Si la matriz del sistema lineal $\mathbf{A}$ es *homogénea*, esto es el stencil es el mismo en cada celda de la grilla y las condiciones de contorno son periódicas, luego puede demostrarse [BF01] que $\mathbf{O}$ diagonaliza a $\mathbf{A}$, esto es $\mathbf{D} = \mathbf{O}\mathbf{A}\mathbf{O}^{-1}$ donde $\mathbf{D}$ es una matriz diagonal cuyas entradas son los valores propios de $\mathbf{A}$. Es así como el cálculo del precondicionador puede proceder simplemente como:

$$\begin{gathered} \mathbf{A}\mathbf{x} = \mathbf{b} \\ \left(\mathbf{O}\mathbf{A}\mathbf{O}^{\mathbf{T}}\right)(\mathbf{O}\mathbf{x}) = (\mathbf{O}\mathbf{b}) \\ \mathbf{D}\tilde{\mathbf{x}} = \tilde{\mathbf{b}} \end{gathered} \tag{2.47}$$

y finalmente, si se considera un vector $\mathbf{z} \neq \mathbf{0}$, tal que todas sus componentes en la fft estén definidas, como por ejemplo la delta de Dirac, luego:

$$\tilde{\mathbf{z}} = \text{fft}(z)$$

$$\mathbf{y} = \mathbf{A}\mathbf{z}$$

$$\tilde{\mathbf{y}} = \text{fft}(y)$$

$$\underline{D_{ii} = \tilde{\mathbf{y}}_i / \tilde{\mathbf{z}}_i}$$

donde puede observarse que el cálculo de $\mathbf{D}$ es una operación extremádamente sencilla, que se realiza en $O(N\log(N))$ operaciones. Una vez que es calculado es utilizado dentro de un algoritmo de Gradientes Conjugados Precondicionado (PCG).

Un concepto clave a considerar cuando se hace uso del tratamientro de fronteras embebidas es que su uso permite incrementar la velocidad de cálculo reduciendo la cantidad de iteraciones necesarias para resolver la Ecuación (2.46). De hecho, como no se necesita satisfacer condiciones ni de contorno ni sobre la superficies de sólidos, el

método va a converger en sólo unas pocas iteraciones (aunque típicamente una iteración es suficiente). La principal diferencia respecto un trabajo previo [SPD+13], en donde fue dicho que sólo se necesitaba una iteración para converger a precisión de máquina, radica en el hecho que la Ecuación (2.42) no es más simétrica. De hecho, los términos δ_e y δ_w dependen del nodo P de la ecuación de advección-difussión, que al mismo tiempo depende directamente del campo de velocidades.

Sin embargo, aunque la ecuación de Poisson para la presión haya perdido la simetría, el PGC todavía brinda una solución relativamente buena. Un análisis profundo del tema puede ser encontrado en [S+94]. A modo de resumen, considere la forma cuadrática en x definida como:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathbf{T}}\mathbf{A}\mathbf{x} - \mathbf{b}^{\mathbf{T}}\mathbf{x} + c \tag{2.48}$$

donde c es un escalar de valor constante, si $\mathbf{A}$ es simétrica y definida positiva luego $f(\mathbf{x})$ es minimizada por la solución del problema de minimización $\mathbf{A}\mathbf{x} = \mathbf{b}$. Cuando $\mathbf{A}$ pierde simetría el PCG convergerá a la solución del problema de minimización $1/2(\mathbf{A}^{\mathbf{T}} + \mathbf{A})\mathbf{x} = \mathbf{b}$. Pruebas numéricas realizadas dan soporte a la idea de que en casos que la asimetría sea pequeña (como lo es en la mayoría de los casos analizados para esquemas explícitos como el utilizado en este trabajo) la solución obtenida por PCG es una solución válida. Si la solución real al sistema fuera necesaria, un método de Gradientes Bi-Conjugado Estabilizado (BICGstab) puede ser escogido para resolver el sistema, con un costo relativamente similar.

2.6. La ecuación de transporte de la temperatura

Siguiendo las ideas de las secciones previas, la ecuación de transporte para la temperatura puede ser escrita como:

$$T_P^{n+1,k+1}(\mathbf{x}) = T_P^n + \frac{\Delta t}{\rho \Delta V}\sum_{nb} A_{nb}^u T_{nb}^{n+1,k} + \frac{\Delta t}{\rho c_p} Q^k \tag{2.49}$$

donde el término fuente, que se utiliza para representar las condiciones de contorno del problema, es actualizado iterativamente como:

$$Q^{k+1} = Q^k + \frac{\rho c_p}{\Delta t}\left(\overline{T^{n+1,k+1}} - T^{n+1,k+1}\right) \tag{2.50}$$

Notar que este término de corrección tiene la misma estructura formal que el término introducido en las ecuaciones de cantidad de movimiento. Los criterios de convergencia por lo tanto son similares a los introducidos previamente.

2.7. Diagrama de flujo del algoritmo propuesto

Haciendo uso del Diagrama (2.2) se explicarán algunos de los conceptos estudiados previamente. Será fundamental comprender bien el esquema de iteraciones involucrados en cada etapa del algoritmo porque de ello dependerá su precisión y velocidad de cálculo.

Para empezar, el lazo que calcula las fuerzas requeridas por el método de fronteras embebidas, denominado como *lazo interno*, se encuentra adentro de un lazo mayor denominado el *lazo externo* de SIMPLE. Este lazo interno calculará la predicción del campo de velocidades (resolviendo las ecuaciones de cantidad de movimiento) calculando en cada iteración las fuerzas necesarias para satisfacer las condiciones de contorno. Resulta que a medida que las iteraciones externas convergen, la cantidad de iteraciones internas se reducen. Por lo tanto, una opción válida es fijar la cantidad de iteraciones internas y dejar que la convergencia la domine el lazo externo.

Como segundo punto importante, el precondicionamiento de la ecuación de Poisson para la presión no necesita ser recalculado en cada iteración externa. Como dicha iteración será utilizada para controlar la convergencia global del problema, el precondicionador podría ser calculado una sola vez, por ejemplo al comenzar la iteración externa, o bien cada n iteraciones externas. Más aún, la cantidad de iteraciones para resolver la ecuacion de Poisson para la presión puede ser también fijada y así dejar nuevamente que la convergencia la controle el lazo externo. Vale la pena mencionar que aunque la ecuación de Poisson para la presión no va a ser satisfecha a precisión de máquina, si es (correctamente) asumido que el campo de velocidades no varía demasiado entre sucesivas iteraciones, luego la solución siguiendo esta estrategia será lo suficientemente buena como para ser útil.

Una vez que los campos de presión y velocidad satisfagan las ecuaciones de cantidad de movimiento y continuidad, la ecuación de transporte para la temperatura será calculada. Podría ser interesante remarcar en este punto que se asumió un acoplamiento termodinámico débil. Si este no fuera el caso, con menores modificaciones se podría ubicar el calculo de la ecuación de transporte para la temperatura dentro del lazo de acople de presión-velocidad.

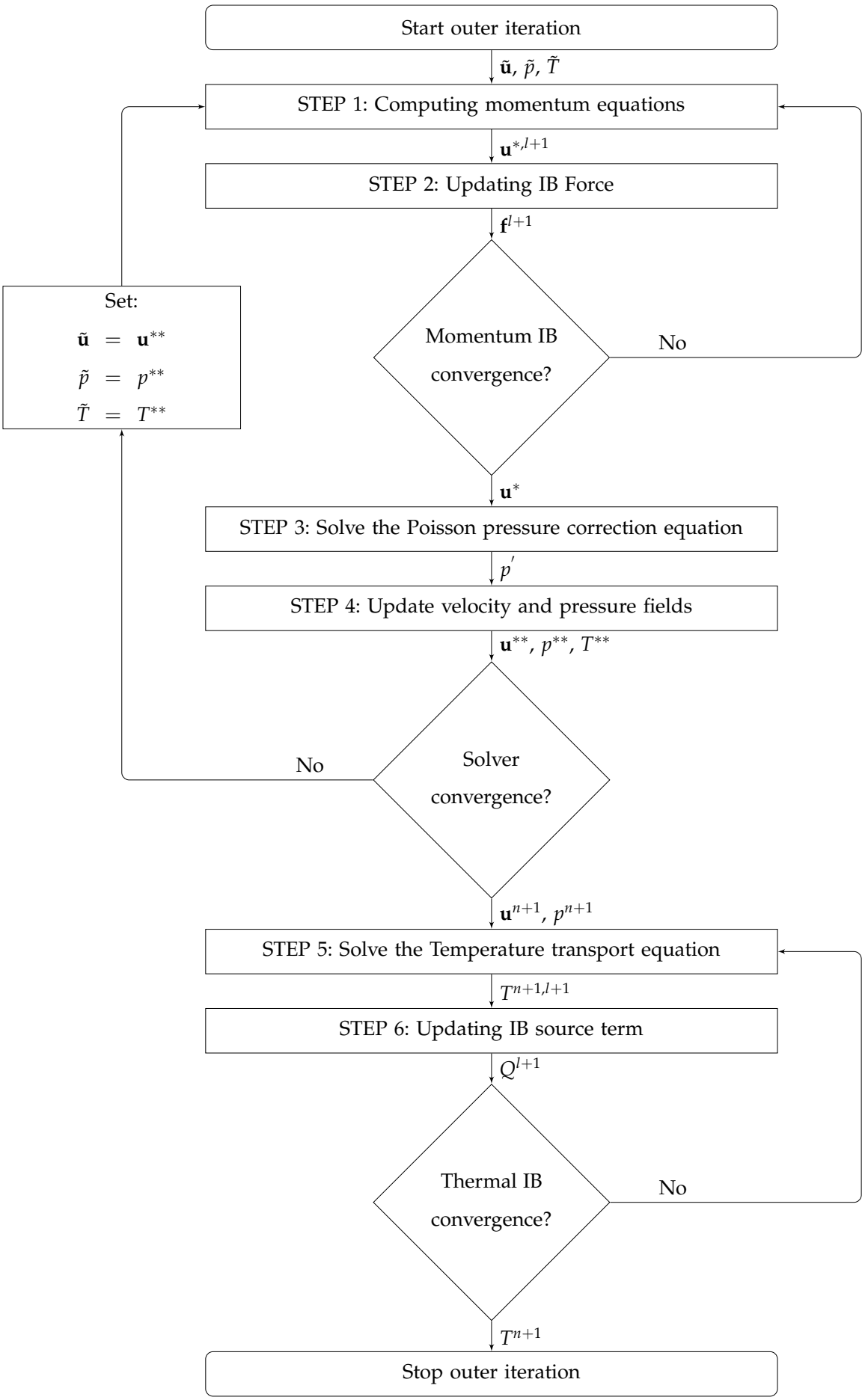


Figura 2.2: Diagrama de flujo del algoritmo SIMPLE propuesto.

Capítulo 3

Caso de estudio: interacción fluido-estructura

3.1. Configuración experimental

El experimento consiste en una esfera de silicona, de ahora en más una boya, con un diámetro $D = 0.1\,[\mathrm{m}]$ y densidad $\rho_s = 366.69\,\left[\mathrm{kg/m^3}\right]$ inmersa en un tanque cúbico de acrílico de dimensiones $0.38 \times 0.38 \times 0.4\,[\mathrm{m}]$ relleno con agua. Un hilo de longitud $0.16\,[\mathrm{m}]$ une la boya con la tapa inferior de la mesa. El tanque se encuentra montado sobre una tabla vibradora, Shake-Table II, que realiza movimientos oscilatorios a diferentes frecuencias impuestas. La evolución de la superficie libre es recortada usando una tapa agregada encima del tanque de forma tal que el tanque resulte en forma cúbica.

El objetivo es estudiar los movimientos de la boya bajo diferentes condiciones impuestas. Para tal fin, los experimentos son grabados usando una cámara AOS de alta velocidad y la secuencia de imágenes es analizada usando una técnica de captura de movimientos (motion capturing). La cámara está posicionada a una distancia perpendicular de $2.25\,[\mathrm{m}]$ del tanque y se utiliza una lente de $35\,[\mathrm{mm}]$. Con una resolución de 800×800 pixeles, el reservorio puede ser completamente apreciado. Tomando 120 imágenes por segundo es suficiente para capturar el movimiento de la boya. Como resultado, una precisión espacial de $0.5\,[\mathrm{mm}]$ y temporal de $0.008\,[\mathrm{s}]$ es obtenida. Con esta configuración aproximadamente 2000 imágenes pueden ser almacenadas cubriendo un tiempo total de $16.7\,[\mathrm{s}]$.

La amplitud experimental de la boya, A_{buoy}, y su fase, ϕ (corrimiento temporal entre el movimiento impuesto y la posición del centro de masa de la boya, convertido a grados) son

obtenidos durante el régimen periódico en el tiempo. Los desplazamientos impuestos en la tabla, A_{tank}, tienen una amplitud de 0.02 [m] y frecuencias, f, en el rango de $0.5 - 1.5$ [Hz], siendo 0.9 [Hz] la frecuencia (analítica) primaria de resonancia del sistema.

3.2. Consideraciones

El flujo puede ser caracterizado haciendo uso de dos números. El primero es el número de Keulegan-Carpenter, K, que relaciona las fuerzas de fricción (drag) e inercia en cuerpos romos oscilantes. Se define como $U_{max}/(fD)$ donde U_{max} es la velocidad máxima del cuerpo. El segundo es el número de Stokes, β, y se define como $D^2\rho f/\mu$ (inversamente relacionado con las fuerzas viscosas). Con base en los resultados experimentales K varia dentro del rango $0.4 - 5.6$ mientras que β lo hace entre $5000 - 15000$.

3.3. Ecuaciones de gobierno de cuerpo rígido

En el análisis actual el movimiento de la boya será reducido a una ecuación de un sólo grado de libertad (DOF) que represente el modelo de la boya sumergida que se muestra en la Figura (3.1). Se asume que la cuerda es recta, que no puede extenderse (ni elongarse) y que el centroide de la boya se encuentra alineado con la cuerda. También se asume (hipótesis que se desprende de los experimentos) que la boya no rota respecto al eje de la cuerda y que el movimiento de su centro está restricto al plano de movimiento (plano x-y de la Figura (3.1)). Por lo tanto, el desplazamiento del cuerpo queda reducido a un DOF, la deflexión angular del centro de la boya θ tomando como positiva la dirección de las agujas del reloj (clockwise). La ecuación de cantidad de movimiento angular correspondiente, escrita en un sistema de referencia no inercial fijo al tanque, es:

$$I_s\ddot{\theta} = m_s g L \sin\theta - m_s a_{tank} L \cos\theta + T_{fl} \tag{3.1}$$

donde I_s es el módulo de inercia de la boya con respecto al punto de fijación, m_s es la masa de la boya, g es la aceleración de la gravedad, a_{tank} es la aceleración dependiente del tiempo del tanque, L es la distancia desde el centro de gravedad al punto de fijación y T_{fl} es un término que representa a los torques de todas las fuerzas del fluido en la boya (positiva en dirección a las agujas del reloj) incluyendo la masa agregada, la fricción y las fuerzas de flotación por acción de los efectos inerciales. Notar que la Ecuación (3.1) considera la fuerza inercial en el sólido por acción de la aceleración del tanque.

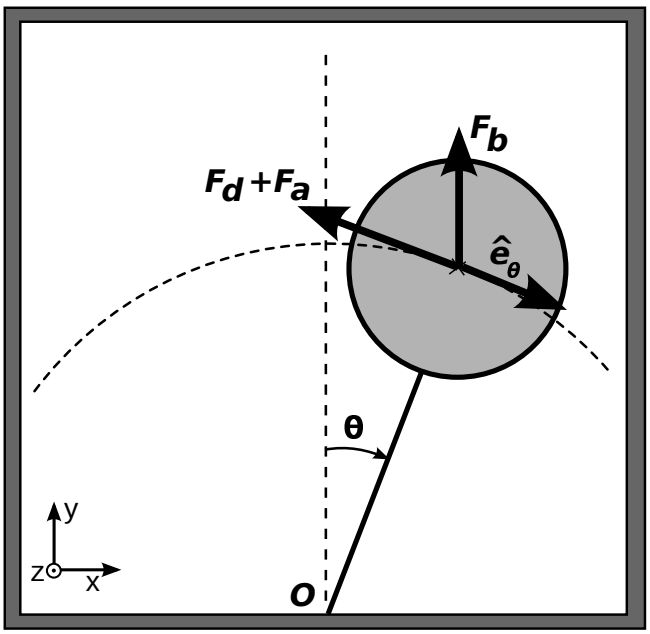


Figura 3.1: Esquema de las fuerzas de fluido actuando sobre la boya sumergida. Modelo simplificado de un DOF.

3.4. Solución analítica aproximada

Con el objetivo de estimar la frecuencia de resonancia, la masa agregada y el coeficiente de fricción, la Ecuación (3.1) puede ser aproximada con el siguiente modelo analítico reducido:

$$\begin{aligned} I_s\ddot{\theta} &= m_s g L \sin\theta - m_s a_{tank} L \cos\theta + T_{bg} + T_{ba} + T_a + T_d, \\ &= m_s g L \sin\theta - m_s a_{tank} L \cos\theta \\ &\quad - m_{\mathrm{fl}} g L \sin\theta + m_{\mathrm{fl}} a_{tank} L \cos\theta - m_a L^2 \ddot{\theta} - 1/2 \rho_f |v|^2 \,\mathrm{sign}(v)\, C_d A_{\mathrm{r}} L \end{aligned} \tag{3.2}$$

donde $v = L\dot{\theta}$ es la velocidad tangencial, T_{bg} y T_{ba} son los torques generados por la gravedad y la aceleración del tanque respectivamente, T_a es el toque por la masa agregada y T_d es el torque producto de la fuerza de fricción. Expresiones estándares fueron asumidas para las fuerzas de fricción y masa agregada. Además m_{fl} es la masa del fluido desplazado por la boya, m_a es la masa agregada, $A_{\mathrm{r}} = \pi D^2/4$ es el área proyectada de la boya y C_d es el coeficiente de fricción. La masa agregada es usualmente adimensionalizada definiendo el coeficiente de masa agregada $C_a = m_a/m_{\mathrm{fl}}$.

Bajo la suposición de pequeños desplazamientos, $\theta \ll 1$, una expresión cerrada para la frecuencia de resonancia puede ser obtenida:

$$2\pi f_n = \sqrt{\left(\frac{m_{\mathrm{fl}} - m_s}{m_s + m_a}\right)\frac{g}{L}}. \tag{3.3}$$

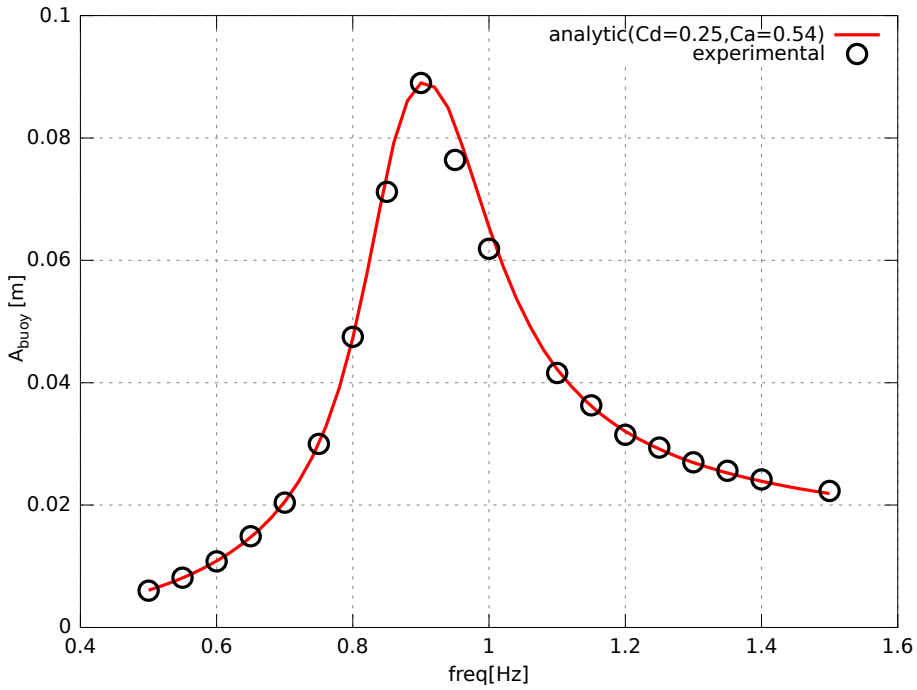
Valores analíticos, numéricos y experimentales para C_d y C_a son reportados en la literatura. Sin embargo C_d es usualmente reportado para una condición de corriente libre estacionaria. Con respecto a la masa agregada, es bien conocida la solución para flujos potenciales (un medio de la masa desplazada para el caso de una boya). Sin embargo, el análisis actual se encuentra lejos de esas situaciones ideales, por ejemplo la velocidad no es constante y el dominio de análisis esta restricto al tanque (no es una condición de corriente libre). Por lo tanto la masa agregada y el coeficiente de fricción son desconocidos para este presente caso de estudio y fueron determinados minimizando el error entre los valores experimentales y analíticos obtenidos examinando las curvas de máximo desplazamiento de la boya en función de la frecuencia de la mesa durante el régimen periódico, como lo muestra la Figura (3.2).

Dado C_d y m_a, la Ecuación (3.2) es resuelta con una técnica estándar de discretización de ecuaciones diferenciales ordinarias (ODE) con un movimiento impuesto del tanque con amplitud $A_{\text{tank}} = 0.02\,[\text{m}]$ y frecuencia f en el rango entre $0.5 - 1.5\,[\text{Hz}]$, como fue reportado previamente. De este análisis los valores máximos de desplazamiento de la boya durante el régimen periódico son determinados y graficados en función de la frecuencia impuesta. La masa agregada y el coeficiente de fricción que mejor representan a las curvas experimentales resultaron $C_a = 0.54$ y $C_d = 0.25$ respectivamente.

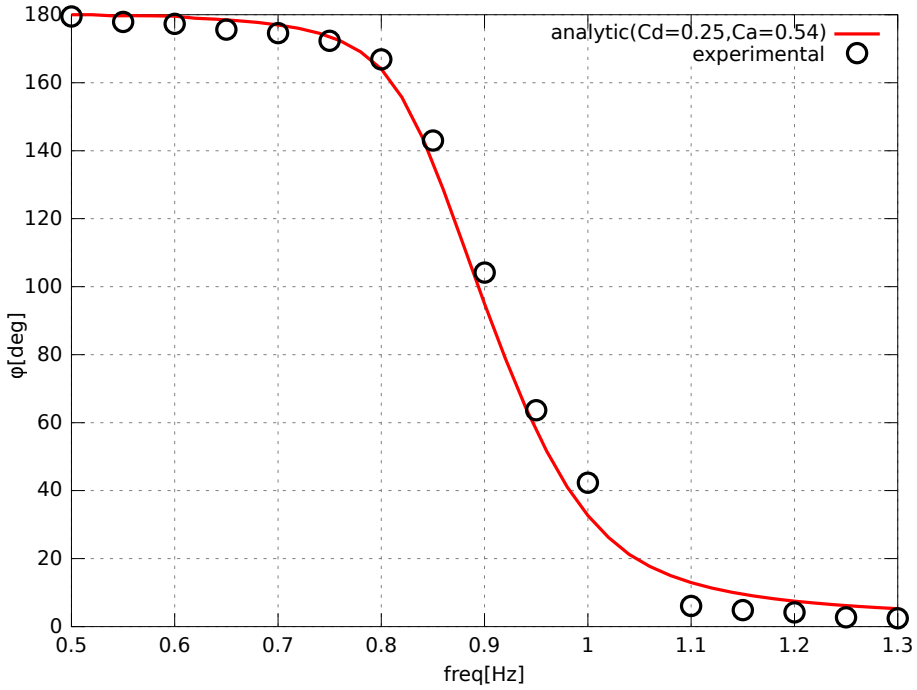
La respuesta en fase del sistema computada con el modelo analítico usando los valores calculados ($m_a = 0.54 m_{\text{fl}}$, $C_d = 0.25$) se muestra en la Figura (3.2), como una verificación.

Es importante mencionar la influencia en la respuesta del sistema, considerando la Figura (3.2), de los diferentes términos presentes en el modelo reducido, es decir, la Ecuación (3.2). En dicha ecuación el término que presenta mayores complicaciones para ser calculado resulta el de la fuerza de fricción, seguido por el de la masa agregada y finalmente el de flotación. En esquemas embebidos, como el utilizado en el presente trabajo, es particulamente difícil capturar la fuerza de fricción dado al efecto de representación de la superficie del sólido (o interfaz sólido-fluido).

A bajas frecuencias ($f < 0.8\,[\text{Hz}] < f_n$), el término dominante es el de flotación por gravedad que actua como un término restitutivo, mientras que la masa agregada y la fuerza de fricción (amortiguamiento) no juegan un papel importante. A mayores frecuencias ($f > 1.0\,[\text{Hz}] > f_n$), la masa agregada es el término dominante por lo cual si el esquema numérico falla en ajustar los valores experimentales significará que la masa agregada no fue correctamente capturada. Finalmente, para frecuencias cercanas a la resonancia ($0.8\,[\text{Hz}] < f < 1.0\,[\text{Hz}]$) las fuerzas restitutivas y de masa agregada



(a) Amplitud.



(b) Fase.

Figura 3.2: Resultados analíticos vs experimentales.

se cancelan y la máxima amplitud queda gobernada sólo por la fuerza de fricción. Es por esto que una prueba importante para los métodos embebidos es que debe describir correctamente la curva cerca del punto de resonancia. Por el otro lado, la frecuencia de resonancia depende directamente de la masa agregada según la Ecuación (3.3). En el análisis presente la frecuencia de resonancia computada con los parámetros previamente

calculados es $f_n = 0.91$ [Hz].

3.5. Modelo 2D de un cuerpo rígido con movimiento prescripto

Como ya fue mencionado, el objetivo de este ejemplo es validar el cómputo de las fuerzas inducidas sólo por el fluido haciendo uso de la técnica de fronteras embebidas propuesta en comparación con un código bien conocido de elementos finitos (FEM/ALE). La configuración geométrica y los parámetros físicos tratarán de representar las condiciones del caso experimental. Note que este ejemplo no resolverá FSI dado que el movimiento del cuerpo será impuesto, pero ayudará a validar las fuerzas de fluido en el sólido usando diferentes técnicas numéricas.

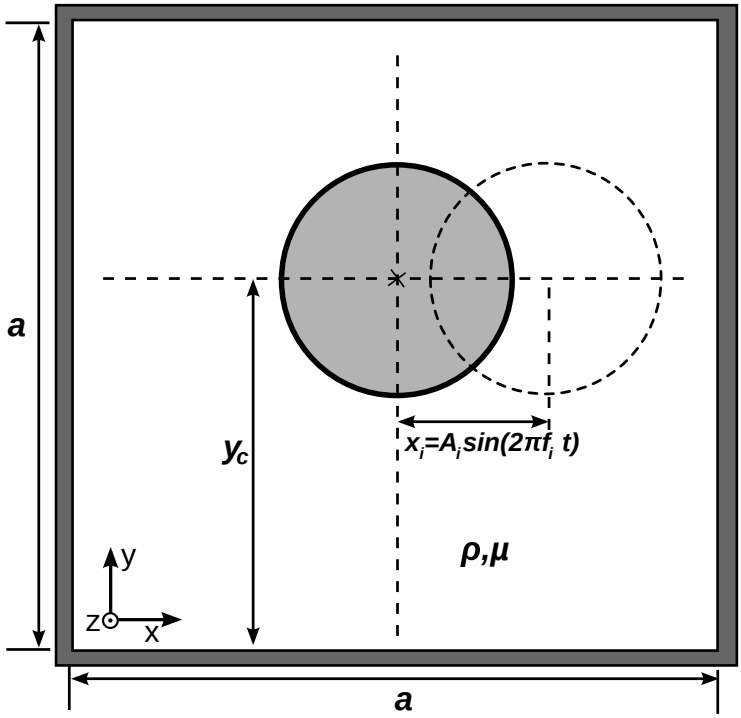


Figura 3.3: Ejemplo de cilindro oscilante. Descripción geométrica.

La Figura (3.3) muestra el dominio computacional mientras que la Figura (3.4) muestra la malla conforme que fue usada para el cálculo FEM/ALE. La malla más gruesa esta compuesta por 16384 nodos con un paso de malla de 1 [mm] en la dirección normal a la superficie del cuerpo, mientras que la malla más refinada presenta 65536 nodos con un paso de malla en dirección a la normal de 0.3 [mm].

Para el cálculo FVM/IBM una discretización homogénea de 512 celdas en cada dirección

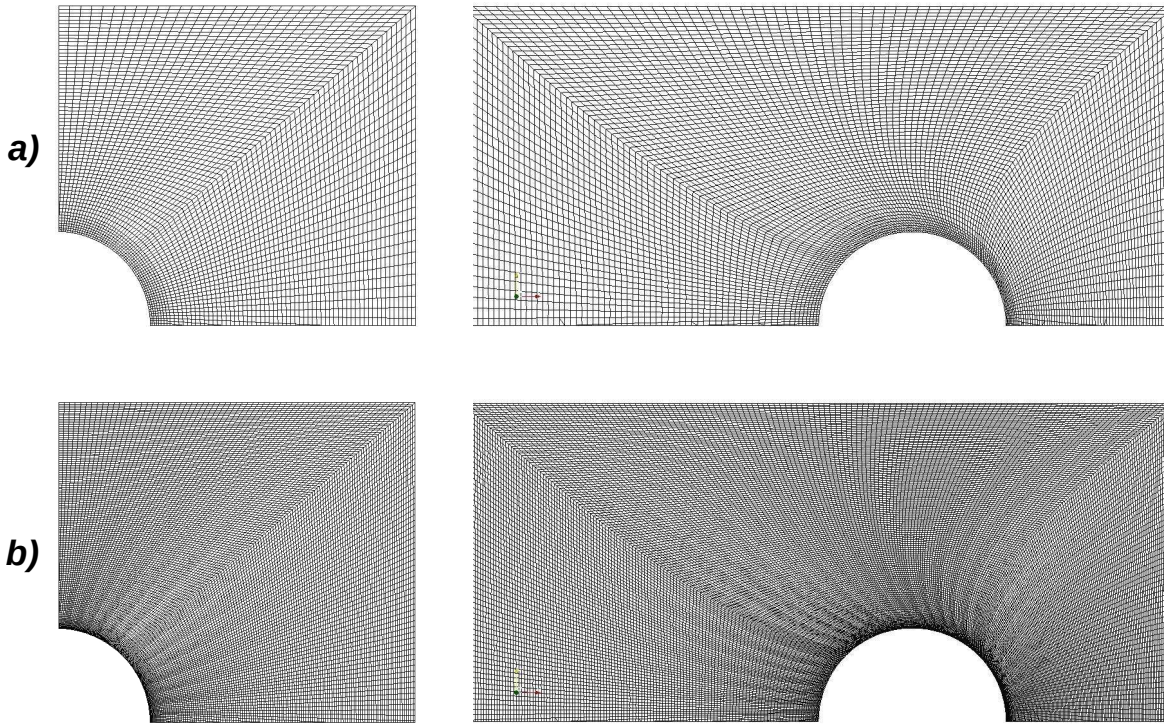


Figura 3.4: Mallas FEM utilizadas para los cálculos. a) Malla gruesa, izquierda: original (sólo se muestra un cuarto de la malla), y derecha: malla deformada durante el máximo desplazamiento (mitad superior de la malla). b) Lo mismo para la malla más fina.

fue adoptada, dando un total de 262144 celdas. El paso de malla para este caso fue de 0.75 [mm], dimensiones comparables a las mallas FEM utilizadas.

Además de las propiedades físicas previamente mencionadas, dos viscosidades dinámicas fueron consideradas $\mu = 0.05\,[\mathrm{kg/ms}]$ y $\mu = 0.001\,[\mathrm{kg/ms}]$. Como se ha dicho previamente, este tipo de flujos se encuentra caracterizado en la literatura [BDGO85] por dos números adimensionales: el número de Keulegan-Carpenter $K = U_{\max}/(f_i D) = 2\pi A_i/D$, donde $U_{\max} = 2\pi f_i A_i$ es la velocidad máxima del cuerpo, y el número de Stokes $\beta = D^2 \rho f_i/\mu$. Este ejemplo considera $K = 3.14$ y β tomando valores entre 180 y 9000, que se obtienen haciendo uso de las dos viscosidades definidas al comienzo del párrafo y un número de Reynolds basado en la velocidad máxima ($\Re = U_{\max} D\rho/\mu = K\beta$) de $\Re = 565.5$ y 28274 respectivamente. Notar que aunque el valor obtenido en el número de Reynolds parece alto, no debe ser directamente asimilado a un valor constante de un campo de velocidades del mismo número de Reynolds, dado que las condiciones del flujo varían durante un típico período. Esto puede promover la idea de que la turbulencia no sea desarrollada para los instantes cortos de tiempo donde se tienen estos números de

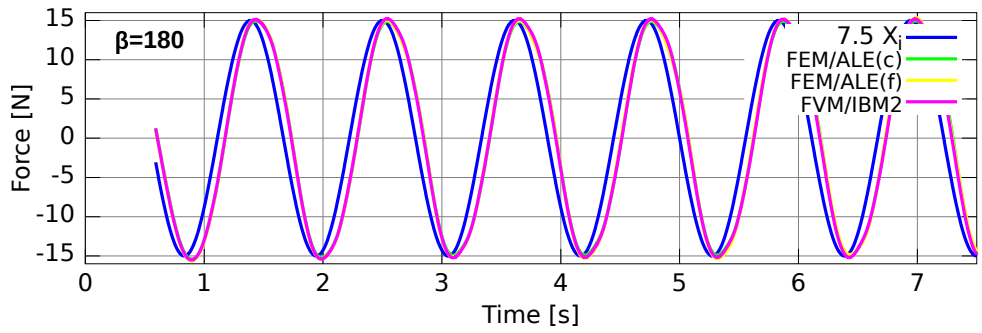
Reynolds, por ejemplo en [BDGO85] la capa límite se asume como laminar al menos hasta $\beta = 1665$. Tomando estas consideraciones se decidió no incluir un modelo de turbulencia. Sin embargo, considerando los resultados satisfactorios obtenidos en este trabajo con respecto a aquellos experimentales [S+76, Sar86, BDGO85] sobre los que se tiene conocimiento, se puede concluir que esta suposición es correcta.

Para el cálculo FEM/ALE se usó un paso de tiempo $\Delta t = 0.002\,[\mathrm{s}]$, mientras que en FVM/IBM se usó $\Delta t = 0.0004\,[\mathrm{s}]$. Respecto al bucle SIMPLE se utilizó una sóla iteración interna ($N_{IBM} = 1$) y una tolencia para el lazo externo de 1×10^{-4} (típicamente $N_{simple} = 3$ iteraciones).

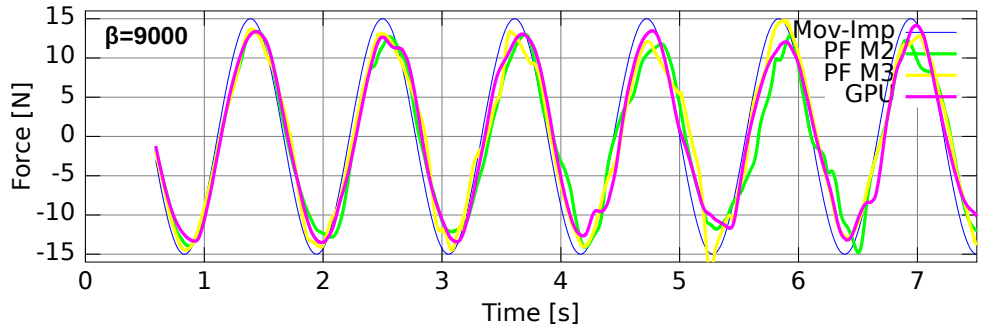
Con el objetivo de comparar los dos métodos, se introducirá el trabajo realizado por el fluido en el sólido, esto es:

$$W(t) = \int_0^T Fv \, \mathrm{d}t \tag{3.4}$$

donde $v = 2\pi f_i A_i \cos(2\pi f_i t)$ es la velocidad horizontal impuesta y F es la fuerza horizontal del fluido reportada en la Figura (3.5). El trabajo así obtenido se muestra en la Figura (3.6).



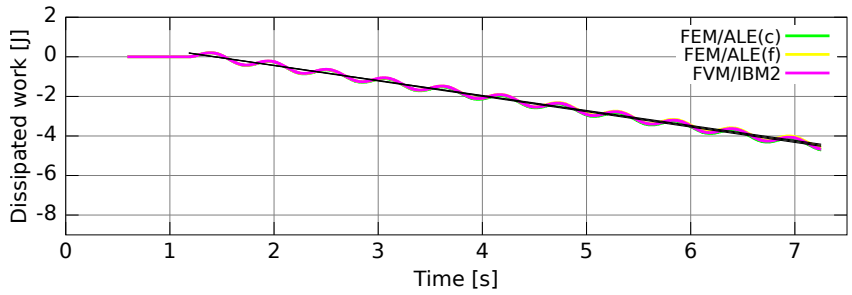
(a) $\beta = 180$



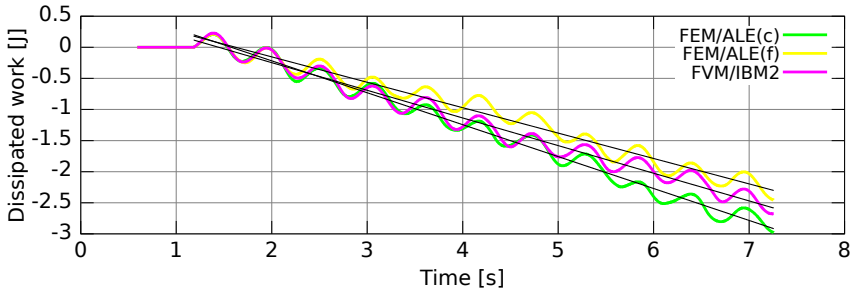
(b) $\beta = 9000$

Figura 3.5: Fuerzas de fluido horizontales en el cuerpo cálculadas para $K = 3.14$.

Haciendo uso de las resultados presentados en la Figura (3.6) se puede calcular la potencia disipada por el fluido en un período $\bar{P}_n = (W((n+1)T) - W(nT))/T$. En la



(a) $\beta = 180$



(b) $\beta = 180$

Figura 3.6: Trabajo realizado por las fuerzas del fluido sobre el cuerpo para $K = 3.14$. Las líneas rectas son regresiones lineales usadas en el cálculo de $\bar{P}$ y C_d.

Tabla (3.1) el promedio de $\bar{P}_n$ sobre un conjunto de ciclos en presentado. Otra posibilidad consiste en computar la pendiente promedio de $W(t)$ con una regresión lineal sobre la misma cantidad de ciclos, valor que también se reporta en dicha tabla. Las diferencias entre estos valores representan las fluctuaciones en las fuerzas. La disipación promedio puede hacerse adimensional en términos del coeficiente de fricción C_d. Asumiendo una expresión estándar para la fuerza, $F = -m_a\dot{v} - 1/2\rho_f v|v|\, C_d D$, y reemplazando esta expresión en la Ecuación (3.4) se obtiene:

$$\begin{aligned}\bar{P} &= -\frac{1}{T}\int_0^T 1/2\rho_f D {A_i}^3 (2\pi f_i)^3 C_d |\cos(2\pi f t)|^3 \mathrm{d}t \\ &= -\frac{16}{3}\pi^2 f_i^3 \rho_f D {A_i}^3 C_d\end{aligned} \tag{3.5}$$

desde la cual un valor para el coeficiente C_d puede ser calculado en términos de la disipación promedio. Note que la fuerza de masa agregada no produce disipación. Más aún, para este problema la fuerza de masa agregada es mucho mayor que la fuerza de fricción y por lo tanto se hace muy difícil desacoplar los dos componentes de la fuerza. Luego la tasa de disipación es un buen indicador de la fuerza de fricción, que de hecho es la componente de fuerza más difícil de obtener precisamente en el cálculo. Estos valores de C_d son también reportados en la Tabla (3.1).

	$\beta = 180$				$\beta = 9000$			
	$\bar{P}$[1]	C_d [1]	$\bar{P}$[2]	C_d [2]	$\bar{P}$[1]	C_d [1]	$\bar{P}$[2]	C_d [2]
FVM/IBM2	0.770	1.604	0.770	1.604	0.445	0.926	0.445	0.926
FEM/ALE(c)	0.752	1.565	0.776	1.616	0.543	1.132	0.503	1.047
FEM/ALE(f)	0.760	1.583	0.781	1.626	0.361	0.752	0.406	0.845

Cuadro 3.1: Disipación promedio y coeficiente de fricción computados como: [1] promedio del ciclo de trabajo, [2] regresión lineal del trabajo. (c) y (f) indican mallas gruesas y finas respectivamente.

En la Figura (3.7) se observa la solución a diferentes instantes de tiempo para $\beta = 180$. La primer fila se corresponde con $t = 0.8\,[\mathrm{s}]$ (aproximadamente $0.72\,T$, cerca del máximo desplazamiento a la izquierda). Las siguientes cinco imágenes son mostradas cada $T/8\,[\mathrm{s}]$ partiendo desde $6.5\,T$, representando un semi ciclo desde la posición central, desplazándose hacia el extremo izquierdo para luego retornar nuevamente a la posición central. Note que en las primeras dos columnas el sólido se observa en blanco dado que la malla es conforme. Por otro lado, en la tercer columna (que se corresponde con el caso FVM/IBM2), el cuerpo tiene una tono predominantemente azul que hace referencia al flujo parásito dentro del sólido, que principalmente esta en reposo exceptuando una capa muy fina cercana a la superficie del sólido.

Un grupo de imágenes con características similares al anterior caso, esta vez para $\beta = 9000$, se muestra en la Figura (3.8). Desafortunadamente las imágenes ya no son directamente comparables dado al comportamiento caótico de los vórtices desprendidos a este alto valor de β, aún entre las distintas soluciones FEM/ALE. Sin embargo el aspecto cualitativo del flujo es similar y los valores cuantitativos de fuerza, potencia disipada y los correspondientes coeficientes de fricción reportados en la Tabla (3.1) coinciden.

3.6. Modelando el experimento

Un análisis de convergencia en malla se reporta en la Figura (3.9), incluyendo las predicciones obtenidas para la amplitud a la frecuencia de resonancia usando los esquemas de volúmenes finitos de primer (FVM/IBM1) y segundo (FVM/IBM2) orden. La frecuencia de resonancia es escogida puesto que es la que presenta mayor dificultar para ser replicada.

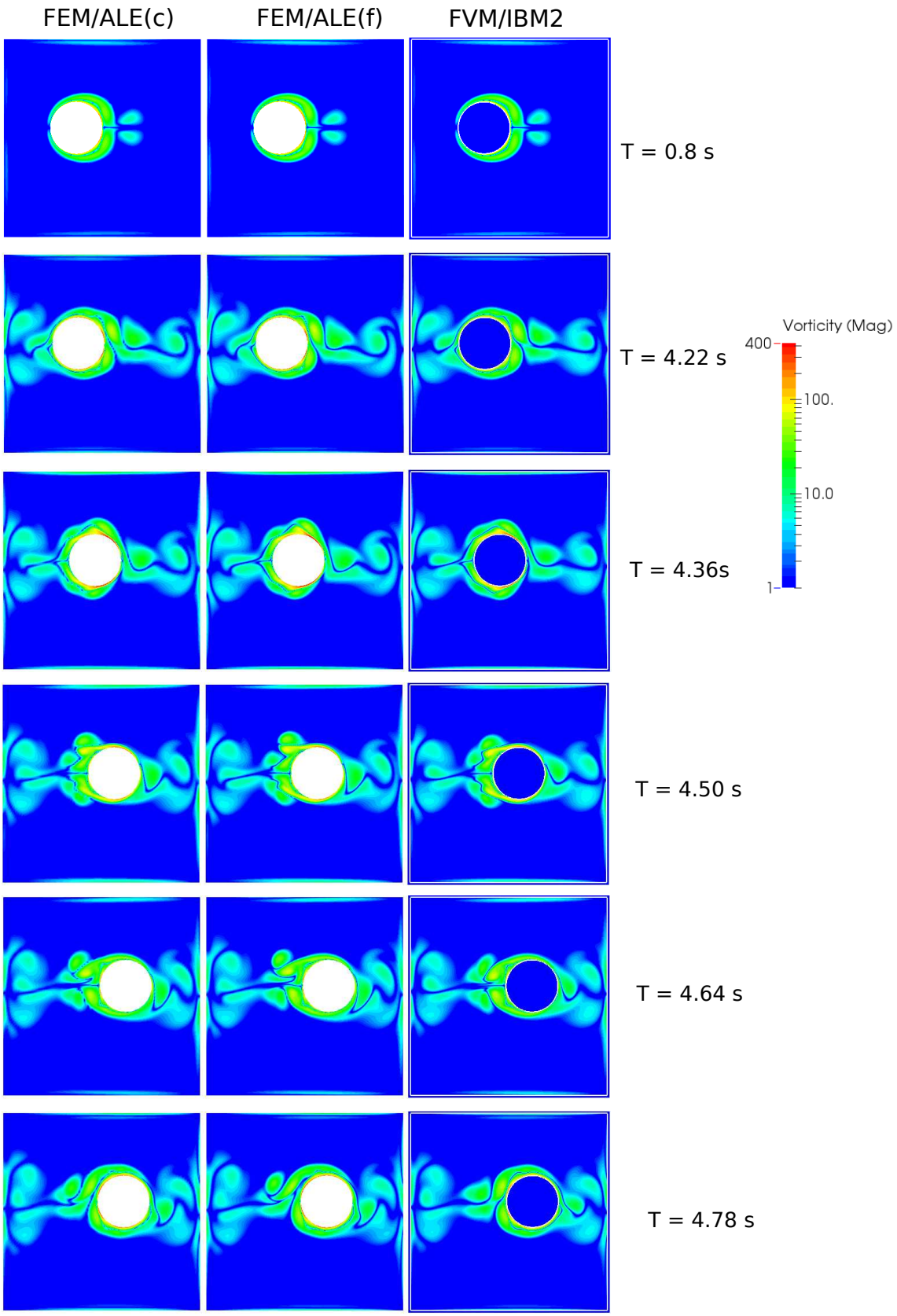


Figura 3.7: Mapa de colores de la vorticidad para el caso del cilindro oscilante, $\beta = 180$.

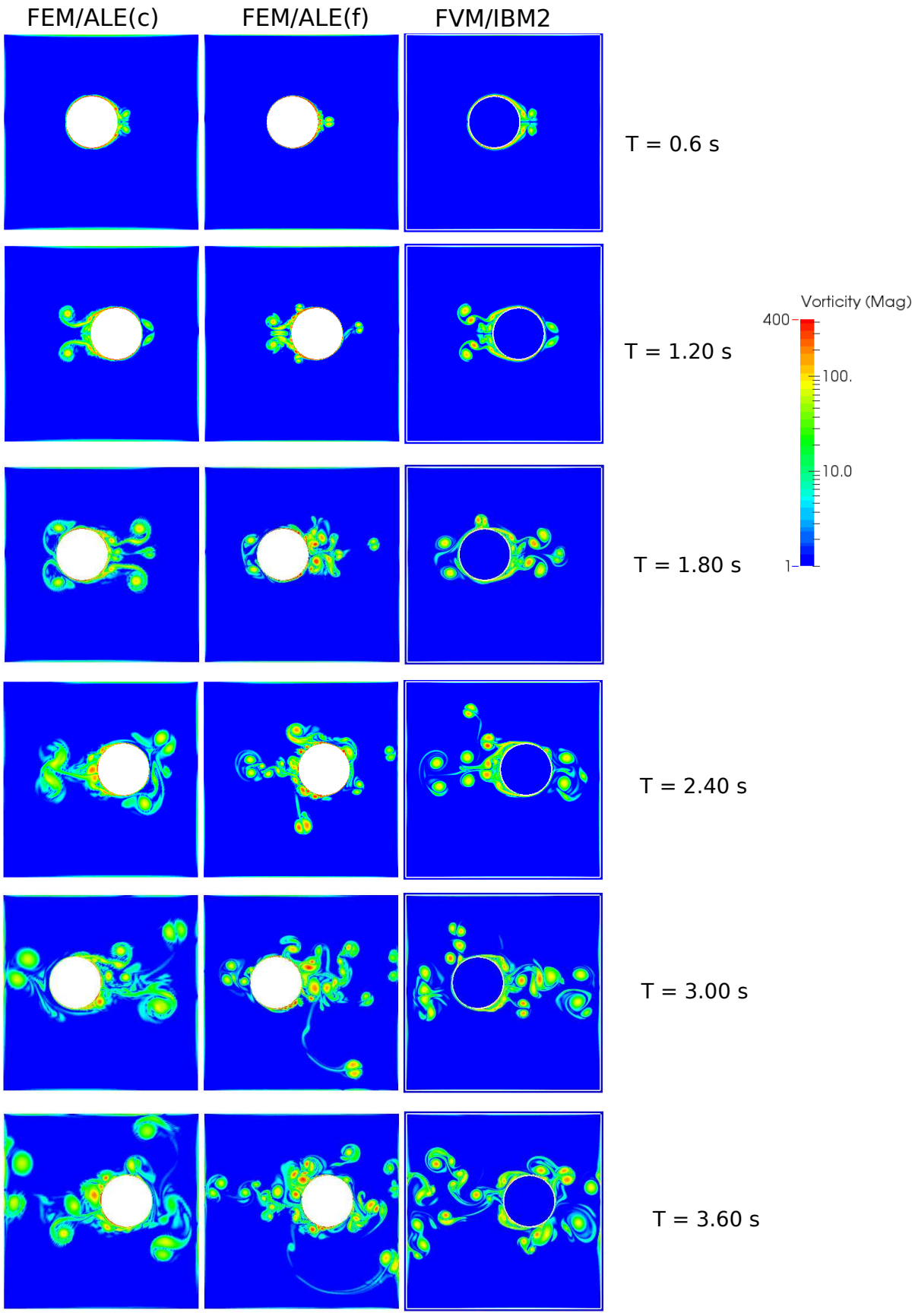


Figura 3.8: Mapa de colores de la vorticidad para el caso del cilindro oscilante, $\beta = 9000$.

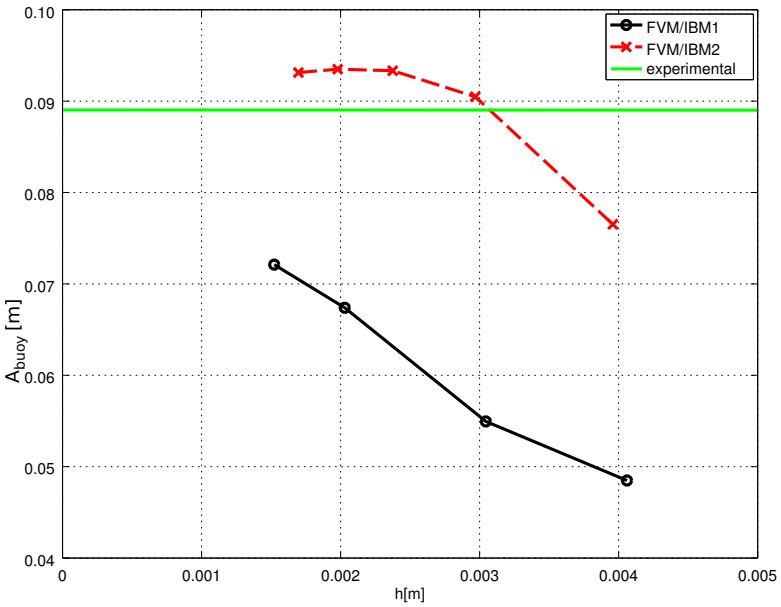
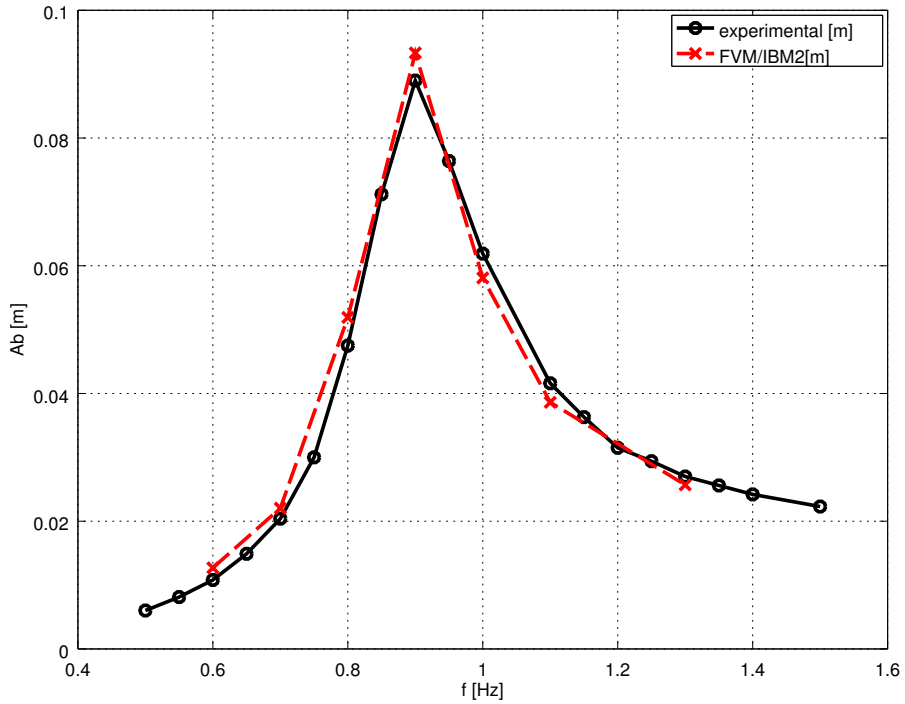


Figura 3.9: Análisis de convergencia en malla para los esquemas FVM/IBM de primer y segundo orden a $f = 0.9\,[\mathrm{Hz}]$.

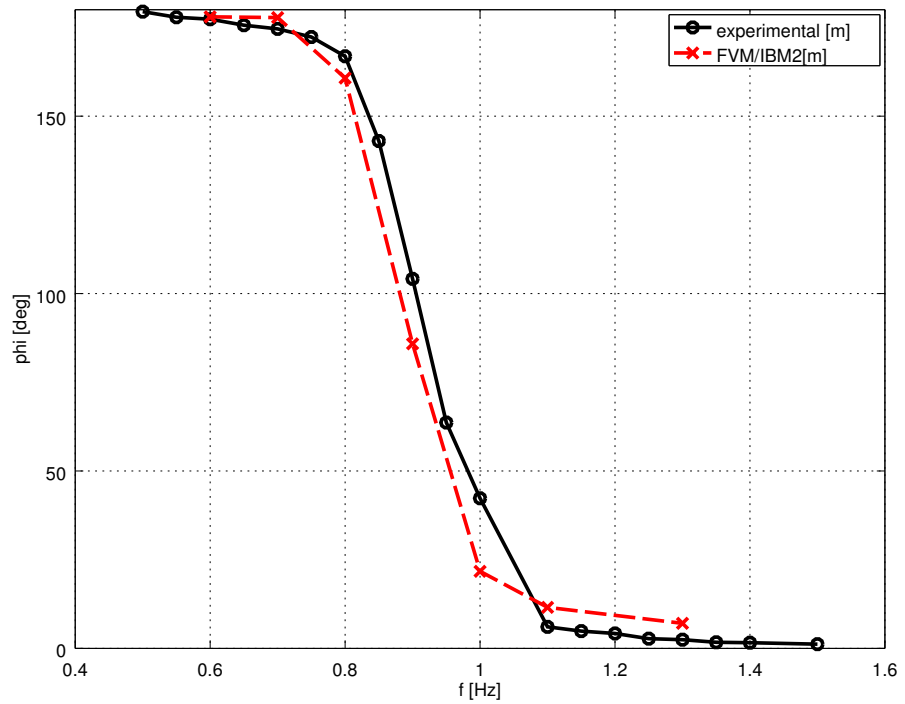
El esquema de primer orden presenta convergencia lineal, mientras que el esquema de segundo orden presenta convergencia cuadrática. Para la malla más gruesa (con 96 celdas por dirección) un error del 45 % es observado respecto al valor experimental con el esquema de primer orden, sin embargo ese error se reduce a 14 % cuando se utiliza el de segundo orden. Para una malla más fina (160 celdas por dirección) los errores se reducen a 14 % y 5 % respectivamente. Estos resultados confirman el buen comportamiento del esquema de segundo orden propuesto.

La Figura (3.10) resume los resultados de las amplitudes y de las fases obtenidas a diferentes frecuencias impuestas haciendo uso del esquema de segundo orden en la malla más fina (192 celdas por dirección, dando un total de 7.08 [Mcell]), mostrando una excelente correlación con los datos experimentales. La amplitud en la frecuencia de resonancia, que es muy sensible a la fuerza de fricción, es de 0.0931 [m], un 4.5 % mayor al encontrado experimentalmente. Una de las razones por la cual esto pudiera estar ocurriendo podría ser que no todos los DOF del sistema esta siendo considerados. Generalmente hablando, en un sistema mecánico su capacidad de disipación de energía será mayor a medida que se incrementen sus DOF, por lo cual si esto ocurriese la amplitud sería menor.

Usando el modelo analítico reducido presentado en secciones precedentes estos resultados pueden ser introducidos para obtener un coeficiente de fricción $C_{d,\mathrm{num}} = 0.23$, el cual se encuentra muy cerca del valor experimental $C_{d,\mathrm{exp}} = 0.25$, mientras que ambos se encuentran lejos del valor estándar de $C_d = 0.47$ reportado en la literatura [PWR11]



(a) Amplitud.



(b) Fase.

Figura 3.10: FVM/IBM2 vs. resultados experimentales.

Una colección de imágenes, mostrando isosuperficies de vorticidad para el caso $f = 0.9\,[\text{Hz}]$, se muestra en la Figura (3.11). Las nueve imágenes se corresponden a los instantes de tiempo $t = (9 + k/8)T\,[\text{s}]$ con $0 \leq k \leq 8$, esto es el décimo ciclo comenzando en $t = 9T\,[\text{s}]$ y efectuando un período completo a intervalos de $T/8\,[\text{s}]$.

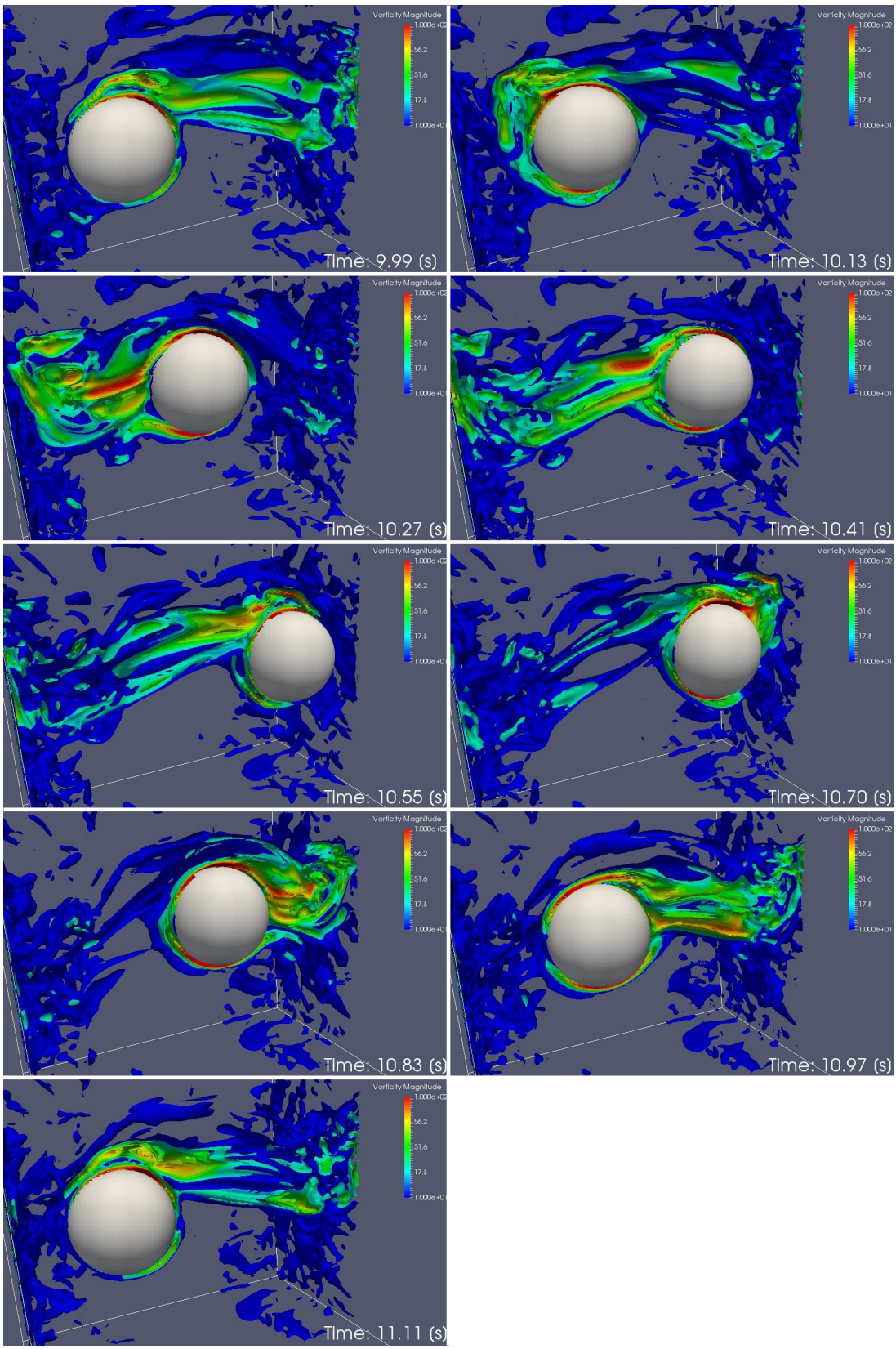


Figura 3.11: Isosuperficies de vorticidad comenzando en $t = 9T$ [s] con un espaciamiento de $T/8$ [s] agrupados linealmente en 10 conjuntos desde $\omega = 10$ a 100 $[s^{-1}]$, donde ω en la norma del vector vorticidad.

3.7. Implementación GPGPU

Las principales características del método GPGPU propuesto serán resumidas en esta sección. En este punto vale la pena indicar que a lo largo de estos años la arquitectura CUDA ha sufrido múltiples modificaciones y que una gran cantidad de *mejores prácticas* (best practices) han cambiado de generación en generación volviendo algunas de ellas en *malas prácticas* y viceversa. Existen muchos ejemplos que lo demuestran, pero uno de los más importantes reside en el uso de la memoria compartida (aunque ha simplificado la tarea de programación).

A efectos de mejorar la eficiencia, el método GPGPU propuesto engloba los siguientes aspectos:

- El código se encuentra separado en varios kernels pequeños reduciendo así la cantidad de registros por hilo (thread).
- Se hace uso de la memoria global y no de la memoria compartida.
- Se usan bloques tridimensionales, con alineamientos que traten de utilizar la información almacenada contiguamente.
- Para reducir la cantidad de registros al mínimo algunas operaciones idénticas son redundantemente computadas en vez de almacenar sus resultados en registros [CCLW11]. En la práctica esto es realizado reemplazando el valor de una variable intermedia por una macro.

Con estas modificaciones se obtuvo una ganancia del 400 % respecto a una versión sin esas modificaciones. La Tabla (3.2) muestra los tiempos relativos de cada etapa del algoritmo propuesto en una NVIDIA K40c y una GTX 580 Tital Black, para un barrido de 160 a 224 celdas por dirección con $N_{simple} = 3$ y $N_{IBM} = 1$.

Stage	IBM Loop	Poisson	Forces	Level Set	Other
Time (%)	60 %	35 %	4 %	0.5 %	0.5 %

Cuadro 3.2: Tiempos de cálculo relativos de las diferentes etapas del método propuesto.

En casos donde una geometría compleja necesita ser analizada, el tiempo computacional para resolver la ecuación de LS con un algoritmo basado en PDE puede estimarse como

el tiempo requerido para computar una ecuación de cantidad de movimiento, así podría representar alrededor del 7 % en el contexto actual. La etapa de reinicialización será necesaria, por lo cual estimando su costo e incorporando su frecuencia, se llega a la conclusión que se tendría un costo extra del 20 %. Note que, la incidencia del cálculo de la ecuación de LS depende fuertemente de la cantidad de iteraciones utilizadas dentro del algoritmo SIMPLE. Si fueran necesarias más iteraciones dentro de SIMPLE el costo relativo de cálculo de la ecuación de LS sería menor. Lo mismo aplica si se hiciese uso de estrategias para incrementar el paso de tiempo, por ejemplo usando una estrategia semi Lagrangiana [CSP+14].

3.7.1. Tiempos de cálculo

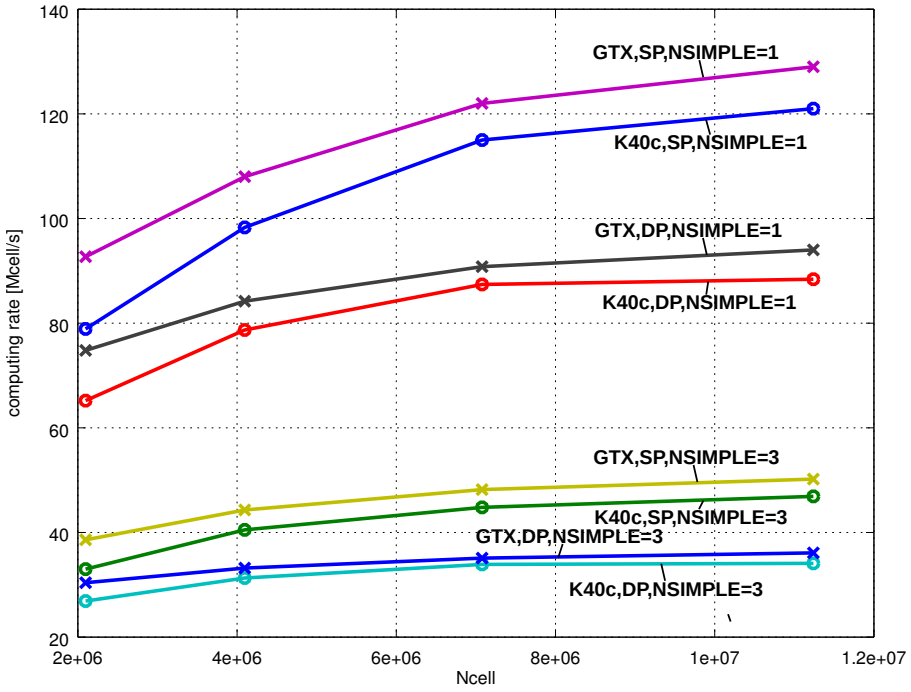


Figura 3.12: Tiempos de computo obtenidos en una NVIDIA Tesla K40c y una GTX 580 Titan Black.

La eficiencia del método FVM/IBM2 fue medida tanto en una NVIDIA Tesla K40c como en una GTX 580 Titan Black (CUDA version 6.5, Driver Version: 340.29). La tarjeta K40c dispone de 2880 procesadores corriendo a @876 [MHz] y unos 12 [GB] de memoria RAM (ECC), mientras que la Titan Black dispone de 2880 procesadores a @980 [MHz] con un total de 6 [GB] de RAM (no ECC). La Figura (3.12) muestra los tiempos de cálculo en [Mcell/s], esto es el número de celdas que se pueden procesar por segundo para un paso de tiempo completo. Este número resulta inversamente proporcional al números de iteraciones tanto

internas (N_{IBM}) como externas (N_{simple}). Para las pruebas que se presentaron previamente, y que se analizarán en esta sección, se utilizó $N_{IBM} = 1$, $N_{simple} = 3$ y CFL= 0.1. Los tiempos de cálculo reportados se corresponden con mallas de 128, 160, 192 y 224 celdas por dirección, manteniendo fija la cantidad de iteraciones externas, $N_{simple} = 1$, y considerando dos casos para las iteraciones internas, $N_{iBM} = 1$ y $N_{iBM} = 3$, en simple y doble precisión (SP/DP). Los tiempos de cálculo crecen monótonamente con el paso de malla, alcanzando los 130 [Mcell/s] para el caso de la GTX 580 Titan Black y una malla de $224^3 = 11.2$ $[Mcell]$ con $N_{simple} = 1$ en SP y 94 [Mcell/s] en DP. La GTX 580 Titan Black presenta un rendimiento mayor a la Tesla K40c en ambos casos, SP y DP. Si se considera $N_{simple} = 3$ luego el rendimiento cae a razón de 3 : 1, es decir a 50.2 $[Mcell/s]$ en SP y 36.1 [Mcell/s] en DP.

Para poner estos resultados en contexto, se puede comparar respecto a la velocidad teórica máxima debido al ancho de banda de la memoria calculada como $r_b = b/(mp)$, donde b hace referencia a la banda efectiva entre la GPGPU y la memoria del dispositivo, m es la cantidad de escalares accedidos por celda en una iteración y p es la precisión en bytes. Considerando que en el presente código $m = 66$ escalares son accedidos y que el ancho de banda efectivo resulta de $b = 234$ [GB/s] para la Tesla K40c y de $b = 228$ [GB/s] par la GTX 580 Titan Black, luego la velocidad máxima se encuentra limitada por $r_b = 886$ [Mcell/s] y $r_b = 863$ [Mcell/s] respectivamente, en SP ($p = 4$). Es así como, considerando que sólo se puede obtener un 15 % o menos del límite teórico, se puede concluir que se dispone de una implementación limitada por la velocidad a la que se están accediendo los datos en la memoria.

Por otro lado el profiler de NVIDIA reporta que la cantidad de operaciones de punto flotante por celda es de aproximadamente 1320 pudiendo inferir de esta forma una velocidad de cálculo de 152 [Gflop/s], nuevamente bien por debajo de los límites teóricos de la placa (2.56 [Tflop/s], sin considerar operaciones Mult-Add [Nvi10]). No obstante es un valor mayor a los resultados encontrados en trabajos similares.

El análisis que se presentará a continuación es similar al realizado en [LCWW14], donde un esquema de diferencias finitas en mallas Cartesianas estructuradas con paso de malla constante es analizado. En dicho trabajo se presenta una formulación con compresibilidad artificial y primer orden en el espacio aunque con la capacidad de realizar refinamiento estructurado. El pico de performance reportado es de 37 [Mcell/s] en una Tesla C2070 (1.3 [Tflop/s] SP). Basado en un número estimado de flop por celda se estima que la velocidad de cálculo es de 17.4 [Gflop/s], un orden menor a los resultados del esquema presentado a lo largo de esta tesis (pero con una placa con menor capacidad de cómputo).

En [CCLW11] los autores reportaron una velocidad de computo de 50 [Mcell/s] (en SP y utilizando elementos tetrahédricos) en una Tesla C1060 (1 [Tflop/s] SP) utilizando una formulación explícita compresible de volúmenes finitos en mallas no estructuradas. Aunque dicha formulación no pueda ser directamente comparable a la presentada en esta tesis, las siguientes conclusiones pueden extraerse. Como indican los autores, los costos de operar con tetraédros escalan en relacion $1:5$ respecto a mallas hexaédricas para la misma precisión. La presente formulación resuelve un problema FSI, resolviendo implícitamente la presión, mientras que su trabajo utiliza formulaciones compresibles, completamente explícitas, siendo bien conocido que esto resulta particularmente eficiente para las GPGPU. Por último, la GTX 580 Titan Black es una placa más actual y por lo tanto su poder de cálculo es 4 a 5 veces superior respecto a la utilizada en dicho trabajo.

Capítulo 4

Ejemplo de uso: acople térmico

4.1. Introducción

Como ya ha sido adelantado previamente, el acople térmico será analizado en esta sección. Más aún, todo el análisis a ser presentado servirá como punto de partida para una serie de estudios que merecen ser llevados a cabo como futuros trabajos. En particular, existen algunos casos de estudio sobre los cuales se está trabajando ahora mismo y esta sección intenta dirigir al equipo de trabajo en la dirección correcta.

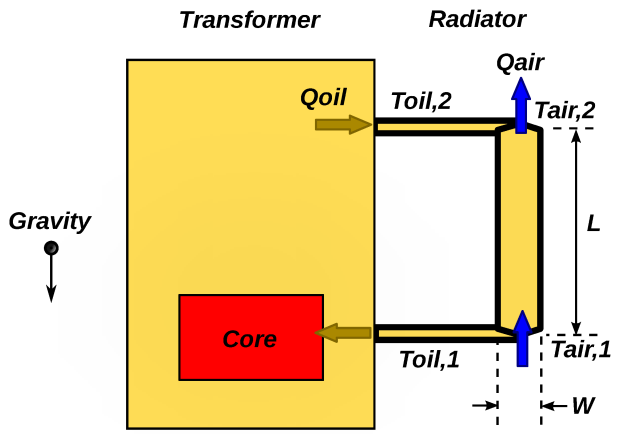


Figura 4.1: Dibujo esquemático del problema estudiado.

En este ejemplo se simulará el proceso de disipación térmica en canales de longitud L, presentes en un radiador, una configuración típica usada en transformadores de potencia eléctrica. El aire, con densidad $\rho = 1.177\,[\mathrm{kg/m^3}]$, calor específico $c_p = 1007\,[\mathrm{J/K}]$, viscosidad cinemática $\nu = 1.5777 \times 10^{-5}\,[\mathrm{m^2/s}]$, número de Prandtl $\mathrm{Pr} = 0.713$, conductividad térmica $\kappa = 2.623 \times 10^{-2}\,[\mathrm{W/mK}]$ y coeficiente de expansión térmica

$\beta = 3.43 \times 10^{-3} \left[\mathrm{K}^{-1}\right]$, entrará en un regimen de convección natural, asumiendo una constante de gravitatoria $g = 9.81 \left[\mathrm{m/s}^2\right]$, por el hecho de ser sometido a un salto de temperatura $\Delta T = T_c - T_a$, donde T_c es la temperatura (constante) de las paredes del canal y T_a es la temperatura del flujo entrante de aire. En estos tipos de problemas, el calor es aportado por el núcleo del transformador y será refrigerado por la acción de un aceite que pasa a través del radiador, como se muestra en la Figura (4.1). Vale la pena mencionar que, aunque la geometría real del transformador pudiera diferir respecto al que aquí se analiza, los resultados obtenidos serán igualmente válidos.

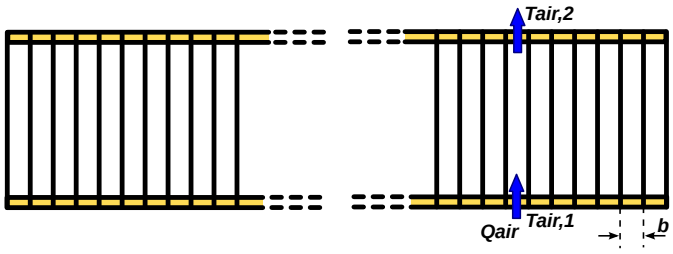


Figura 4.2: El radiador se compone por un arreglo de canales. El proceso de convección natural ocurrirá entre canales.

Así, el aceite caliente va a circular por el radiador, que se compone de un arreglo de canales, y será aquí donde el proceso de refrigeración tomará lugar. Debido a una diferencia de temperatura entre las paredes de los canales y la temperatura ambiente, un flujo se desarrollará entre los canales como consecuencia de la convección natural. La situación se muestra en la Figura (4.2), donde el canal ha sido rotado 90° sólo a efectos de visualización. Como resultado, será interesante estudiar como mejorar este proceso de transferencia considerando que las propiedades termofísicas permanecen fijas.

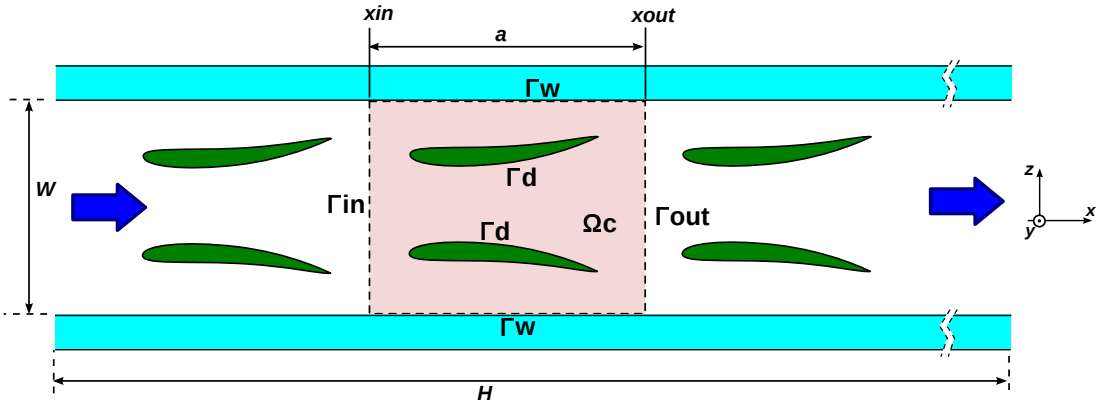


Figura 4.3: Croquis de generadores de vórtices dentro de un canal. Una celda (y su versión espejada) se muestra en color rojo claro.

Con esto en mente, un concepto conocido como *Generadores de Vórtices (VG)* va a ser

estudiado. La idea por detrás de estos VG es la de modificar, pasivamente, el patrón de flujo con el objetivo de mejorar el proceso de transferencia térmica perturbando la geometría original. Un ejemplo de VG se muestra en la Figura (4.3), donde un arreglo de perfiles aerodinámicos puede ser visto. Será entonces de vital importancia escoger el mejor VG, la distancia más conveniente entre ellos, la correcta separación entre el objeto y la pared, entre muchos otros factores a optimizar.

La geometría computacional a estudiar se muestra en la Figura (4.4), de aquí en más una *celda*. Aunque al final un arreglo bidimensional de VG será utilizado, una configuración con una sóla celda será estudiada a efectos de reducir la complejidad (computacional) del problema. Más aún, la forma de ala que se mostró previamente ha sido reemplazada por una forma del tipo delta [GJ97]. Como resultado, la nueva geometría consistirá en un VG con forma de delta con longitud de cuerda $L_c = 12.7 \times 10^{-3}\,[\mathrm{m}]$ dentro de una caja rectangular de dimensiones $8L_c \times 4L_c \times 4L_c$. El centroide de la delta estará localizado en $(1.5L_c, 2L_c, L_c/4)$. El calor aportado al aire por el aceite será modelado como una pared caliente. Note también que la geometría mostrada en la Figura (4.4) es diferente a aquella mostrada en la Figura (4.3), pues sólo la mitad de la celda definida previamente será utilizada. De esta forma, se podrá comparar el calor transferido por la pared con el VG respecto a aquella que no presenta VG.

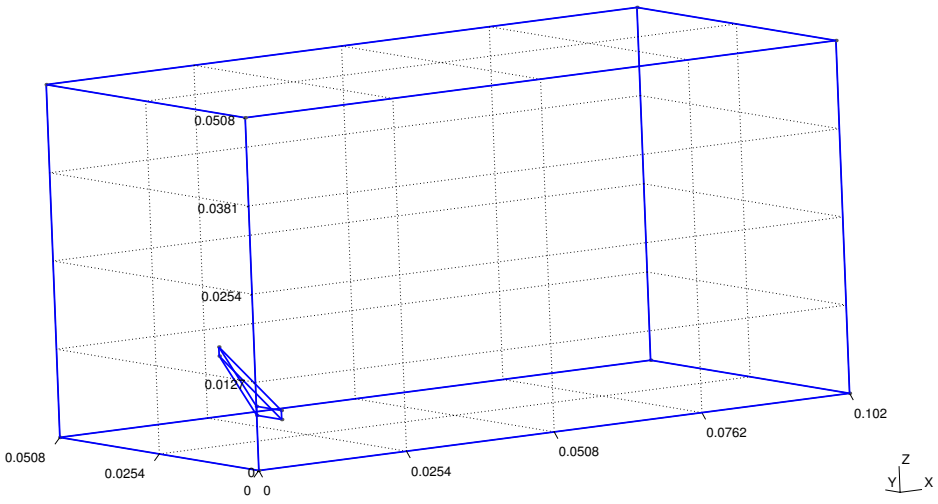


Figura 4.4: Delta dentro de un dominio rectangular.

Como mayor reto, debido a que esta prueba consiste en un arreglo de deltas, los costos computacionales crecerán proporcionalmente al número de celdas simuladas. Con eso en mente, se analizará y comparará dos configuraciones diferentes. En la primer

configuración se resolverá un arreglo de celdas haciendo uso del software OpenFOAM, aprovechando su capacidad para resolver en paralelo problemas de gran tamaño. En la segunda configuración se resolverá sólo una celda pero esta vez usando una condición de contorno en la temperatura dependiente del tiempo haciendo uso del método propuesto en CUDA. Como resultado, su solución (y metodología) será validada comparándola con la solución obtenida por un código bien conocido y luego se utilizará esta formulación para resolver un problema de optimización.

4.2. El factor λ

Con el fin de comparar las soluciones obtenidas en las dos diferentes configuraciones (OpenFOAM y CUDA), un factor denotado como λ va a ser calculado. Este factor será esencialmente la tasa de decaimiento exponencial del campo de temperaturas en la dirección del flujo. Puede ser también visto como un factor de calidad, similar a otros parámetros introducidos en trabajos similares [GJ97]. A tal fin se considerará que el flujo es periódico por celda aunque por supuesto esto no será el caso debido principalmente a las diferencias por efectos de flotación dentro de cada celda, pero en aras de mantener el cálculo simple será el punto de partida.

Considerando la solución a la ecuación de calor sujeta a condiciones de contorno Dirichlet en $x = 0$ y Neumann en $x = L$, luego de una cierta longitud de entrada térmica será válido que:

$$T(\mathbf{x} + \hat{\mathbf{i}}L; t) = \lambda T(\mathbf{x}; t) \tag{4.1}$$

donde L es la longitud de la celda en la dirección x y el factor λ puede ser calculado por un ajuste de mínimos cuadrados como:

$$\lambda_{j,j+1} = \frac{\int_{\Gamma_{in}} \theta(\mathbf{x} + \hat{\mathbf{i}}L; t)\theta(\mathbf{x}; t)\ d\Gamma}{\int_{\Gamma_{in}} |\theta(\mathbf{x}; t)|^2\ d\Gamma} \tag{4.2}$$

donde $\theta = T - T_h$ y T_h hace referencia a la temperatura de la pared. Este método será utilizado para el caso del arreglo de celdas (OpenFOAM).

El factor puede ser también calculado considerando la descomposición de la matriz del sistema. Para ello se introducirá notación que será utilizada a continuación. Considerando la Figura (4.5), se denotará como I_j a los DOF de la interface Γ_j, esto es entre las celdas Ω_{j-1} y Ω_j, y por D_j a los DOF en el interior de Ω_j. La descomposición matricial diagonal por bloques puede ser escrita, considerando que muchas de las submatrices son idénticas

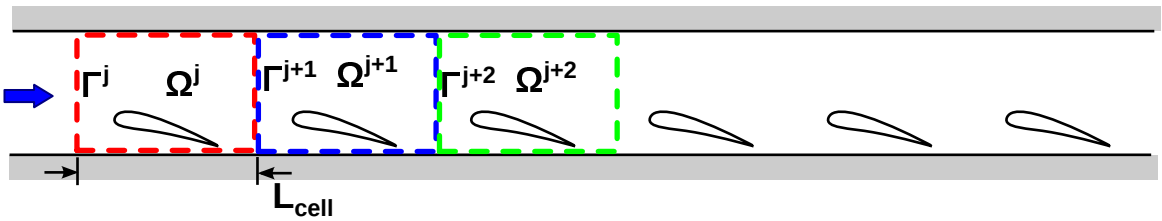


Figura 4.5: Arreglo de celdas.

debido a la hipótesis de la periodicidad, como:

$$\begin{aligned}
\mathbf{C_1 x_1^I} + \mathbf{D_1 x_1^D} &= \mathbf{b_1^I} \\
\mathbf{E x_1^I} + \mathbf{F x_1^D} + \mathbf{G x_2^I} &= \mathbf{b_1^D} \\
\mathbf{B x_1^D} + \mathbf{C x_2^I} + \mathbf{D x_2^D} &= \mathbf{b_2^I} \\
&\vdots \\
\mathbf{E x_N^I} + \mathbf{F x_N^D} + \mathbf{G x_{N+1}^I} &= \mathbf{b_N^D} \\
\mathbf{B x_N^D} + \mathbf{C_{N+1} x_{N+1}^I} &= \mathbf{b_{N+1}^I}
\end{aligned} \tag{4.3}$$

donde la primer y la última ecuación puede diferir de las restantes debido a las condiciones de contorno del problema.

Notar que una ecuación típica para los nodos interiores, considerando que no existen fuentes de calor, es:

$$\mathbf{E x_j^I} + \mathbf{F x_j^D} + \mathbf{G x_{j+1}^I} = \mathbf{0} \tag{4.4}$$

por lo cual $\mathbf{x_j^D}$ puede ser estáticamente condensada (reducida) como:

$$\mathbf{x_j^D} = -\mathbf{F^{-1}}(\mathbf{E x_j^I} + \mathbf{G x_{j+1}^I}) \tag{4.5}$$

La introducción de esta última expresión dentro de una típica ecuación de interfaz permite derivar la siguiente ecuación:

$$\mathbf{H^0 x_{j-1}^I} + \mathbf{H^1 x_j^I} + \mathbf{H^2 x_{j+1}^I} = 0 \tag{4.6}$$

donde:

$$\begin{aligned}
\mathbf{H^0} &= -\mathbf{B F^{-1} E} \\
\mathbf{H^1} &= \mathbf{B F^{-1} G} - \mathbf{C} + \mathbf{B F^{-1} E} \\
\mathbf{H^2} &= -\mathbf{D F^{-1} G}
\end{aligned} \tag{4.7}$$

Considerando ahora que la Ecuación (4.1) es válida, siguiendo la notación previamente introducida se obtiene:

$$\mathbf{x_j^I} = \lambda \mathbf{x_{j-1}^I} \tag{4.8}$$

que en esencia es una ecuación cuadrática de valores propios:

$$(\mathbf{H^0} + \lambda \mathbf{H^1} + \lambda^2 \mathbf{H^2}) \mathbf{x}^{\mathbf{I}}_{\mathbf{j-1}} = \mathbf{0} \tag{4.9}$$

Más aún, la Ecuación (4.9) puede ser transformada de cuadrática a lineal definiendo una nueva variable, $\mathbf{y_j}$, cuya definición resulta:

$$\mathbf{y_j} = \begin{bmatrix} \mathbf{x_j} \\ \mathbf{x_{j+1}} \end{bmatrix} \tag{4.10}$$

y ahora la Ecuación (4.9) puede reescribirse como:

$$\begin{bmatrix} \mathbf{H^1} & \mathbf{H^2} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{y_j} = \begin{bmatrix} -\mathbf{H^0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{y_{j-1}} \tag{4.11}$$

que es una ecuación de valores propios de la forma:

$$\mathbf{y_j} = \mathbf{Q} \mathbf{y_{j-1}} = \lambda \mathbf{y_{j-1}} \tag{4.12}$$

donde:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{H^1} & \mathbf{H^2} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} -\mathbf{H^0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{4.13}$$

Una vez que la matriz del sistema se ensambla el cálculo del factor λ puede ser llevado a cabo resolviendo la Ecuación (4.12). Como resultado, el máximo valor propio λ_{max}, con $|\lambda_{max}| < 1$, será obtenido. Considerando que es extremedamente común encontrarse con miles de celdas sobre una interfaz, este cálculo puede volverse complejo y por lo tanto a continuación se presentará un cálculo de menor costo computacional.

Finalmente el factor λ puede ser obtenido utilizando el *Método de la Potencia Iterada (PIM)* [BF01], el cual básicamente establece que luego de aplicar sucesivas veces el operador de *mapeo* a un vector, los componentes de alta frecuencia serán filtrados y sólo restará un componente activo. Matemáticamente hablando, consistirá en una serie de etapas en las cuales dos cantidades serán calculadas. Primero, se calculará la Ecuación (4.2) y luego se realizará una renormalización (necesaria por PIM), esto es:

$$\theta^{k+1}(\mathbf{0}; t) = \frac{\theta^k(\hat{\mathbf{i}}L; t)}{\|\theta^k(\hat{\mathbf{i}}L; t)\|} \tag{4.14}$$

donde debe cumplirse que:

$$\|\theta^k(\hat{\mathbf{i}}L; t)\| \to \lambda \tag{4.15}$$

Este método será utilizado en la solución con una sola celda (CUDA).

4.3. Solución con arreglo de celdas

Con el objetivo de resolver tantas celdas como sea posible, resulta obligatorio reducir al mínimo la cantidad de celdas de discretización sin hacer caso omiso a la calidad de la malla final. A tal fin, SnappyHexMesh fue utilizado, obteniendo una malla de alrededor de 2 millones de celdas computacionales por celda. En este punto vale la pena mencionar que como el número de Rayleigh de este problema es pequeño, $\mathrm{Ra} = 1.0 \times 10^5$, no fue necesario incluir gran cantidad de superficies de refinamiento (layers). En particular se utilizaron 2 para la delta y 3 para las paredes, con un radio de crecimiento $1 : 2$ y refinamiento por distancia donde sea necesario. Como ejemplo, la Figura (4.6) muestra la malla utilizada como referencia.

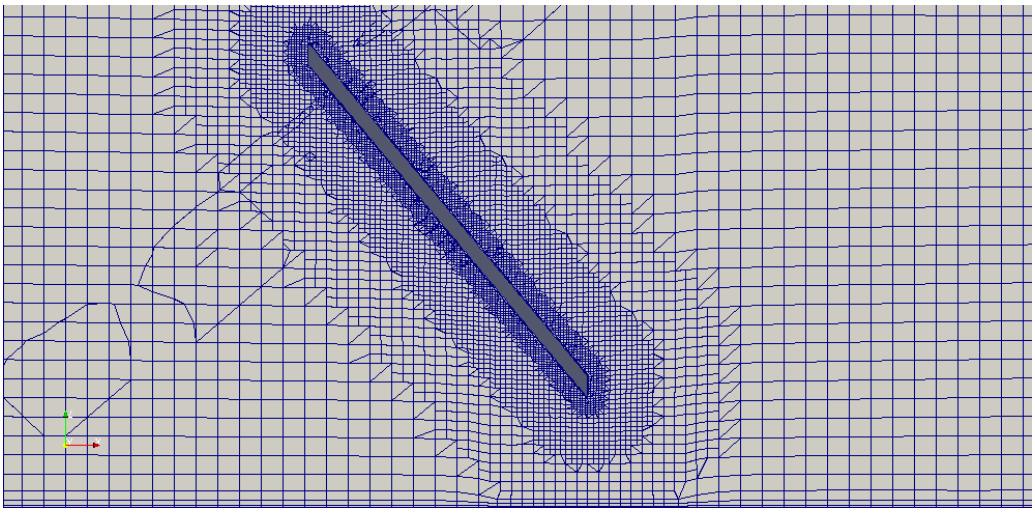


Figura 4.6: Un corte de la malla utilizada como referencia.

Esta malla de referencia ha servido como base para la configuración de arreglo de celdas. Una combinación de *translatePoints*, *mergeMeshes* y *stitchMesh* (todas herramientas provistas por OpenFOAM) han sido utilizadas para trasladar, unir y ajustar las condiciones de contorno de la malla final que, considerando 32 celdas, estuvo compuesta por cerca de 55 millones de celdas computacionales.

El problema fue corrido en el Cluster HPC Seshat del CIMEC que se encuentra compuesto por 69 nodos de cálculo $E5 - 1620$ v2 @3.70 [GHz] (4 cores) con 292 núcleos disponibles para cálculo. En particular, tomó aproximadamente 11400 horas núcleo calcular los resultados que serán expuestos a continuación.

Se utilizó *buoyantBoussinesqSimpleFoam* y las condiciones de contorno utilizadas se muestran en la Tabla (4.1). Es importante remarcar que, como la presión no es la fuerza impulsora principal en este tipo de problemas, fue necesario ajustarla para que sea la

flotación el motor principal del movimiento. Esto puede ser visto en la condición de contorno de entrada sobre *p_rhg*, con un valor de $gL_g \left[\mathrm{m}^2/\mathrm{s}^2\right]$ donde $L_g = n_{ca}L$ y n_{ca} es la cantidad total de celdas en el arreglo de celdas, en este caso 32.

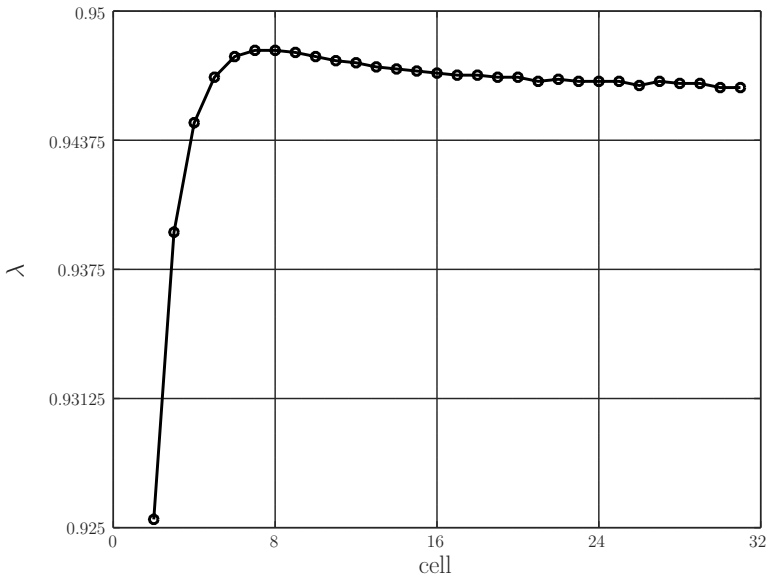
	U	p_rgh	T
inlet	type pressureInletOutletVelocity value uniform(0,0,0)	type prghTotalPressure rho rhok p0 uniform 31.894 value uniform 0	type fixedValue value uniform 303.15
outlet	type pressureInletOutletVelocity value uniform(0,0,0)	type prghTotalPressure rho rhok p0 uniform 0 value uniform 0	type zeroGradient
sides	type symmetry	type symmetry	type symmetry
walls	fixedValue uniform(0,0,0)	type fixedFluxPressure	type fixedValue value uniform 304.15
delta	type fixedValue value uniform(0,0,0)	type fixedFluxPressure	type zeroGradient

Cuadro 4.1: Condiciones de contorno utilizadas para la simulación de arreglo de celdas (OpenFOAM).

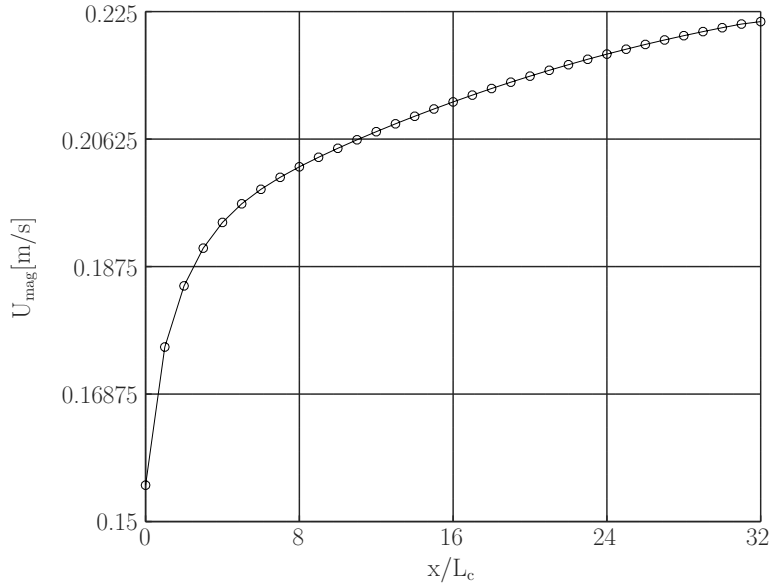
Sin entrar en tantos detalles, se utilizaron los siguiente parámetros para la simulación: factores de relajación, 0.7 para la velocidad, 0.3 para la presión y 0.5 para T, utilizando el método SIMPLE con una iteración por no ortogonalidad; tolerancia de 1×10^6 para la convergencia al estacionario en todas las variables; tolerancia absoluta (relativa) de 1×10^8 (1×10^4) para p, mientras que se utilizó una tolerancia absoluta (relativa) de 1×10^8 (1×10^3) para $\mathbf{u}$ y T; respecto a los métodos de resolución de los sistemas lineales, para p ($\mathbf{u}$ y T) se utilizó PCG (PBiCG) con DIC (DILU) como precondicionador; finalmente se utilizaron diversos esquemas para cada término en las distintas ecuaciones, por ejemplo para la advección se utilizaron esquemas limitados y linearUpwind.

La convergencia del factor λ, obtenido al aplicar la Ecuación (4.2), se muestra en la Figura (4.7), donde puede observarse una convergencia hacia un valor cercano a 0.946. Dicho valor puede ser visto como una mejora en la transferencia de calor en la celda de

alrededor del 5 %.



(a) Convergencia del factor λ.



(b) Convergencia del valor máximo de velocidad.

Figura 4.7: Resultados para la configuración de arreglo de celdas obtenidos haciendo uso de *buoyantBoussinesqSimpleFoam*.

Los resultados obtenidos por la configuración de arreglo de celdas se muestran en la Figura (4.8). En resumen:

- El campo de velocidades muestra una velocidad máxima de $0.225\,[\mathrm{m/s}]$. Como fue dicho previamente, como la presión no juega un papel importante en este problema

y la temperatura del aire alcanzará (en el límite) la temperatura de la pared, estos campos no serán mostrados aquí.

- La capa límite térmica es más fina donde se encuentra la delta, respecto a aquella en ausencia del VG, incrementando así la transferencia térmica. Más aún, es esperado que la transferencia térmica aumente a medida que el salto de temperaturas sea mayor, como lo será en condiciones operativas.

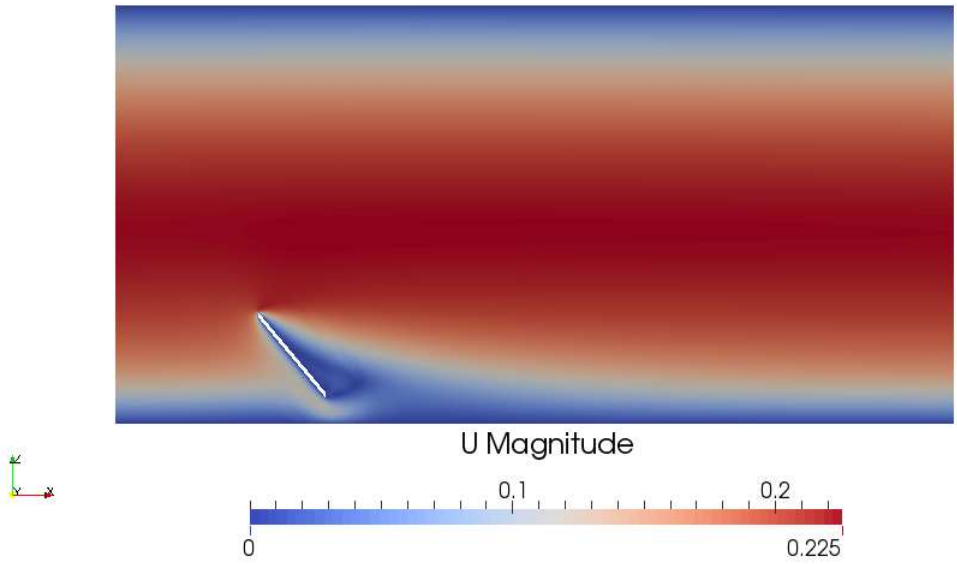
- Otra característica importante es que, con estas propiedades termofísicas y con este salto de temperatura, no se puede generar un vórtice de magnitud importante detrás de la delta. Mientras más energía se introduzca al problema (incrementando el salto de temperatura, por ejemplo) un vórtice sostenido se generará en la parte posterior de la delta, que absorberá calor de la capa límite térmica y la mezclará con el flujo de la corriente media (a menor temperatura) incrementando aún más la transferencia de calor.

4.4. Solución con una celda

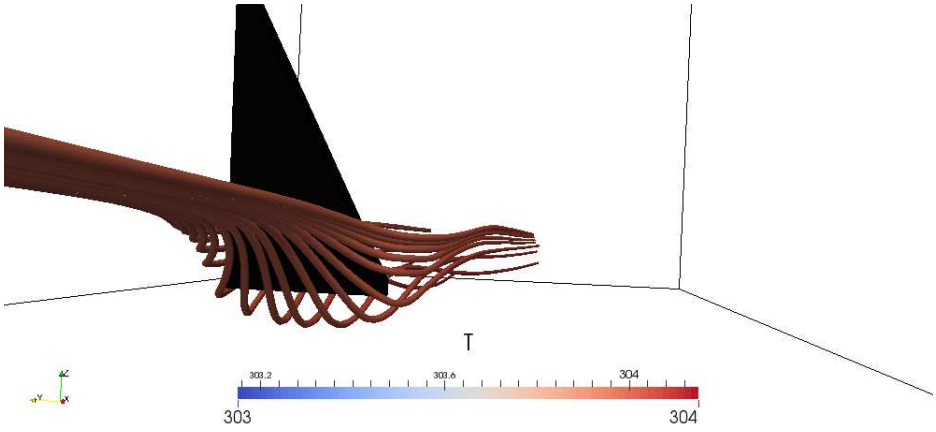
A efectos de computar la solución con una celda haciendo uso del método propuesto en CUDA, el PIM será modificado como se detalla a continuación. El flujo convergerá al mismo tiempo que lo hace el cálculo del factor λ y este último será recalculado en cada iteración externa luego de resolver la ecuación de transporte para la temperatura. Existen muchas formas de acoplar el cálculo del factor λ con el cálculo del flujo, pero esta configuración en particular dió buenos resultados.

Vale la pena mencionar que sería interesante estudiar con mayor grado de detalle si esta forma de calcular el factor λ, es decir conjuntamente conforme se resuelve el flujo, es la forma correcta (en términos de velocidad de cálculo) para resolver este problema. Otra posibilidad podría ser la de mantener fijo el factor λ mientras se calcula el estado estacionario del flujo y una vez convergido el flujo recalcular el factor λ y volver a repetir la búsqueda del estacionario del flujo hasta satisfacer alguna métrica (no sólo relacionada con el flujo sino también con el factor λ).

Considerando la misma división de fronteras que la solución de arreglo de celdas, esto es inlet/outlet (dirección x), sides (dirección y), walls (dirección z) y delta, las condiciones de contorno ahora aplicadas se presentan en la Tabla (4.2). Notar que la



(a) Campo de velocidad.



(b) Acercamiento a la delta y sus líneas de corriente.

Figura 4.8: Resultados para la configuración de arreglo de celdas, en particular de la última celda, obtenidos haciendo uso de *buoyantBoussinesqSimpleFoam*.

condición de contorno de la temperatura en el inlet cambia en función del tiempo (más precisamente, en función del factor λ) y este cambio será obtenido calculando la Ecuación (4.14). Notar también que no se requieren de condiciones de contorno explícitas para la presión [FVOMY00] (lo mismo ocurría en el caso de estudio de la Sección (3)). Además, se utilizaron parámetros de convergencia un poco más relajados en comparación con aquellos utilizados en la configuración de una celda (OpenFOAM).

Los resultados, obtenidos por una NVIDIA GTX 580 Tital Black en 22 horas de GPGPU, se muestran en la Figura (4.9), donde una malla de $256 \times 128 \times 128$ celdas computacionales ha sido utilizada. Más aún, para $\mathbf{u}$ (T) se utilizaron 3 (2) iteraciones externas, alcanzando así tolerancias relativas de $1.0 \times 10^{-1} - 1.0 \times 10^{-2}$, para ambos casos, y se fijaron en 1 (1)

	U	T
inlet	cyclic	Ecuación (4.14)
outlet	cyclic	$\nabla T \cdot \hat{i} = 0$
sides	symmetric	symmetric
walls	no-slip	T_h
delta	no-slip	$\nabla T \cdot \hat{n} = 0$

Cuadro 4.2: Condiciones de contorno en la configuración de una celda (CUDA).

las iteraciones internas.

Notar que la delta, y la superficie de la caja rectangular, se muestran con contorno negro, puesto que son mostradas al aplicar un filtro (de contorno) de Paraview. De esta forma, en caso que existiesen inconsistencias geométricas respecto a la geometría original esto puede ser explicado bien por una incompleta resolución de la malla (cerca de la punta de la delta, por ejemplo) o bien por limitaciones en el filtro de contorno de Paraview.

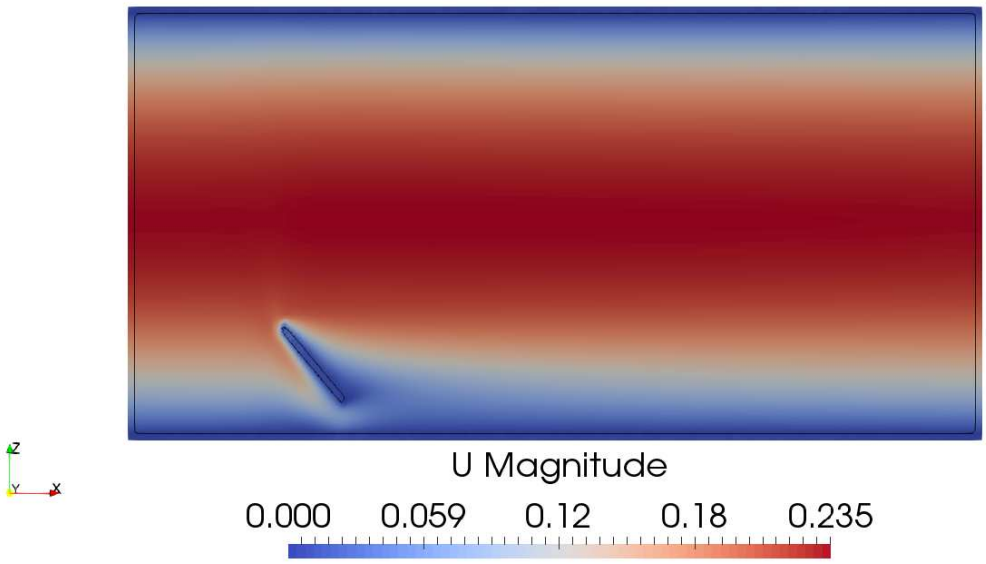


Figura 4.9: Resultados de la configuración de una celda (CUDA).

Finalmente, el factor λ obtenido por el presente cálculo converge al valor de 0.958, con una diferencia relativa próxima al 1 % respecto a la solución con el arreglo de celdas.

4.4.1. Tiempos de cálculo

Antes de avanzar con una tentativa de comparación de tiempos de cálculo entre la versión de una celda (CUDA) y la versión con un arreglo de celdas (OpenFOAM), se introducirá la fórmula a utilizar para evaluar el rendimiento de cada una. En particular, si se considera a N_{cell} como la cantidad total de celdas computacionales, a N_{step} como la cantidad de pasos de tiempo (iteraciones) requeridos para alcanzar el estado el estacionario, N_{up} como la cantidad de unidades de procesamiento y a t como el tiempo total de cómputo, luego el rendimiento por paso de tiempo puede ser calculado como:

$$r = \frac{N_{step} N_{cell}}{N_{up} t} \tag{4.16}$$

No obstante, se considera que el tiempo de cálculo en OpenFOAM podría ser optimizado configurando mejor los parámetros de relajación, los parámetros de convergencia tanto del solver como de cada variable dependiente, los métodos de resolución de los sistemas lineales, los esquemas utilizados para cada término en las diversas ecuaciones, entre otros. Vale la pena aclarar en este punto que las optimizaciones propuestas no necesariamente producirán un incremento en el rendimiento sino que tal vez podrían reducir el tiempo de cálculo compensando a una eventual reducción de cantidad de iteraciones.

Además se considera que la utilización (o no) del factor lambda y la metodología de cálculo influye poco (o nada) en el rendimiento, puesto que no va a influenciar el tiempo de cálculo por iteración sino en la cantidad de iteraciones para lograr el estado estacionario compensando con una reducción (o aumento, en el peor caso) del tiempo de cálculo total.

En lo que respecta a los resultados de OpenFOAM se tiene que $N_{step} = 210835$, $N_{cells} = 5.5 \times 10^7$, $N_{up} = 20$ (nodos, con 4 procesadores cada uno) y $t = 513000\,[\text{s}]$. Introduciendo estos valores en la Ecuación (4.16) se obtiene un rendimiento de $r_{OF} = 1.1302\,[\text{Mcell/s}]$ (o de $22.6\,[\text{Mcell/s}]$ si se considera $N_{up} = 1$). Mientras que para los resultados de CUDA se tiene que $N_{step} = 2.64 \times 10^5$, $N_{cells} = 4 \times 10^6$ y $t = 79000\,[\text{s}]$. De esta forma se obtiene un rendimiento de $r_{CU} = 14\,[\text{Mcell/s}]$ (DP).

Como puede notarse, la performance de ambas configuraciones difiere en prácticamente un orden. No obstante OpenFOAM hace uso de 80 núcleos (con el costo monetario asociado que eso conlleva) mientras que el mismo rendimiento es alcanzado por una GPGPU relativamente antigua (comprada en el 2015). Note además que los resultados de la configuración con una celda difieren de los presentados en la Figura (3.12) (DP), donde para un problema con dimensiones similares se tenía un rendimiento de aproximadamente

33 [Mcell/s]. Esto se debe principalmente al hecho que el problema que se resuelve en esta sección presenta mayor cantidad de cálculos que el de la sección previa. Finalmente el rendimiento a SP presenta una mejora respecto a los de DP de aproximadamente un 50 %.

De todas formas, la comparación resulta un tanto injusta. Primero que todo OpenFOAM es un software paralelo en CPU de propósito general (con varias décadas de desarrollo y miles de personas trabajando en él). Resuelve todo tipo de flujos y en particular utiliza mallas no estructuradas (aunque en este caso de estudio se hizo uso de mallas estructuradas reduciendo así las no ortogonalidades), mientras que el método propuesto utiliza grillas cartesianas de paso homogéneo por dirección (con lo cual se pueden realizar una gran cantidad de simplificaciones). Además, permite resolver problemas de gran tamaño mientras que el método propuesto se encuentra altamente limitado por la memoria (no se dispone aún de una implementación MPI en memoria distribuida). Por lo tanto es importante hacer énfasis en la utilización mutua y no alternativa.

Capítulo 5

Conclusiones

Los resultados obtenidos en un trabajo previo, donde una combinación de Transformadas Rápidas de Fourier y un algoritmo de Gradientes Conjugados aplicado a la solución de las ecuaciones de Navier-Stokes en CUDA fue expuesto y profundamente estudiado, ha definido los fundamentos, y también la dirección, de la presente tesis doctoral.

Primero que todo, la implementación de un resolverdor de las ecuaciones de Navier-Stokes en CUDA fue desarrollado. Algunos casos claves de academia fueron satisfactoriamente resueltos utilizando dicho método y, tan rápido como los primeros resultaron fueron obtenidos, resultó claro que eran requeridas una serie de mejoras para incrementar la calidad y extender la aplicabilidad del método propuesto.

La primera mejora consitió en el uso de métodos lagrangianos en combinación con el método BFECC, para la etapa de advección, y poder así incrementar el paso de tiempo y reducir finalmente los tiempos de cálculo. Aunque este método satisfactoriamente incrementó la velocidad de cálculo, su implementación en la arquitectura CUDA demostró que no era un método apropiado para la GPGPU.

Luego una mejora en la representación de cuerpos embebidos fue necesaria. Como resultado, técnicas de fronteras embebidas fueron satisfactoriamente implementadas y validadas respecto a resultados experimentales en un problema de interacción fluido-estructura. El método BFECC, que había sido utilizado anteriormente, fue removido de la presente formulación puesto que tendía a degradar el rendimiento.

Finalmente la aproximación de Boussinesq fue introducida y se comenzó a estudiar un problema industrial que consistía en la mejora de la transferencia de calor dentro de un transformador industrial modificando el patrón de flujo haciendo uso de dispositivos pasivos. Aunque los resultados obtenidos no están completos, han servido y se utilizarán

por otros investigadores y estudiantes doctorales en trabajos futuros.

A modo de resumen más técnico: una implementación escrita enteramente en CUDA capaz de resolver problemas de interacción fluido-estructura, o con acople térmico, basado en el método de los volúmenes finitos en mallas Cartesianas estructuradas con fronteras embebidas, fue presentado. El objetivo en cada etapa fue el de utilizar los algoritmos que mejor se adapten a la arquitectura. Considerando sus aplicaciones, un particular énfasis ha sido puesto en la evaluación y la descripción de las fuerzas de fluido en un cuerpo rígido y en el correcto cálculo de ciertos parámetros térmicos, comparando siempre los resultados obtenidos con otros experimentales o numéricos. Para ello, un esquema de segundo orden en el espacio fue desarrollado basado en esquemas colocados estabilizados con Rhie-Chow y el algoritmo SIMPLE fue modificado para incluir el operador que garantiza las condiciones tanto de contorno como sobre la superficie de los sólidos embebidos.

Considerando los dos casos estudiados, las siguientes conclusiones pueden ser obtenidas:

- Un experimento capaz de detectar los efectos de la fuerza de fricción y masa agregada fue diseñado. Básicamente consiste en una esfera (boya) con flotación positiva sujeta al fondo de un tanque con agua por una cuerda. La mesa esta sujeta a un movimiento controlado resultando en un sistema que se asemeja a un péndulo invertido. Un algoritmo de captura de movimiento fue implementado para extraer la posición de la boya durante el movimiento. Las capacidades de la presente formulación fueron validadas en comparación con otra estrategia FEM/ALE y una estrategia previa de primer orden reportada en la literatura. Los resultados del esquema de segundo orden exhiben un comportamiento excelente. En particular, las simulaciones de la boya sumergida satisfactoriamente ajustaron los datos experimentales confirmando que las fuerzas de fricción y masa agregada fueron realmente bien capturados.

- La estrategia del factor lambda puede ser utilizada como un indicador objetivo para validar el ajuste térmico realizado sobre el esquema de segundo orden presentado. Los resultados GPGPU obtenidos, haciendo uso de sólo una celda, coinciden con aquellos resultados obtenidos con OpenFOAM haciendo uso de un arreglo de celdas. Esto es interesante dado principalmente por dos aspectos: primero, toda la física involucrada en el problema real se reduce a sólo una celda; y segundo, reduce así los costos computacionales. Como resultado, puede ser incorporada a un algoritmo de optimización que busque la combinación que maximice la transferencia de calor

entre las paredes y el fluido. Esto será el principal objetivo de este caso de estudio, que será aplicado luego a problemas industriales.

Finalmente merece ser reiterado que la técnica de fronteras embebidas utilizada en la presente formulación ha dado resultados más que interesantes. Debe considerarse que en todo momento tanto los efectos de cuerpos sólidos como de todas las condiciones de contorno fueron impuestas haciendo uso del mismo método. Es simple, económico y efectivo. Estas técnicas podrían ser utilizadas en *conjunto* con otros códigos en memoria distribuída, como OpenFOAM por ejemplo. La implementación actual puede brindarles soluciones computacionalmente económicas que pueden ser luego utilizadas en simulaciones más detalladas. Como ejemplo, la simulación realizada en CUDA puede introducirse dentro de un optimizador y, una vez que la solución sea encontrada, la simulación más fina podría ser llevada a cabo por otro código más general.

Capítulo 6

Trabajos futuros

Como fue dicho anteriormente, existe una gran cantidad de mejoras o de temas interesantes que quedan aún por realizarce. En particular todos los resultados obtenidos en la Sección (4), relacionados con los VG con forma de delta, que serán potencialmente aplicados a la industria, merecen ser atacados desde múltiples puntos de vista. Una lista no exhaustiva de trabajos futuros será presentada aquí. Vale la pena aclarar que algunos de ellos fueron parcialmente cubiertos pero sus resultados no fueron agregados al presente trabajo dado a su escaso grado de avance:

- Multi-GPGPU: El código GPGPU desarrollado en este trabajo ha sido concebido, desde principios, como muchos otros algoritmos del tipo CA, es decir, como los métodos del tipo *Lattice Boltzmann* [CD98] o los bien conocidos *juegos de la vida (game of life)*. Esto es, las ecuaciones de cantidad de movimiento (o las clásicas ecuaciones de transporte) van a necesitar cambios menores para distribuir el trabajo entre diferentes GPGPUs. Sin embargo, la estrategia para resolver la ecuación de Poisson para la presión podría no ser tan directa para extenderse, o al menos no con cambios menores, y va a necesitar más codificación. Sería interesante extender la resolución de la ecuación de Poisson para la presión a múltiples GPGPUs o bien incorporar estrategias del tipo multigrillas (multigrid) [BL11].

- Extender el modelo numérico para incluir más DOF con el objetivo de tratar con una descripción completa del problema FSI: El objetivo principal entonces será el de estudiar si el agregado de nuevos DOF resultan en una mejor determinación de la amplitud y fase de la boya.

- Método de partículas para calcular las funciones de LS: este es uno de los

puntos principales a atacar con el fin de calcular flujos internos y externos en geometrías complejas. Existe una gran diversidad de excelentes trabajos relacionados al tema [OF06, EFFM02, ELF05]. El uso de partículas va a incrementar el nivel de detalle de los objetos (con fronteras embebidas, más aún cuando se encuentren en movimiento) y esto va a expandir potencialmente el uso de método propuesto.

- Estudiar con mayor grado de detalle si el cálculo del factor lamba debe ser recalculado en cada iteración, una vez que el estado estacionario es alcanzado para un factor lambda dado, o si existe otra opción mejor para acoplar dicho cálculo al método propuesto.

- Optimización por algoritmos genéticos para los generadores de vórtices: al momento un optimizador por *algoritmos genéticos* totalmente compatible con el método propuesto en CUDA ha sido implementado, de esta forma se optimizará la solución en base a un conjunto dado de parámetros. A modo de ejemplo, la primer prueba podría consistir en explorar el mejor ángulo de ataque, la dimensiones óptimas del dominio (o dicho de otro modo, la mejor separación entre dos deltas sucesivas), entre otros; más aún, considerando que la geometría se encuentra parametrizada se puede ir mucho más allá de simples optimizaciones para, por ejemplo, optimizar también la geometría del dispositivo pasivo.

Capítulo 7

Anexos

Título:

A FFT Preconditioning Technique for the Solution of Incompressible Flow on GPU's

Contribución:

Implementación GPGPU del método numérico

Firma del director de tesis:

A FFT Preconditioning Technique for the Solution of Incompressible Flow on GPU's

Mario A. Storti[1], Rodrigo R. Paz[1], Lisandro D. Dalcin[1],
Santiago D. Costarelli[1], Sergio R. Idelsohn[1,2,3]

[1]*CIMEC, INTEC - Universidad Nacional del Litoral y CONICET, Santa Fe, Argentina.*

[2]*Institució Catalana de Recerca i Estudis AvanÃğats (ICREA), Barcelona, Spain*

[3]*International Center for Numerical Methods in Engineering (CIMNE), Technical University of Catalonia (UPC), Gran Capitán s/n, 08034 Barcelona, Spain*

Abstract

Graphic Processing Units have received much attention in last years. Compute-intensive algorithms operating on multidimensional arrays that have nearest neighbor dependency and/or exploit data locality can achieve massive speedups. Simulation of problems modeled by time-dependent Partial Differential Equations by using explicit time-stepping methods on structured grids is an instance of such GPU-friendly algorithms. Solvers for transient incompressible fluid flow cannot be developed in a fully explicit manner due to the incompressibility constraint. Segregated algorithms like the fractional step method require the solution of a Poisson problem for the pressure field at each time level. This stage is usually the most time-consuming one. This work discuss a solver for the pressure problem in applications using immersed boundary techniques in order to account for moving solid bodies. This solver is based on standard Conjugate Gradients iterations and depends on the availability of a fast Poisson solver on the whole domain to define a preconditioner. We provide a theoretical and numerical evidence on the advantages of our approach versus classical techniques based on fixed point iterations such as the Iterated Orthogonal Projection method.

Keywords: Graphics Processing Units; Incompressible Navier-Stokes; Poisson equation

Contents

1 Introduction

Graphics Processing Units (GPU) are computer co-processors used in desktop computers and workstations to off-load the renderization of complex graphics from the main processor (CPU). They have evolved to complex systems containing many processing units, a large amount of on-board memory and a computing power in the order of teraflops. They are instances of massively parallel architectures and Single Instruction Multiple Data (SIMD) paradigms.

Recently, GPU's are becoming increasingly popular among scientists and engineers for High Performance Computing (HPC) applications [MCPN08, RRB+08, MCP+09, ELD08, GSMY+08, LMU+09, CCLW11, TS09, APB07, BG09, MRDI12, KWBH09]. This tendency motivated GPU manufacturers to develop General Purpose Graphics Processing Units (GPGPU) targeting the HPC market.

In the pursuit of more realistic visualization algorithms for video games and special effects, solving Partial Differential Equations (PDE) has become a necessary ingredient [ETK+08, IGLF06, CLT07, RGBVC08, WLL04]. Numerical schemes employed in these applications usually sacrifice accuracy for speed, resulting in very fast implementations when comparing to engineering codes.

The resolution of Computational Fluid Dynamics (CFD) problems on GPU's requires specialized algorithms due to the particular hardware architecture of these devices. Algorithms that fall in the category of Cellular Automata (CA) are the best fitted for GPU's. For instance, explicit Finite Volume or Finite Element methods, jointly with *immersed boundary* techniques [WWZ12] to represent solid bodies, can be used on structured cartesian meshes. In the case of incompressible flows, it is not possible to develop a purely explicit algorithm, due to the essentially non-local nature of the incompressibility condition.

Segregated algorithms solve an implicit Poisson equation for the pressure field, being this stage the most time-consuming in the solution procedure. Using fast Poisson solvers like Multigrid (MG) or Fast Fourier Transform (FFT) is tempting but treating moving solid

bodies becomes cumbersome in the case of MG or unsuitable for FFT. To surpass these difficulties, Molemaker *et.al* proposed in [MCPN08] the Iterated Orthogonal Projection (IOP) method which requires a series of projections on the complete grid (fluid and solid) to enforce the incompressibility and boundary conditions.

In this work we propose an alternative to IOP, that we call Accelerated Global Preconditioning (AGP). The solver is based on using a Preconditioned Conjugate Gradients (PCG) algorithm, so that, it is an accelerated iterative method in contrast to the stationary scheme used in IOP. In addition, AGP method iterates only on pressure, whereas IOP iterates on both pressure and velocity.

The remainder of this article is organized as follows. Section (2) describes the IOP solver and the QUICK scheme for advection terms. Also, the rate of convergence of IOP is studied. Section (3) introduces the Accelerated Global Preconditioning solver providing a theoretical evidence on the advantages of our approach versus classical techniques based on fixed point iterations such as the IOP method. The numerical performance of the method is studied in Section (5). Concluding remarks are given in Section (6).

2 The Iterated Orthogonal Projection (IOP) solver

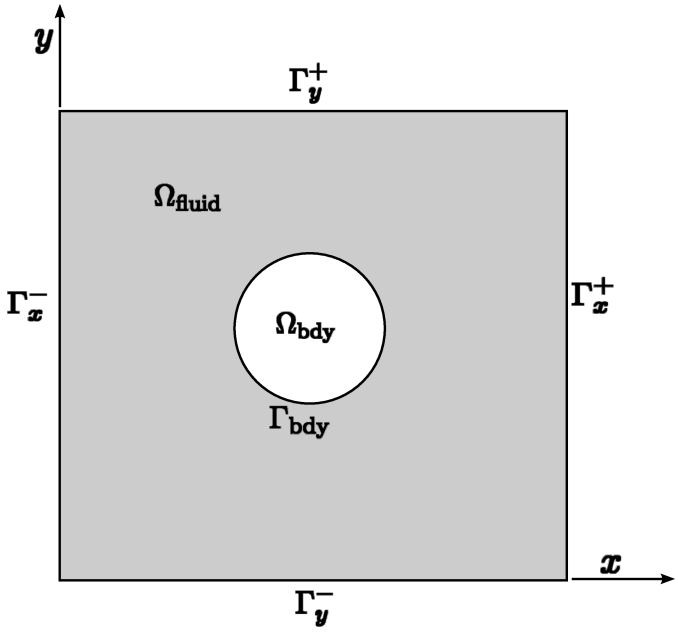


Figure 1: Geometrical description of problem.

The Navier-Stokes governing equations for an incompressible, laminar, constant

viscosity fluid are (see Figure (1)):

$$\begin{aligned}
&\frac{\partial u_i}{\partial t} + \frac{\partial (u_j u_i)}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \Delta u_i + f_i, && \text{in } \Omega_{\text{fluid}} \\
&\frac{\partial u_j}{\partial x_j} = 0, && \text{in } \Omega_{\text{fluid}} \\
&\mathbf{u} = \mathbf{u}_{\text{bdy}}, && \text{at } \Gamma_{\text{bdy}} \\
&\text{periodic B.C's}, && \text{at } \Gamma_{\infty} \\
&p = 0, && \text{at } \mathbf{x}_0
\end{aligned} \tag{1}$$

where u_i are the components of velocity, ρ density, p is pressure, Δ is the Laplace operator, x_j are the spatial coordinates, f_i a gravity field, and t is time. Einstein's summation convention on repeated indices is assumed. Periodic boundary conditions are imposed on the far boundary:

$$\Gamma_\infty = \Gamma_x^+ \cup \Gamma_x^- \cup \Gamma_y^+ \cup \Gamma_y^- \cup \Gamma_z^+ \cup \Gamma_z^- \tag{2}$$

this is:

$$\begin{aligned}
\mathbf{u}_{\Gamma_x^+} &= \mathbf{u}_{\Gamma_x^-} \\
p_{\Gamma_x^+} &= p_{\Gamma_x^-}
\end{aligned} \tag{3}$$

(and similar expressions for y and z). Also, pressure is defined up to a constant.

The numerical scheme is based on a Fractional Step-like solver, using the Quadratic Upstream Interpolation for Convection Kinematics (QUICK, see [Leo79]) on a staggered grid. QUICK is an advection scheme with very low numerical dissipation and is well suited for structured finite difference schemes.

The rectangular box $\Omega = \{0 \leq x/L_x, y/L_y, z/L_z \leq 1\}$ is discretized with $N_x \times N_y \times N_z$ continuity cells. The mesh size $h = L_j/N_j$ is assumed to be the same for all the spatial directions (see figure (2)). The mesh is *staggered*, so that cells for the discrete balance of continuity and each of the momentum equations do not coincide. The continuity cells are centered around $\mathbf{x} = (i + 1/2, j + 1/2, k + 1/2)h$ positions, and pressure values are assumed to be positioned at the center of these cells. The cells for x-momentum are shifted $h/2$ in the x direction, i.e. they are centered at $\mathbf{x} = (i, j + 1/2, k + 1/2)h$, and similarly, *mutatis mutandis*, for the other directions. Internal solid bodies will be treated as embedded and the details will be discussed later. The QUICK scheme can acommodate general boundary conditions, but in this article slip or non-slip will be represented in terms of thin solid bodies covering the boundary of the domain, due to the use of the FFT solver.

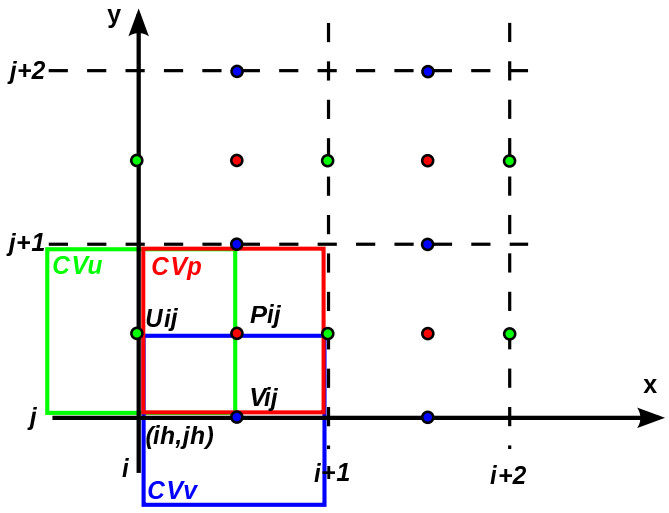


Figure 2: Staggered scheme (2D). Continuity cells (CVp, in red) are centered at the pressure nodes $\mathbf{x} = (i, j, k)h$. x-momentum cells (CVu, in green) are centered at the u nodes $\mathbf{x} = (i, j + 1/2, k + 1/2)h$, and y-momentum cells (CVv, in blue) are centered at the v nodes $\mathbf{x} = (i + 1/2, j, k + 1/2)h$.

2.1 The predictor step. QUICK advection scheme

In the first stage, the velocity field $\mathbf{u}^n$ is advanced to an intermediate state $\mathbf{u}^{n+1,p}$:

$$\frac{u_i^{n+1,p} - u_i^n}{\Delta t} = \frac{\partial}{\partial x_j}(u_j^n u_i^n) + f_i \tag{4}$$

where the superindex p stands for *predictor*. This predicted field may be not divergence free, so that it is corrected via a Poisson stage (or IOP, which is equivalent) to be explained later. The QUICK implementation of the predictor stage is discussed in this section.

Let's consider the x component of the balance equation. For the discretization of the x-momentum balance the corresponding x-momentum cell is used, which is shifted $h/2$ in the x direction, as described before. The discrete equation is obtained by a Finite Volume Method (FVM) approach, i.e. by performing a momentum balance on the cell as:

$$\Omega^c \left(\frac{u_x^{n+1,p}(\mathbf{x}) - u_x^n(\mathbf{x})}{\Delta t} \right)_{(i,j+1/2,k+1/2)} = M^n_{x,(i,j+1/2,k+1/2)} + \Omega^c f_{x,(i,j+1/2,k+1/2)} \tag{5}$$

where Ω^c is the cell volume. Note that all terms are evaluated at the center of the x-momentum cells. Discretization of temporal and external force field terms are straightforward. The nonlinear convection term is evaluated as:

$$\begin{aligned} M_{x,(i,j+1/2,k+1/2)} = {} & S_x \left[(u^c u^Q)_{i+1/2,j+1/2,k+1/2} - (u^c u^Q)_{i-1/2,j+1/2,k+1/2} \right] \\ & + S_y \left[(v^c u^Q)_{i,j+1,k+1/2} - (v^c u^Q)_{i,j,k+1/2} \right] \\ & + S_z \left[(w^c u^Q)_{i,j+1/2,k+1} - (w^c u^Q)_{i,j+1/2,k} \right] \end{aligned} \tag{6}$$

where S_x is the area of the cell faces perpendicular to the x-axis, and so on. The superscripts c and Q stand for *centered* and QUICK respectively, and the superindex n has been dropped since all quantities are evaluated in t^n. Each term represents the flux of momentum through a cell face. Each contribution involves the product of the velocity normal to the surface (which is approximated with a *centered* expression) and the velocity component which is being advected (which is approximated with a QUICK-upwinded expression). In Equation (6) the advected component is always u (since the x-momentum is considered) whereas the normal velocity may be u, v, or w, depending on the face of the cell to be considered. Note that in the first term, different approximations (centered or QUICK) are used for the same component u depending on whether it is used as a normal velocity or an advected component.

The centered approximations are:

$$\begin{aligned} u^c_{i+1/2,j+1/2,k+1/2} &= 1/2(u_{i,j+1/2,k+1/2} + u_{i+1,j+1/2,k+1/2}) \\ v^c_{i,j,k+1/2} &= 1/2(v_{i-1/2,j,k+1/2} + v_{i+1/2,j,k+1/2}) \\ w^c_{i,j+1/2,k} &= 1/2(w_{i-1/2,j+1/2,k} + w_{i+1/2,j+1/2,k}) \end{aligned} \tag{7}$$

Note that the right hand sides involve u values in the center of the corresponding x-momentum cells whereas the QUICK-upwinded approximations are:

$$\begin{aligned} u^Q_{i-1/2,j+1/2,k+1/2} &= \begin{cases} (c_0 u_i + c_1 u_{i-1} + c_2 u_{i-2})_{j+1/2,k+1/2}, & \text{if } u^c_{i-1/2,j+1/2,k+1/2} > 0 \\ (c_0 u_{i-1} + c_1 u_i + c_2 u_{i+1})_{j+1/2,k+1/2}, & \text{if } u^c_{i-1/2,j+1/2,k+1/2} < 0 \end{cases} \\ u^Q_{i,j,k+1/2} &= \begin{cases} \left(c_0 u_{j+1/2} + c_1 u_{j-1/2} + c_2 u_{j-3/2}\right)_{i,k+1/2}, & \text{if } v^c_{i,j,k+1/2} > 0 \\ \left(c_0 u_{j-1/2} + c_1 u_{j+1/2} + c_2 u_{j+3/2}\right)_{i,k+1/2}, & \text{if } v^c_{i,j,k+1/2} < 0 \end{cases} \\ u^Q_{i,j+1/2,k} &= \begin{cases} \left(c_0 u_{k+1/2} + c_1 u_{k-1/2} + c_2 u_{k-3/2}\right)_{i,j+1/2}, & \text{if } w^c_{i,j+1/2,k} > 0 \\ \left(c_0 u_{k-1/2} + c_1 u_{k+1/2} + c_2 u_{k+3/2}\right)_{i,j+1/2}, & \text{if } w^c_{i,j+1/2,k} < 0 \end{cases} \end{aligned} \tag{8}$$

The coefficients c_j are $c_0 = 3/8$, $c_1 = 6/8$, $c_1 = -1/8$. They are the basis of QUICK and guarantee that the upwinded approximations are precise to third order.

The advection step is applied to the whole domain $\Omega = \Omega_{\text{bdy}} \cup \Omega_{\text{fluid}}$, independently of the position of the cell (inside the body, boundary, or fluid). If some of the involved cell values fall outside the fluid domain, they are obtained from interior values via the periodic boundary conditions, i.e. all indices i, j, k are assumed to be cyclic *modulo* N_x, N_y, N_z.

2.2 The projection step in FSM

Once the predicted field $\mathbf{u}^{n+1,p}(\mathbf{x})$ is computed, it may not satisfy the divergence condition (second line in Equation (1)), neither the boundary conditions:

$$\mathbf{u}^{n+1,p} = \mathbf{u}_{\text{bdy}}, \quad \text{at } \Gamma_{\text{bdy}} \tag{9}$$

where $\mathbf{u}_{\text{bdy}}$ is the velocity of the body. In the standard Fractional Step method these conditions are enforced by computing a Poisson stage:

$$\mathbf{u}^{n+1} = \mathbf{u}^{n+1,p} - \nabla P \tag{10}$$

where $P = (\Delta t/\rho)p$ and p is pressure. P is computed through the following Poisson equation:

$$\begin{aligned} \Delta P &= \nabla \cdot \mathbf{u}^{n+1,p}, \quad \text{in } \Omega_{\text{fluid}} \\ \left.\frac{\partial p}{\partial n}\right|_{\Gamma_{\text{bdy}}} &= (\mathbf{u}_{\text{bdy}} - \mathbf{u}^{n+1,p}) \cdot \hat{\mathbf{n}} \end{aligned} \tag{11}$$

where $\hat{\mathbf{n}}$ is the unit vector normal to Γ_{bdy} pointing towards the fluid. An alternative form to enforce these conditions is the Iterated Orthogonal Projection (IOP) method (see [MCPN08]). The idea behind IOP is that as the mesh is structured and cartesian, there are fast solvers (as Multigrid (MG) or Fast Fourier Transform (FFT)) that may solve the Poisson equation very efficiently, provided there are no *holes* (i.e. bodies) in the domain. Given a non-solenoidal vector field $\mathbf{u}$, the *orthogonal projection* operator $\mathbf{\Pi}_{\text{div}}$ defined by:

$$\mathbf{u}' = \mathbf{\Pi}_{\text{div}}(\mathbf{u}) \quad \Longrightarrow \quad \begin{cases} \mathbf{u}' = \mathbf{u} - \nabla P \\ \Delta P = \nabla \cdot \mathbf{u} \end{cases} \quad \text{in } \Omega \tag{12}$$

projects orthogonally with respect to the L_2 norm, onto the subspace of solenoidal fields S_{div}. Note that the Poisson equation is solved in the whole domain, so that the projected velocity $\mathbf{u}'$ may be non zero in Ω_{bdy}. $\mathbf{u}'$ is then projected onto the subspace S_{bdy} of velocity fields that satisfy the solid boundary condition:

$$\mathbf{u}'' = \mathbf{\Pi}_{\text{bdy}}(\mathbf{u}') \quad \Longrightarrow \quad \begin{cases} \mathbf{u}'' = \mathbf{u}_{\text{bdy}}, & \text{in } \Omega_{\text{bdy}} \\ \mathbf{u}'' = \mathbf{u}', & \text{in } \Omega_{\text{fluid}} \end{cases} \tag{13}$$

In the case that the solid body is moving then $\mathbf{u}_{\text{bdy}} \neq 0$ and then S_{bdy} is an *affine subspace*. It is easy to see that $\mathbf{\Pi}_{\text{bdy}}$ is also an orthogonal projection operator with respect to L_2. Of course, if the $\mathbf{u}''$ velocity field satisfies also the continuity condition (i.e. $\mathbf{u}'' \in S_{\text{div}}$) then $\mathbf{u}'' \in S_{\text{div}} \cap S_{\text{bdy}}$, and the algorithm stops, i.e. $\mathbf{u}'' = \mathbf{u}^{n+1}$. In the general case a sequence

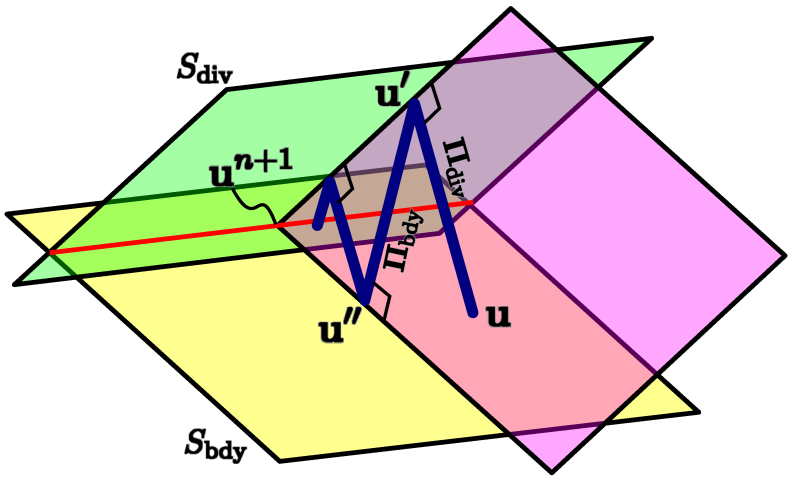


Figure 3: Geometric interpretation of the convergence of IOP. Starting at a velocity field $\mathbf{u}$ the $\mathbf{\Pi}_{\text{div}}$ operator projects it to $\mathbf{u}'$ int the space of solenoidal velocity fields, but perhaps violating the boundary condition. Then the $\mathbf{\Pi}_{\text{bdy}}$ projects $\mathbf{u}'$ on $\mathbf{u}''$ in the space of velocity fields that satisfy the boundary conditions. It can be shown that the generated sequence is convergent, provided that both projection operators are orthogonal in the same norm.

$\mathbf{w}^k$ with $\mathbf{w}^0 = \mathbf{u}^{n+1,p}$ and $\mathbf{w}^\infty = \mathbf{u}^{n+1}$ is generated via the successive application of the projection operators $\mathbf{\Pi}_{\text{div}}$ and $\mathbf{\Pi}_{\text{bdy}}$:

$$\mathbf{w}^{k+1} = \mathbf{\Pi}_{\text{bdy}}\mathbf{\Pi}_{\text{div}}\mathbf{w}^k \tag{14}$$

It can be shown that the sequence converges [MCPN08], *provided that the projection operators are orthogonal in the same norm*, as it is the case here. The geometric interpretation of the algorithm is shown in Figure (3).

2.3 Rate of convergence of IOP

The convergence of IOP is related to the eigenvalue spectrum of the combined operator $\mathbf{G} = \mathbf{\Pi}_{\text{bdy}}\mathbf{\Pi}_{\text{div}}$. If an eigenfunction of that operator can be found, i.e.:

$$\mathbf{G}\bar{\mathbf{v}} = \gamma\bar{\mathbf{v}} \tag{15}$$

then the IOP sequence for that velocity field will be:

$$\bar{\mathbf{v}}, \gamma\bar{\mathbf{v}}, \gamma^2\bar{\mathbf{v}}, \ldots, \gamma^k\bar{\mathbf{v}}, \ldots \tag{16}$$

Such an eigenfunction can be found in a particular geometry and sheds light on the convergence of IOP and the behavior of the rate of convergence with respect to refinement

and the geometrical characteristics of the domain, as for instance aspect ratio of the immersed bodies.

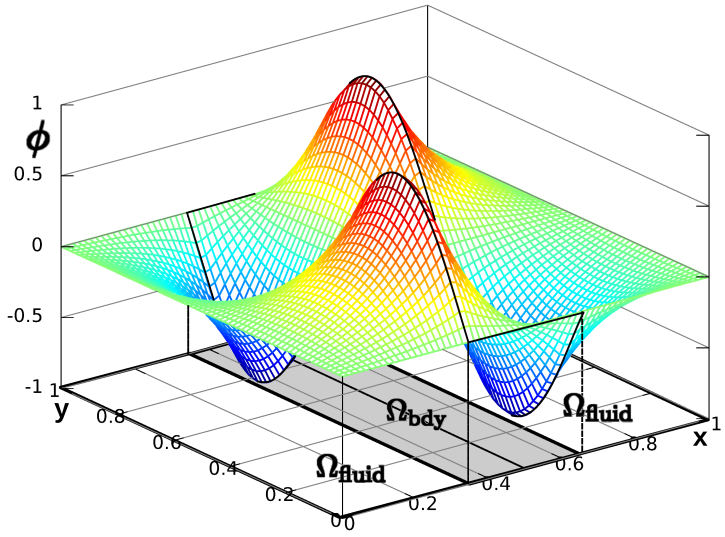


Figure 4: Eigenmode with slowest convergence rate for IOP.

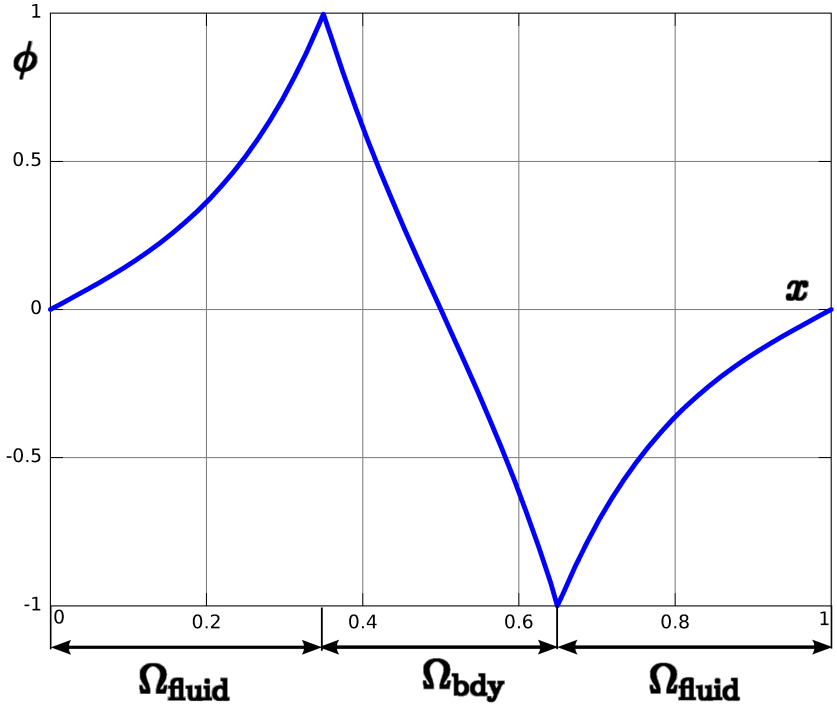


Figure 5: Eigenmode with slowest convergence rate for IOP. Cut at $y = 0.25$.

Consider a 2D problem in $\Omega = [0, L] \times [0, L]$, and a solid body which is a vertical strip of width b centered at $x = L/2$, i.e.:

$$\Omega_{\text{bdy}} = \{|x - L/2| < b/2\} \tag{17}$$

Consider now the following function:

$$\phi(\mathbf{x}) = \begin{cases} \dfrac{\sinh(kx)}{\sinh(kL_f)}\,\sin(ky), & \text{for } x < x_- \\ -\dfrac{\sinh(k(x-L/2))}{\sinh(kb/2)}\,\sin(ky), & \text{for } x_- < x < x_+ \\ -\dfrac{\sinh(k(L-x))}{\sinh(kL_f)}\,\sin(ky), & \text{for } x > x_+ \end{cases} \tag{18}$$

where $x_\pm = (L \pm b)/2$ are the vertical boundaries of the solid domain, $L_f = (L-b)/2$ is the half length of fluid domain, and $k = 2\pi/L$ is the wave number. This function is shown at Figures (4) and (5). By construction, $\Delta\phi = 0$ except at the boundary $\Gamma_{\text{bdy}} = \{|x - L/2| = b/2\}$.

Let's start with the following velocity field:

$$\mathbf{u} = \begin{cases} \nabla\phi, & \text{in } \Omega_{\text{fluid}} \\ 0, & \text{in } \Omega_{\text{bdy}} \end{cases} \tag{19}$$

By construction $\nabla \cdot \mathbf{u} = 0$ in Ω_{fluid}, Ω_{bdy}. At the interface Γ_{bdy}, the divergence can be computed in the sense of distributions and:

$$\begin{aligned} \nabla \cdot \mathbf{u} &= (u|_{(x_-)^+} - u|_{(x_-)^-})\delta(x - x_-) + (u|_{(x_+)^+} - u|_{(x_+)^-})\delta(x - x_-) \\ &= -\left.\frac{\partial\phi}{\partial x}\right|_{(x_-)^-} \delta(x - x_-) + \left.\frac{\partial\phi}{\partial x}\right|_{(x_+)^+} \delta(x - x_+) \\ &= k\coth(kL_f)\,\sin(ky)\,[-\delta(x - x_-) + \delta(x - x_+)] \end{aligned} \tag{20}$$

where δ is Dirac's δ distribution.

Next, the second equation of (12) must be solved for P. The result is that P is a scalar multiple of ϕ, i.e. $P = c\phi$. It results that:

$$\begin{aligned} \Delta P &= c\left(\left.\frac{\partial\phi}{\partial x}\right|_{(x_-)^+} - \left.\frac{\partial\phi}{\partial x}\right|_{(x_-)^-}\right)\delta(x - x_-) \\ &\quad + c\left(\left.\frac{\partial\phi}{\partial x}\right|_{(x_+)^+} - \left.\frac{\partial\phi}{\partial x}\right|_{(x_+)^-}\right)\delta(x - x_+) \\ &= ck\left[\coth(kb/2) + \coth(kL_f)\right]\,\sin(ky)\,[-\delta(x - x_-) + \delta(x - x_+)] \end{aligned} \tag{21}$$

and then:

$$c = \frac{\coth(kL_f)}{\coth(kb/2) + \coth(kL_f)} \tag{22}$$

Applying the correction $-\nabla P$ to $\mathbf{u}$ (the first line in Equation (12):

$$\begin{aligned} \mathbf{u}' &= \mathbf{u} - \nabla P \\ &= \begin{cases} (1-c)\,\nabla\phi, & \text{in } \Omega_{\text{fluid}} \\ c\,\nabla\phi, & \text{in } \Omega_{\text{bdy}} \end{cases} \end{aligned} \tag{23}$$

Finally, the application of the $\mathbf{\Pi}_{\text{bdy}}$ consists in simply setting to zero the velocity field in Ω_{bdy}, so that:

$$\begin{aligned}\mathbf{u}'' &= \mathbf{\Pi}_{\text{bdy}}\mathbf{u}' \\ &= \begin{cases} (1-c)\,\nabla\phi, & \text{in } \Omega_{\text{fluid}} \\ 0, & \text{in } \Omega_{\text{bdy}} \end{cases}\end{aligned} \tag{24}$$

Comparing Equation (24) with Equation (19) shows that $\mathbf{u}$ is an eigenvalue of $\mathbf{G} = \mathbf{\Pi}_{\text{bdy}}\mathbf{\Pi}_{\text{div}}$ with eigenvalue:

$$\begin{aligned}\gamma &= 1 - c \\ &= 1 - \frac{\tanh(kb/2)}{\tanh(kb/2) + \tanh(kL_f)}\end{aligned} \tag{25}$$

As expected, the amplification factor γ results to be $0 < \gamma < 1$, otherwise the scheme would be non-convergent.

A family of eigenfuctions can be constructed in the same way, simply replacing the wave number k by higher frequency ones, with the restriction that an integer number of wavelengths must be present in the y direction, i.e.:

$$k_m = \frac{2\pi m}{L}, \quad m = 1, 2, \ldots \tag{26}$$

and the corresponding amplification factors are:

$$\gamma_{\text{as},m} = \frac{\tanh(k_m L_f)}{\tanh(k_m b/2) + \tanh(k_m L_f)} \tag{27}$$

Of course, they fall all in the range $0 < \gamma_m < 1$. It can be shown also that for $b < L/2$ the γ_m is a decreasing sequence $\gamma_{m+1} < \gamma_m$, so the mode with slowest rate of convergence (highest γ) is that one corresponding to k_1, i.e. Equation (25).

Note that the modes given by Equation (18) are *antisymmetric* (hence the "as" superscript) with respect to the center of the strip, i.e. $\phi(x - L/2) = -\phi(L/2 - x)$. There is another branch of eigenvalues corresponding to symmetric modes, in that case the amplification factors are:

$$\gamma_{\text{s},m} = \frac{\tanh(k_m b/2)}{\tanh(k_m b/2) + \tanh(k_m L_f)} \tag{28}$$

where the "s" subindex stands for *symmetric*. For $b < L/2$ it can be shown that for this branch of amplification factors $0 < \gamma_{\text{s},m} < 1/2$, so the slowest rate of convergence is still the first antisymmetric mode.

Also, it can be shown that the set of symmetric and antisymmetric modes together represent a complete set of eigenfunctions for the Stekhlov operators [PNS06, PS05, SDP+06], and then the convergence rate given by Equation (25) is not an upper bound but rather the spectral radius of the amplification matrix **G** and hence the best estimate for the rate of convergence of the algorithm.

2.4 Convergence in the discrete case

The rate of convergence given in Equation (25) corresponds to the continuum case. However, discretization affects mostly the high frequency (large k_m) modes, but as it has been shown the global rate of convergence is governed by the slowest mode, which is slightly affected by refinement. In fact, as the mesh is refined ($h \to 0$) the convergence rate approaches the value of the continuum Equation (25). This means that *the rate of convergence for IOP does not degrade with refinement*.

2.5 Convergence and aspect ratio

Note that for high aspect ratio (i.e. $L/b \to \infty$) the amplification factor $\gamma \to 1$, i.e. convergence degrades. In fact for $b \ll L$ the amplification factor is:

$$\begin{aligned} \gamma &\approx 1 - \frac{\pi b/L}{\tanh(2\pi)} \\ &\approx 1 - 3.14\frac{b}{L} \end{aligned} \tag{29}$$

i.e., the rate of convergence r (as number of iterations per order of magnitude) is:

$$\begin{aligned} \gamma^r &= 1/10 \\ r &= -\frac{\log 10}{\log \gamma} \approx 0.37\frac{L}{b}[\text{iter/OM}] \end{aligned} \tag{30}$$

which is proportional to the aspect ratio and *iter/OM* means *"iterations per order of magnitude"*.

This result has been confirmed with numerical experiments for other geometries as well. The physical explanation is that IOP has good convergence when $\mathbf{\Pi}_{\text{div}}$ is close to the solution of the Poisson problem *on the fluid domain only* (recall that $\mathbf{\Pi}_{\text{div}}$ solves the Poisson problem in the whole domain $\Omega_{\text{fluid}} + \Omega_{\text{bdy}}$). Conversely, convergence is poor when the inclusion of the solid body domain Ω_{bdy} distorts too much the solution and this is just what happens when the aspect ratio is large for a mode like Equation (18). Note that the

sources on opposite sides of the strip have different sign and then a large (spurious) flow is generated inside the body.

3 The Accelerated Global Preconditioning

The algorithm proposed in this paper is based in the IOP, in the sense of using a fast solver on the whole mesh, but the main difference is that this global solution is used as a preconditioner for the Preconditioned Conjugate Gradient (PCG) method. Recall that the PCG method is an *accelerated* convergence method, i.e. the rate of convergence increases during iteration, hence the name Accelerated Global Preconditioning (AGP) (see [Kel95]).

Consider a situation like that in Figure (6), with a solid body described by the boundary Γ_{bdy}. This is embedded in a structured grid of constant mesh size h. In the traditional Fractional Step Method a Poisson problem is solved *outside* the body. Recall that pressure nodes are at the center of the continuity cells (marked with dashed lines in the figure). In order to construct an approximation to the Poisson equation a Finite Element (FEM) mesh is considered where the pressure nodes are at the corner of the finite elements (marked as solid lines in the figure). Note that the cell mesh is *dual* to the continuity cell mesh, i.e. the center of the continuity cells (which are the pressure nodes) are at semi-integer positions $(i + 1/2, j + 1/2, k + 1/2)h$ coincident with the corner of the finite elements and, *vice versa*, the center of the finite elements are then at full integer positions (ih, jh, kh). The center of the finite elements are computed and it is checked whether the element falls inside or outside the body. In this way the body is approximated by a *staircase geometry* as is shown in gray in the figure. (This degrades de convergence to $O(h)$ instead of $O(h^2)$ and can be fixed by more sofisticated techniques as the Immersed Boundary Method [Pes02], but this will be not discussed here.) As it is usual in FEM discretizations the imposition of the homogeneous Neumann condition is done by simply assembling only those elements that are in the fluid part. The other elements that are not in gray are *ghost elements* and are not assembled for the solution of the Poisson problem. Only the pressure in the nodes connected to some element that is assembled are relevant, i.e. those that are marked in blue and red. Those that are marked in green are *ghost* and are not computed. From those that are computed, the set that are surrounded completely by computed elements (and then are not connected to ghost elements) are classified as *interior to the fluid*, (subindex F) and the rest are classified as *boundary* (subindex B, filled in red in the figure). So the Poisson

problem is:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{31}$$

and the splitting of nodes induces a matrix splitting like this:

$$\begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FB} \\ \mathbf{A}_{BF} & \mathbf{A}_{BB} \end{bmatrix} \begin{bmatrix} \mathbf{x}_F \\ \mathbf{x}_B \end{bmatrix} = \begin{bmatrix} \mathbf{b}_F \\ \mathbf{b}_B \end{bmatrix} \tag{32}$$

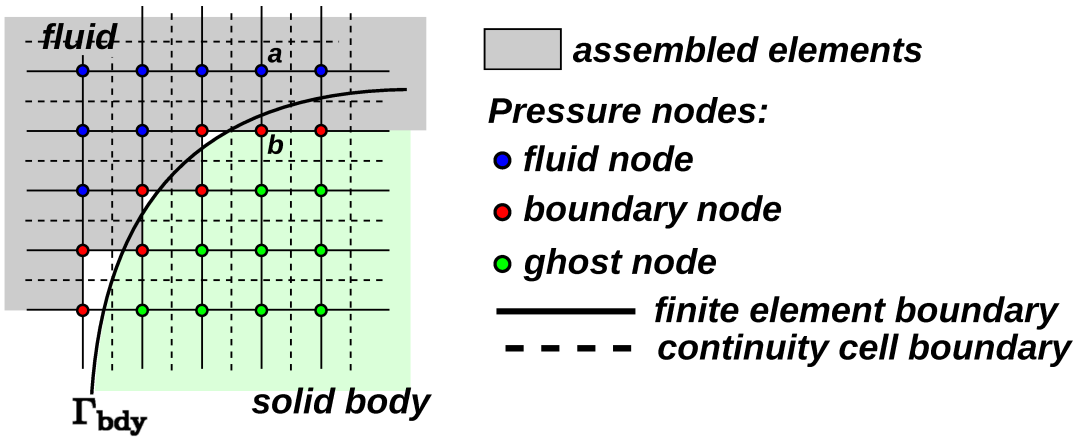


Figure 6: Description of nodes and elements used in the AGP.

In the following, the preconditioning operator P will be described. First consider the whole matrix for the Laplace operator $\tilde{\mathbf{P}}$, i.e. assembling over fluid and ghost cells for F, B, and G nodes. Note that a symbol different from $\mathbf{A}$ is used for this matrix since it is assembled on a different set of element/cells. The preconditioning is then defined formally as $\mathbf{y}_{FB} = \mathbf{P}\mathbf{x}_{FB}$, where $\mathbf{x}_{FB}$ is the solution of:

$$\begin{bmatrix} \tilde{\mathbf{P}}_{FF} & \tilde{\mathbf{P}}_{FB} & \tilde{\mathbf{P}}_{FG} \\ \tilde{\mathbf{P}}_{BF} & \tilde{\mathbf{P}}_{BB} & \tilde{\mathbf{P}}_{BG} \\ \tilde{\mathbf{P}}_{GF} & \tilde{\mathbf{P}}_{GB} & \tilde{\mathbf{P}}_{GG} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{FB} \\ \mathbf{x}_G \end{bmatrix} = \begin{bmatrix} \mathbf{y}_{FB} \\ \mathbf{0}_G \end{bmatrix} \tag{33}$$

However it can be seen that

- $\tilde{\mathbf{P}}_{FF} = \mathbf{A}_{FF}$ since the F nodes are those for which all neighbor elements are assembled in the Poisson problem.
- $\tilde{\mathbf{P}}_{FB} = \mathbf{A}_{FB}$, and $\tilde{\mathbf{P}}_{BF} = \mathbf{A}_{BF}$ since for instance, such a coefficient would link nodes as a and b in the figure. This coefficient comes from the assembly of all the elements that are connected to a *and* b, but since a is an F node, it means that all elements connected to a are assembled.
- $\tilde{\mathbf{P}}_{FG} = \tilde{\mathbf{P}}_{GF} = 0$ since F nodes are only connected to fluid elements and G are only connected to ghost elements, so that they can not share an element.

So:

$$\begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FB} & \mathbf{0} \\ \mathbf{A}_{BF} & \tilde{\mathbf{P}}_{BB} & \tilde{\mathbf{P}}_{BG} \\ \mathbf{0} & \tilde{\mathbf{P}}_{GB} & \tilde{\mathbf{P}}_{GG} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{FB} \\ \mathbf{x}_G \end{bmatrix} = \begin{bmatrix} \mathbf{y}_{FB} \\ \mathbf{0}_G \end{bmatrix} \tag{34}$$

$\mathbf{x}_G$ can be eliminated from the bottom line, and then the following equations is obtained:

$$\begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FB} \\ \mathbf{A}_{BF} & \tilde{\mathbf{P}}_{BB} - \tilde{\mathbf{P}}_{BG}\tilde{\mathbf{P}}_{GG}^{-1}\tilde{\mathbf{P}}_{GB} \end{bmatrix} \mathbf{x}_{FB} = \mathbf{y}_{FB} \tag{35}$$

This allows to obtain an explicit expression for the preconditioning matrix:

$$\mathbf{P} = \begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FB} \\ \mathbf{A}_{BF} & \tilde{\mathbf{P}}_{BB} - \tilde{\mathbf{P}}_{BG}\tilde{\mathbf{P}}_{GG}^{-1}\tilde{\mathbf{P}}_{GB} \end{bmatrix} \tag{36}$$

A first consequence of this expression is that most eigenvalues of the preconditioned matrix will be 1. Consider the subspace of all vectors $\mathbf{x}$ such that the boundary component B is null, i.e. the non null entries are only on F nodes, then:

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{P}\mathbf{x} \\ \mathbf{P}^{-1}\mathbf{A}\mathbf{x} &= \mathbf{x} \end{aligned} \tag{37}$$

so that $\mathbf{x}$ is an eigenvector with eigenvalue 1.

The proposed technique is named Accelerated Global Preconditioning (AGP) because the solution of Equation (33) is done on an infinite mesh with periodic boundary conditions so that it can be solved via FFT transform, which is very efficient, but the whole analysis does not depend on how the solution of this system is done; i.e. it may be obtained by Multigrid iteration as well.

3.1 Convergence of AGP

Note that in the IOP method after the first application of $\mathbf{u}' = \mathbf{\Pi}_{\text{div}}\mathbf{u}$ the velocity field $\mathbf{u}'$ is solenoidal everywhere. After the $\mathbf{u}'' = \mathbf{\Pi}_{\text{bdy}}\mathbf{u}'$ step, the field $\mathbf{u}''$ is solenoidal at both Ω_{fluid} and Ω_{bdy} (it is zero if the body is stationary, and a rigid motion if it is in movement) but not at the interface. In the second application of $\mathbf{\Pi}_{\text{div}}$ the right hand side in Equation (12) is zero everywhere, except for a possible concentrated source term at the interface, i.e. a Dirac's δ.

Something similar happens for AGP, after the first iteration of the PCG the right hand side for the preconditioning step (Equation (33)) is zero everywhere, except at the boundary

nodes. In Section (3) the AGP was introduced in the discrete version; the continuum counterpart will be used now for assessing its convergence properties. Both the Poisson problem (Equation (31)) and the preconditioner are written as mappings between surface values at Γ_{bdy} and solenoidal pressure fields that satisfy the Laplace equation everywhere, except at the interface.

First, the Poisson equation in the fluid domain Ω_{fluid} is written in abstract form as:

$$\mathcal{A}(\psi) = g \tag{38}$$

where g is a source term in Γ_{bdy} and ψ a function defined on Ω_{fluid} that satisfies:

$$\begin{cases} \Delta\psi = 0, & \text{in } \Omega_{\text{fluid}} \\ \nabla\psi \cdot \hat{\mathbf{n}} = -g, & \text{at } \Gamma_{\text{bdy}} \end{cases} \tag{39}$$

where $\hat{\mathbf{n}}$ is the normal to Γ_{bdy} pointing into the fluid.

Next, the preconditioner is written in abstract form as:

$$\mathcal{P}(\psi) = g \tag{40}$$

where g is also a source term on Γ_{bdy} and ψ is defined on Ω_{fluid} and is obtained from the following problem. Let ψ' defined in $\Omega = \Omega_{\text{fluid}} \cup \Omega_{\text{bdy}}$, satisfying:

$$\begin{cases} \Delta\psi' = 0, & \text{in } \Omega_{\text{fluid}} \text{ and } \Omega_{\text{bdy}} \\ [(\nabla\psi')^+ - (\nabla\psi')^-] \cdot \hat{\mathbf{n}} = -g, & \text{at } \Gamma_{\text{bdy}} \end{cases} \tag{41}$$

where $(\nabla\psi')^+$ is evaluated on the side of the fluid region and $(\nabla\psi')^-$ from the body region. Now, the AGP algorithm can be put in the continuum case as solving Equation (38) with PCG, using Equation (40) as a preconditioner. Then, rates of convergence for the AGP an be estimated in terms of the condition number of the preconditioned operator:

$$\kappa = \text{cond}(\mathcal{P}^{-1}\mathcal{A}) \tag{42}$$

The condition number κ will be computed for the strip problem used before (see Section (2.3)) for assessing the convergence of IOP.

The eigenfuctions of the preconditioned problem should satisfy:

$$\mathcal{P}^{-1}\mathcal{A}\psi = \lambda\psi \tag{43}$$

If we define $g = \mathcal{A}\psi$, then it is equivalent to:

$$\begin{aligned} \mathcal{P}\psi &= (1/\lambda)g \\ \mathcal{A}\psi &= g \end{aligned} \tag{44}$$

and it can be rewritten as:

$$\begin{aligned} \Delta\psi' &= 0, && \text{in } \Omega_{\text{fluid}}, \Omega_{\text{bdy}} \\ \frac{\partial\psi'}{\partial n} &= -g, && \text{at } \Gamma_{\text{bdy}} \\ \left(\frac{\partial\psi'}{\partial n}\right)^{+} - \left(\frac{\partial\psi'}{\partial n}\right)^{-} &= -(1/\lambda)g, && \text{at } \Gamma_{\text{bdy}} \end{aligned} \tag{45}$$

and set ψ to the restriction of ψ' to Ω_{fluid}. It can be shown that the function ϕ defined in Equation (18) satisfies this set of equations. Effectively ϕ by construction satisfies the first line of Equation (45) and the second and third lines give:

$$\begin{aligned} g &= k_m \coth(k_m L_f) \sin(k_m y) \\ \frac{1}{\lambda} g &= k_m [\coth(k_m L_f) + \coth(k_m b/2)] \sin(k_m y) \end{aligned} \tag{46}$$

so it gives:

$$\begin{aligned} g &= k_m \coth(k_m L_f) \sin(k_m y) \\ \lambda_{\text{as},m} &= \frac{\coth(k_m L_f)}{\coth(k_m L_f) + \coth(k_m b/2)} \end{aligned} \tag{47}$$

Again, it can be shown that for $b < L/2$ λ_m is a monotonically increasing sequence, and in addition $\lambda_\infty = 1/2$. The subindex "as" stands for *antisymmetric*. Now for the symmetric modes it can be shown that:

$$\lambda_{\text{s},m} = \frac{\tanh(k_m L_f)}{\tanh(k_m L_f) + \tanh(k_m b/2)} \tag{48}$$

and that $\lambda_{\text{s},m}$ is monotonically decreasing and $1 = \lambda_{\text{s},1} > \lambda_{\text{s},m} > \lambda_{\text{s},\infty} = 1/2$. Then, the highest eigenvalue is $\lambda_{\text{s},1} = 1$, and the lowest is $\lambda_{\text{as},m}$. Then, the condition number is:

$$\begin{aligned} \kappa &= \frac{\lambda_{\text{s},1}}{\lambda_{\text{as},1}} \\ &= \frac{\tanh(\pi b/L) + \tanh(2\pi L_f/L)}{\tanh(\pi b/L)} \end{aligned} \tag{49}$$

Again, this approximation holds also in the discrete case, since both the maximum and minimum eigenvalues have low frequency. Following the same arguments in Section (2.4), it can be shown that *the condition number of the preconditioned AGP operator and hence the rate of convergence for PCG (with AGP) does not degrade under refinement*.

3.2 High aspect ratio limit

For a high aspect ratio strip ($L/b \gg 1$) the limit is:

$$\begin{aligned} \kappa &\approx \frac{\tanh(2\pi)}{\pi} \frac{L}{b} \\ &\approx 0.32 \frac{L}{b} \end{aligned} \tag{50}$$

3.3 Spectrum of AGP operator and IOP convergence

It can be shown that the amplification factors for IOP (see Equation (27) and Equation (28)) are related to the eigenvalues of the AGP method by the simple relation:

$$\gamma_m = 1 - \lambda_m \tag{51}$$

so that the slowest rate of convergence (the γ_m closer to 1) corresponds to $m = 1$, i.e.:

$$\begin{aligned} &\gamma = 1 - \lambda_{\min} \\ &r \approx \log 10 \frac{1}{\lambda_{\min}}, \quad \text{for } \lambda_{\min} \ll 1 \end{aligned} \tag{52}$$

4 Comparison of IOP and AGP

The differences and similitudes of both methods are summarized here:

- Both solvers are based on the fact that the Poisson equation on the fluid domain can be approximated by solving on the global domain (fluid+solid). Of course, this represents more computational work than solving the problem only in the fluid, but this can be faster in a structured mesh using some fast solvers such as Multigrid or FFT.
- Both solvers have their convergence governed by the spectrum of the AGP preconditioned operator $\mathcal{P}^{-1}\mathcal{A}$, more precisely on its lowest eigenvalue $\lambda_{\min}$.
- It has been shown that for a fixed geometry $\lambda_{\min} = O(1)$, i.e. it *does not degrade with refinement*, so that IOP has a linear convergence with limit rate $O(1)$, i.e. it does not degrade with refinement.
- By the same reason, the condition number for AGP *does not degrade with refinement*.
- IOP is a *stationary method* and its limit rate of convergence is given by:

$$\begin{aligned} &\|\mathbf{r}^{n+1}\| \leq \gamma \|\mathbf{r}^n\| \\ &\gamma = 1 - \lambda_{\min} \\ &\lambda_{\min} = \frac{\tanh(kb/2) + \tanh(kL_f)}{\tanh(kL_f)} \qquad \text{(strip of aspect ratio } L/b) \\ &\gamma \approx 1 - 3.14 \frac{b}{L}, \qquad\qquad (L/b \gg 1) \end{aligned} \tag{53}$$

- AGP is an *accelerated method* (PCG) and the convergence for AGP can be assessed from

the condition number of the preconditioned operator, which is:

$$\begin{aligned}\kappa(\mathcal{P}^{-1}\mathcal{A}) &= \frac{1}{\lambda_{\min}} \\ &= \frac{\tanh(kL_f)}{\tanh(kb/2)+\tanh(kL_f)}, \qquad \text{(strip of aspect ratio } L/b\text{)} \\ &\approx 0.32\,\frac{L}{b} \qquad\qquad (L/b \gg 1)\end{aligned} \tag{54}$$

- In fact, it can be shown that the iterates for both IOP and AGP can be put as polynomials on the preconditioned operator $\mathcal{P}^{-1}\mathcal{A}$, and due to the *minimization property characteristic of Krylov space methods* like CG (see [Kel95]) the convergence of AGP is always better than that of IOP.
- IOP iterates over both the velocity and pressure fields, whereas AGP iterates only on the pressure vector (which is better for implementation on GPU's architectures, since reduces memory access).
- As the minimum eigenvalue is proportional to the reciprocal of the aspect ratio (i.e. $\lambda_{\min} \propto b/L$) the convergence of both algorithms degrade for high aspect ratio bodies. In the case of IOP, the rate of convergence is proportional to L/b. In the case of AGP the condition number of the preconditioned operator $\kappa(\mathcal{P}^{-1}\mathcal{A})$ is proportional to L/b. Due to the estimates of rate of convergence for CG as compared to stationary methods, it is expected that the convergence rates of AGP will be comparatively much better than that for IOP for geometries with high aspect ratio bodies.

4.1 Solving the Poisson equation with the FFT

Both IOP and AGP are based on the fact that a fast solver in the whole domain (solid+fluid) exists. There are at least two possibilities: MG and FFT. The second has been chosen in this work, and the basis of this component of the algorithm will be given here.

The linear system to be solved is denoted as:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{55}$$

Let $\tilde{\mathbf{x}} = \mathbf{O}\mathbf{x}$ denote the application of the Discrete Fourier Transform (DFT) to a vector $\mathbf{x}$. It can be shown that $\mathbf{O}$ is an orthogonal matrix (i.e. $\mathbf{O}^T\mathbf{O} = \mathbf{I}$), where $(\,\cdot\,)^T$ denotes transpose. By applying the transformation to Equation (55) the transformed equation is obtained:

$$(\mathbf{O}\mathbf{A}\mathbf{O}^T)(\mathbf{O}\mathbf{x}) = (\mathbf{O}\mathbf{b}) \tag{56}$$

It can be shown that the transformed system is diagonal (i.e. $\mathbf{O}\mathbf{A}\mathbf{O}^T = \mathbf{D}$, with $\mathbf{D}$ a diagonal matrix) provided that the matrix $\mathbf{A}$ is invariant under translations, i.e. the stencil

of the operator is the same for all the cells of the mesh. Also, the boundary conditions must be periodic (but this restriction will be removed below, see Section (5.1.3)).

Now consider the following algorithm that computes the solution of the linear system

- Compute the transform of the right hand side: $\tilde{\mathbf{b}} = \mathbf{O}\mathbf{b}$
- Solve the diagonal system in the transformed basis $\tilde{\mathbf{x}} = \mathbf{D}^{-1}\tilde{\mathbf{b}}$,
- Obtain the antitransformed solution vector by applying the inverse DFT: $\mathbf{x} = \mathbf{O}^T\tilde{\mathbf{x}}$.

The total operation count for this algorithm is two DFT's, plus one element-by-element vector multiply (the reciprocals of the values of the diagonal of $\mathbf{D}$ are precomputed), For N a power of 2 (i.e. $N = 2^p$) the Fast Fourier Transform (FFT) is an algorithm that computes the DFT (and its inverse) in $O(N\log(N))$ operations, then the cost of the algorithm is $O(N\log(N))$.

Another possibility is Multigrid (MG), which is a *stationary iterative* method with cost $O(N)$ *for a given tolerance*, i.e. its cost is $O(N\log(\epsilon))$, where ϵ is the tolerance used as stopping criterion. On the other hand the FFT solver is a *direct method* with operation count $O(N\log(N))$ to solve the linear system to machine precision.

5 Numerical experiments

5.1 Convergence of IOP and AGP. Condition number of AGP

5.1.1 Convergence of IOP iteration

Figures (7) and (8) show the convergence of IOP iteration for a $16 \times 16 \times 16$ and $64 \times 64 \times 64$ meshes on a computational domain which is a cube of unit side $L = 1$. The body domain is a sphere of radius 0.3, i.e. $\Omega_{\text{bdy}} = \{||\mathbf{x}|| \leq R = 0.3\}$. As can be seen the IOP iteration curves exhibit the *typical linear rate of convergence of stationary methods*. The convergence for both meshes start at a high convergence rate of less than 3 iter/OM (iterations per order of magnitude), and then they switch to a slower convergence of 13.7 iter/OM for the 16^3 mesh and 12.3 iter/OM for the fine 64^3 mesh. However note that *the rates of convergence are independent of mesh refinement*.

5.1.2 Condition number for AGP does not degrade with refinement

The condition number of matrices for the Poisson problem have been computed with and without preconditioning (see Figure (9)).

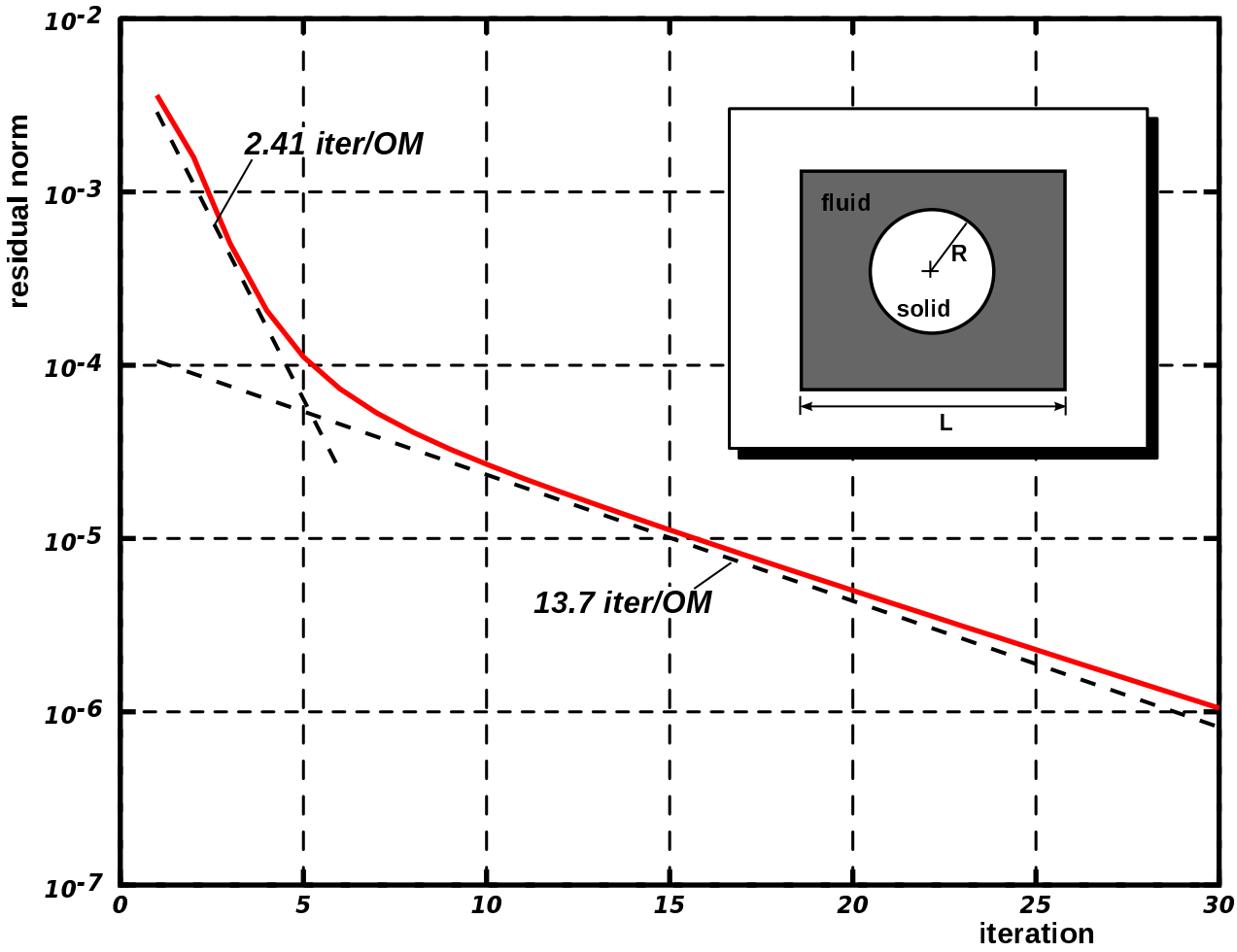


Figure 7: Convergence of IOP loop for a sphere of $R = 0.3$ in a cube of $L = 1$, with a coarse mesh of $16 \times 16 \times 16$.

- In the experiments the number of cells N_x along x ranges from 8 to 64.
- The Poisson problem is computed selecting the quadrangles whose center fall outside the body problem.
- In all cases the domain is the unit square with periodic boundary conditions.
- The bodies considered are: cylinder of radius 0.2, a vertical strip of width 0.5, and a square of side 0.5.
- The condition numbers are computed with Octave `cond()` function.

Note that in all cases the non preconditioned matrix condition number grows as $O(N_x^2)$, whereas *with the preconditioning it remains constant*.

5.1.3 Bodies with large aspect ratio

Note that both IOP and AGP preconditioning are based on the inclusion of the solid domain in the computation of the pressure Poisson equation. As it have been shown, this causes

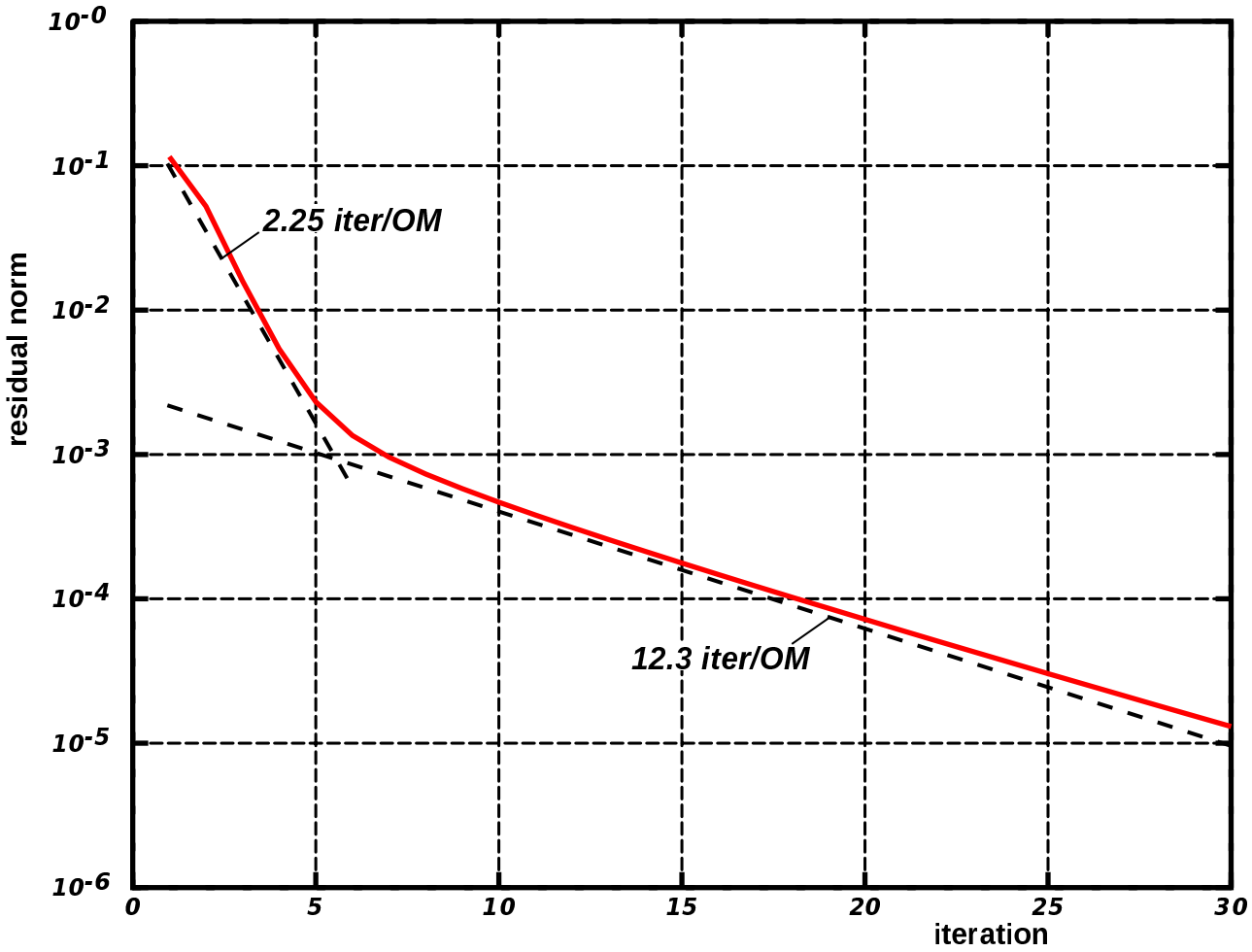


Figure 8: Convergence of IOP loop for a sphere of $R = 0.3$ in a cube of $L = 1$, with a fine mesh of $64 \times 64 \times 64$.

convergence to degrade when objects with large aspect ratio are present in the domain. Consider the case where the fluid occupies the *interior* of a square:

$$\Omega_{\text{fluid}} = \{(x, y) \quad / \quad \max(|x - L/2|, |y - L/2|) < L/2 - b\} \tag{57}$$

where b is the width of the wall (see Figure (10)).

Two cases are considered, a fixed wall width of thickness $b = 0.05L$, and the case $b = h$, i.e. the width of the wall is of just one element. So as the mesh is refined, the aspect ratio of the wall increases. The condition number for the problem with and without preconditioning are shown.

In the case of a fixed value $b = 0.05$ the condition number of the preconditioned case is bounded, whereas for the case of $b = h$ the condition number increases with aspect ratio. If no preconditioning is used the condition number increases as $O(N_x^2)$.

This case is of practical interest because it is a workaround to solve problems with solid boundary conditions. In its simplest form the FFT solver requires periodic boundary

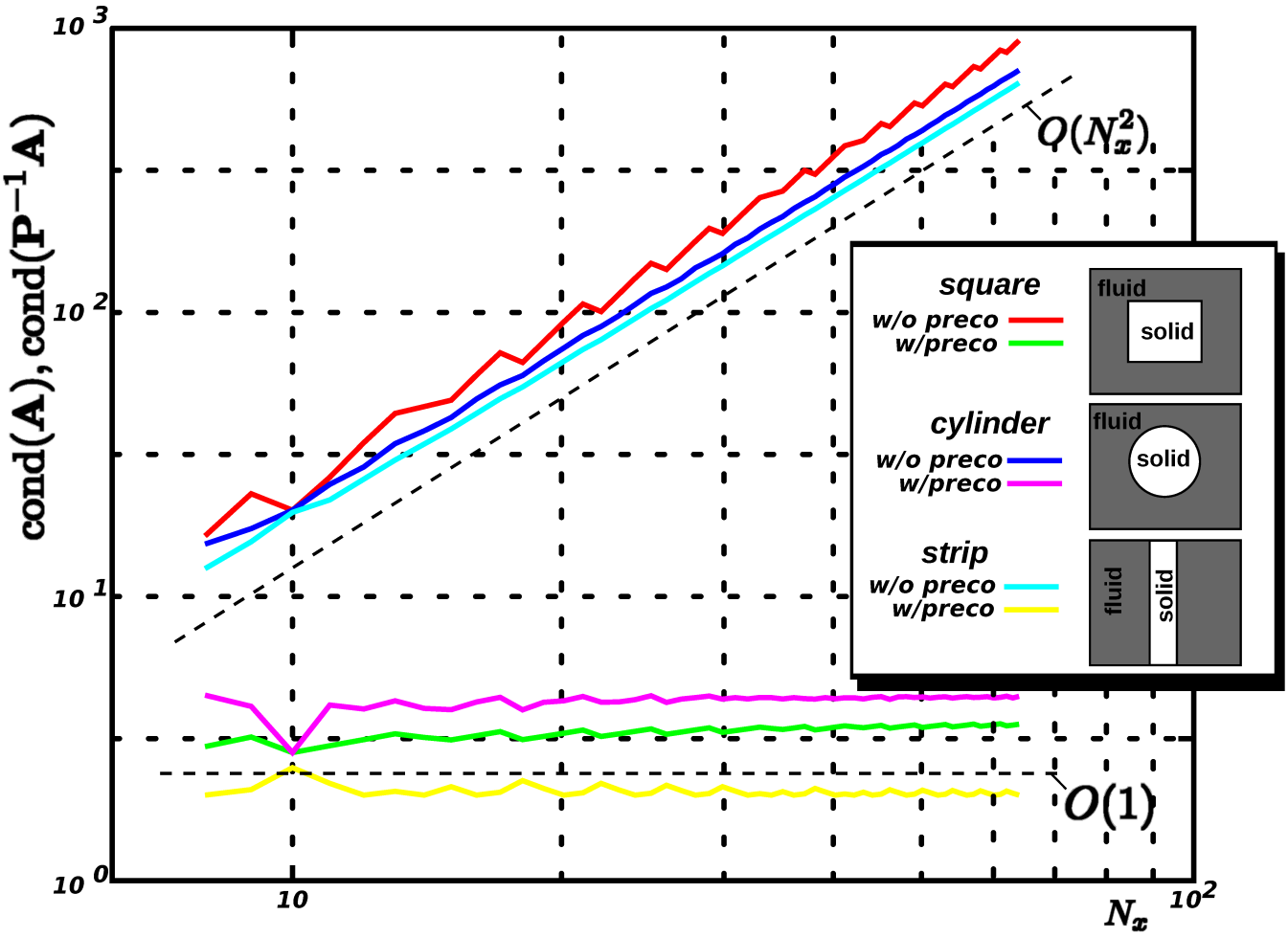


Figure 9: Condition number of Poisson problem with and without the AGP preconditioning.

conditions in order to be applied. Other common boundary conditions like homogeneous Dirichlet and Neumann boundary conditions can be also implemented using different flavors of the FFT, but if all combinations are to be considered (i.e. Dirichlet on some sides of the box, and Neumann on the others) it requires a tedious programming of all the cases and dispatch to the appropriate FFT routine.

But a solid boundary condition can be represent also by simply putting a thin layer of solid at the boundary. However, this can be inefficient if the additional work represented by the layer is significant. This numerical example shows that a layer as thin as a 5% of the side length of the box can be used, with very little degradation of the condition number. In such a case the increase in the computational time is not significant, approximately 30%, because the layer covers all the 6 sides of the box, but it can be shown that smaller widths b can be used for finer meshes. In the numerical results shown in following sections with closed cavities the width of the solid layer is 2.5% of the domain length. In such a case the

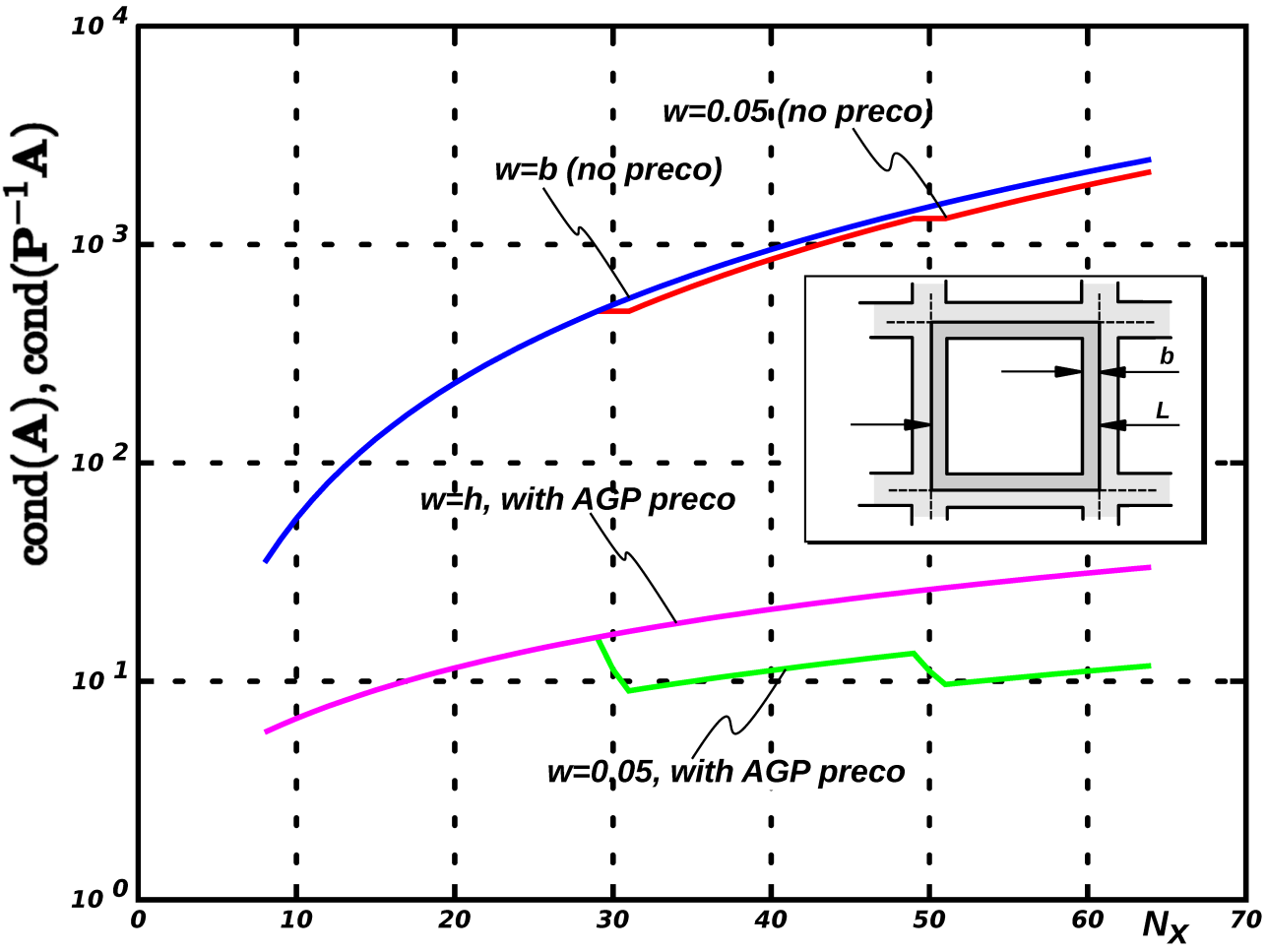


Figure 10: Condition number for Poisson problem on a square, with and without AGP preconditioning.

computation overhead due to the wall layer is only 15%.

5.1.4 Convergence histories for IOP and AGP compared

In Figure (11) the convergence histories for AGP and IOP in a 2D problem, with a circular body of radius $R = 0.3$, and several degrees of refinement $N_x = 8, 16, 32, 64$, where N_x is the number of cells per side are shown.

It is observed that the convergence histories tend to a fixed rate of convergence as the mesh is refined, in fact the convergence histories are almost the same for $N_x = 32$ and 64. This verifies the estimates discussed in Section (2.4) and Section (3.1).

The rate of convergence is much higher for AGP. Note that if higher (weaker) tolerances (for instance 10^{-3}) are used then the convergence of both methods is similar. This is acceptable for non-critical applications like video-game and special effects, but usually not for engineering computations. If lower (stronger) tolerances (let's say 10^{-6}) are enforced then the difference is substantial.

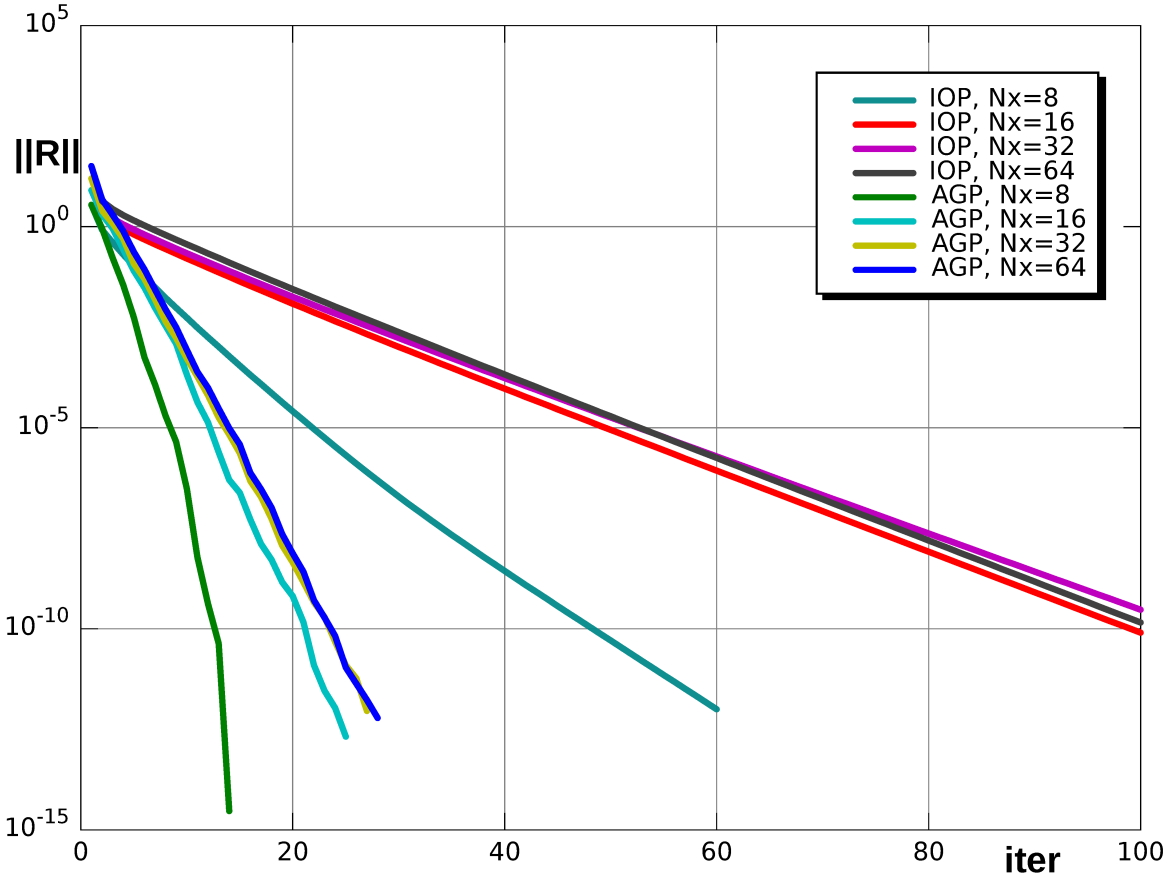


Figure 11: Convergence histories for a 2D problem with a circular body of radius $R = 0.3$. Convergence is shown for both AGP and IOP, and several refinements. (N_x is the number of elements per side).

5.2 Computational efficiency on GPU hardware

5.2.1 Computing times of FFT on GPU and CPU hardware

As it has been discussed, for large problems the most consuming time component of the algorithm are the two FFT applications per AGP iteration (same for IOP), so the efficiency of the available libraries will be assessed.

The GPU implementation was coded using the Compute Unified Device Architecture (CUDA) from Nvidia [NBGS08, Far11] (release 4.2, V0.2.1221). CUDA comes with an efficient FFT implementation called the CUFFT library. On the other hand, for CPU the Fastest Fourier Transform in the West (FFTW) [FJ98, FJ12] (release 3.1-2) library was used. The computing rates for this two libraries on the Nvidia GTX-580 GPU's, and processors Intel i7-3820@3.60GHz, and Intel W3690@3.47Ghz are shown in Figure (12), (13), and (14). The computing rate in Gflops is computed as:

$$\text{rate[Gflops]} = 10^{-9} \times \frac{2N_v \log_2(N_v)}{\text{elapsed time [s]}} \tag{58}$$

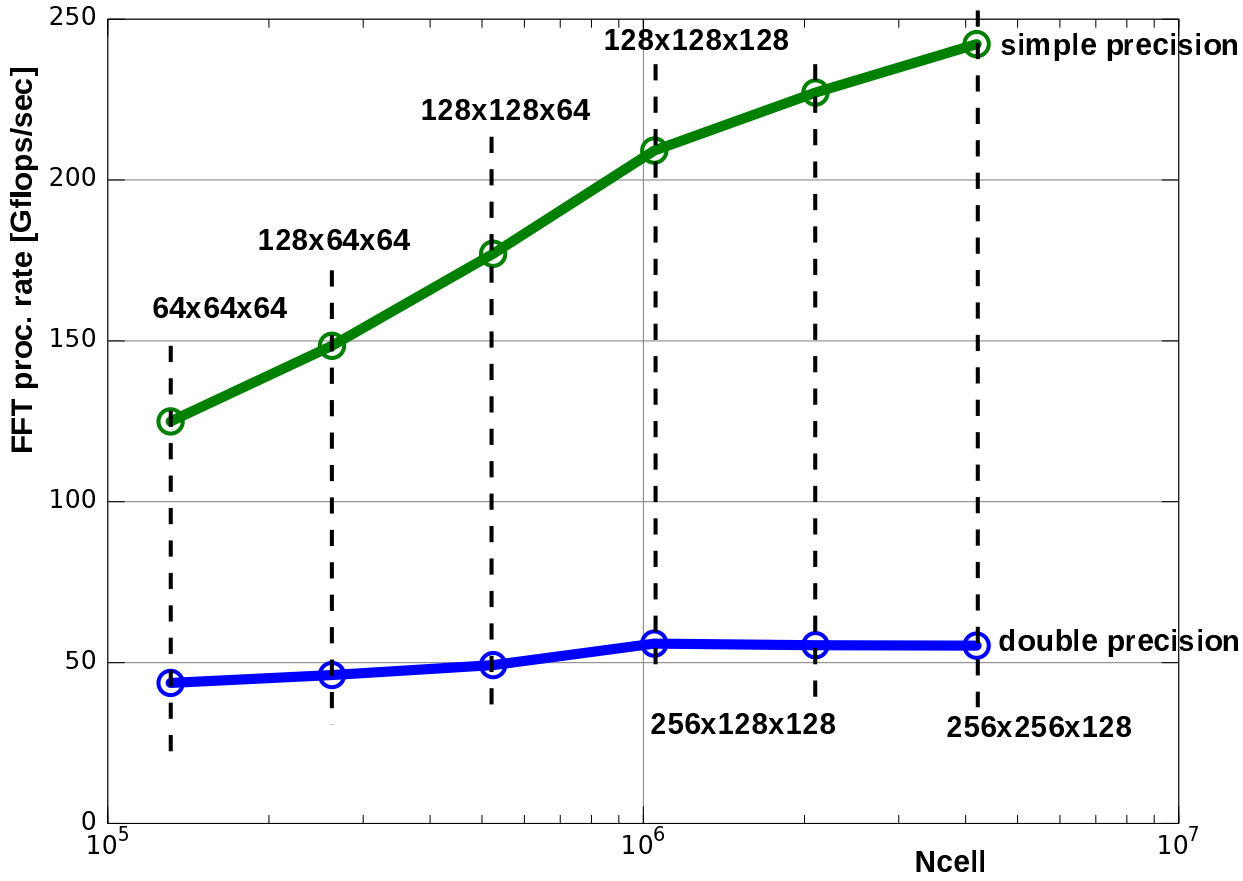


Figure 12: Computing rates for the CUFFT implementation on the Nvidia GTX-580 GPU.

where $N_v = N_x \cdot N_y \cdot N_z/2$ is the total size of the complex vector to be transformed. Note that this is half the number of cells, since using the *R2C* (for *Real to Complex*) flavor of the FFT the number of operations can be reduced by a half. The computing rate for the GTX-580 is near 240 Gflops in simple precision for meshes of $256 \times 128 \times 128$ (8 million cells, 4 million elements in the complex vector). For double precision the rate drops by almost a factor of 4. Note that previous boards *not* in the Tesla family had a typical speed relation of 8:1 from simple to double precision, so this ratio 4:1 signifies an improvement for the GTX-580. Typical boards on the Tesla family have a speed ratio of 2:1.

On the other hand the fastest CPU processor tested is the Sandy Bridge i7-3820 which (multi-threaded in its 6 cores) peaks at 20 Gflops for vectors of size $O(10^5)$. However this performance drops at almost 8 Gflops for large vectors, when the vector does not fit in the processor's cache. So, in double precision for large vectors there is a speedup of a factor 8 between the FFT on the GPU board and the CPU.

Note also that, in contrast with the deterioration in performance of the CPU's, the computing rate of the CUFFT in simple precision seams to steadily increase as the vector

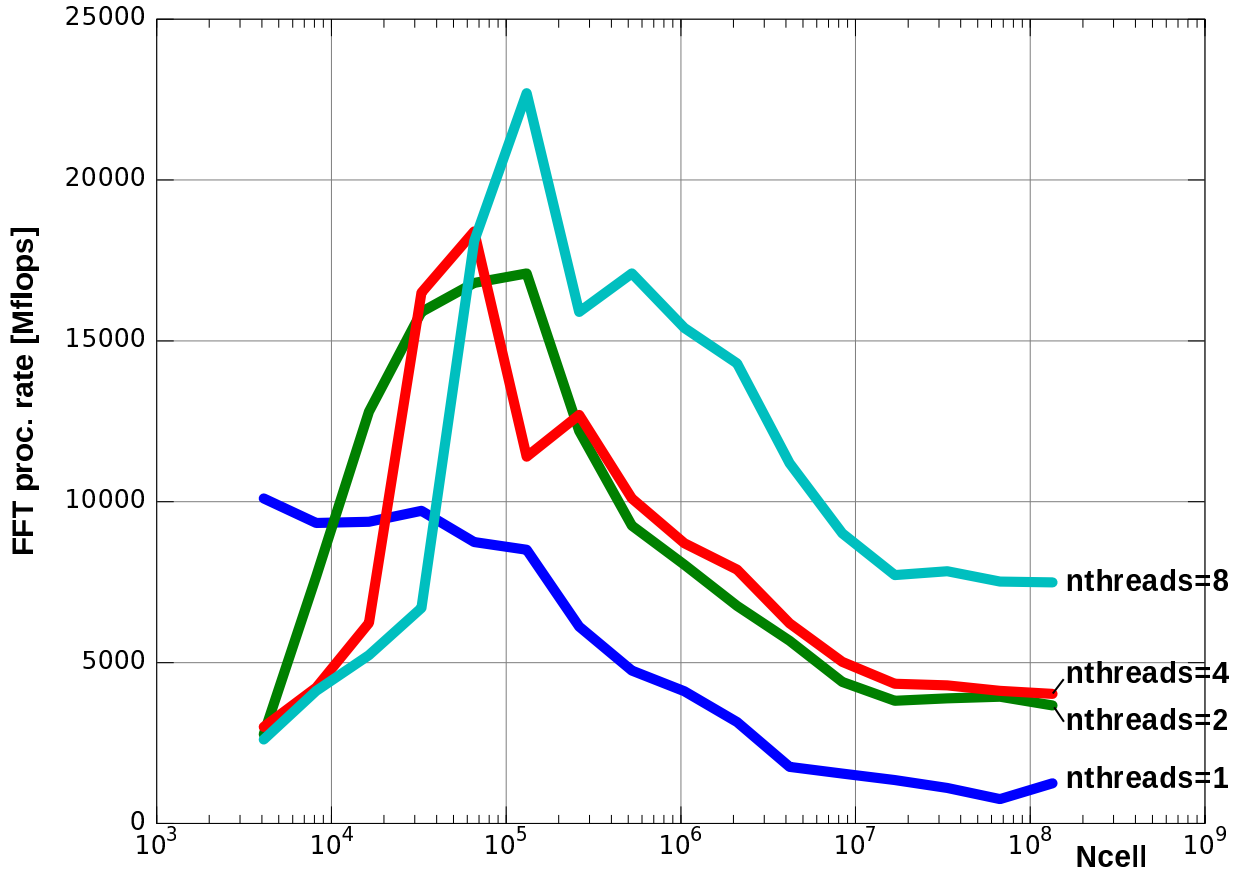


Figure 13: Computing rates for the FFTW (SMP) implementation on the Intel i7-3820 (Sandy Bridge) CPU (double precision).

length increases, whereas the double precision shows a small increase in performance. This means that it is likely that the performance will be kept for boards with a larger device memory (the GTX-580 has 3 GB RAM) allowing for larger computations in a single device.

5.2.2 Computing rates

In Figure (15) the computing rates in Mcell/sec for the code presented in this article on an Nvidia GTX-580 GPU, and a Nvidia Tesla C2050, with single (SP) and double precision (DP), are shown. In DP for large meshes it reaches a rate of 60 Mcell/sec. As a reference, the same algorithm was implemented in CPU using the GNU g++ compiler (with optimization flags `-O3 -funroll-loops`), obtaining a rate on one core of the Intel i7-3820@3.47 GHz (Sandy Bridge) of 1.7 Mcell/sec. Assuming perfect scalability a maximum of 6.8 Mcell/sec at most would be reached using the four cores of the i7-3820, which translates in a speedup of at least 7:1 for the GPU over this CPU.

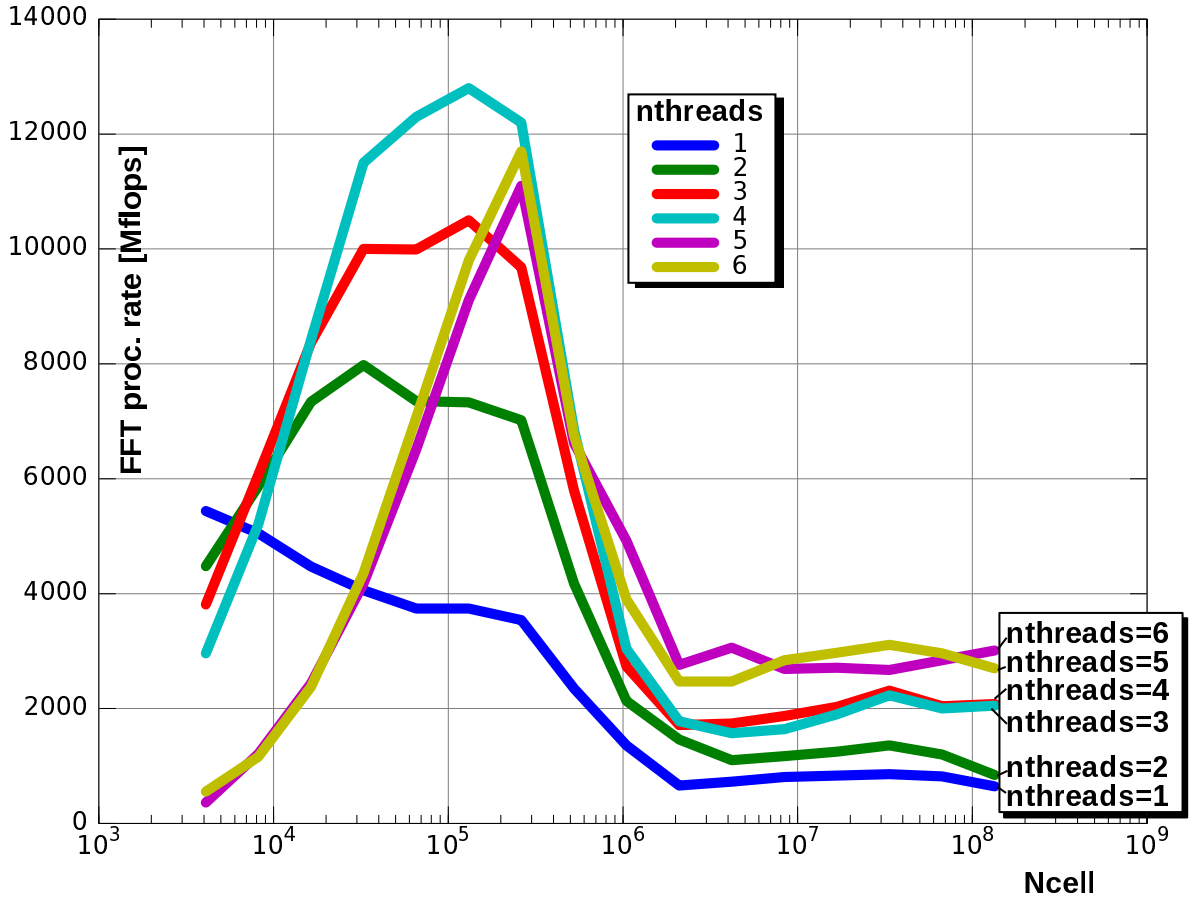


Figure 14: Computing rates for the FFTW (SMP) implementation on the Intel W3690 (Nehalem) CPU (double precision).

5.3 Real time computing

Many applications in engineering need Real Time Computing (RTC), i.e. to have a code fast enough such that $T_{\text{comp}} \leq T_{\text{sim}}$, where T_{comp} is the computing time needed for simulating T_{sim} seconds of the physical problem. For instance this is the case in applications where the computations are needed for take some action back on the physical process, as in control or disaster management.

The approach presented here allows to do RTC in moderately large meshes. Consider for instance a mesh of 128^3 cells ($\approx$ 2 Mcell). The computing time on the GTX-580 in SP is (see Figure (15)) 140 Mcell/sec, so each time step takes approximately 2 Mcell/(140 Mcell/sec) = 0.014 secs per time step, i.e. 70 steps per second can be computed.

A von Neumann stability analysis shows that the QUICK stabilization scheme is inconditionally unstable if advanced in time with Forward Euler. With a second order Adams-Bashfort scheme the critical Courant-Friedrichs-Lewy (CFL) number is CFL<0.588

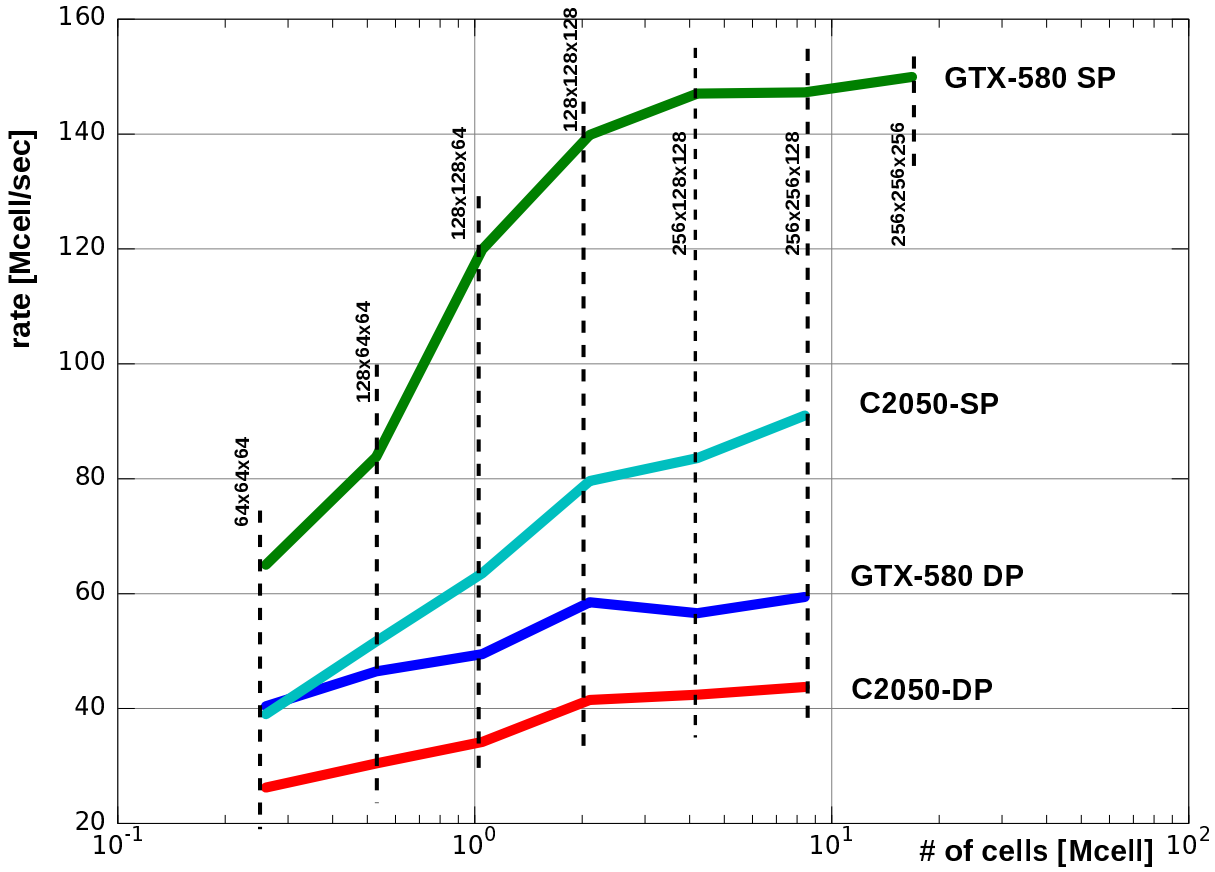


Figure 15: Computing rates in Mcell/sec for the algorithm presented in this paper in an Nvidia GTX-580 GPU, and a Nvidia Tesla C2050, with single (SP) and double precision (DP).

for an scalar advection problem, and for Navier-Stokes (at high Reynolds numbers) it is somewhat lower, CFL<0.5. If $L = 1\,[\mathrm{m}]$, and maximum velocity $u = 1\,[\mathrm{m/s}]$, mesh step is $h = 1/128[\mathrm{m}]$, then the critical time step is $\Delta t = 0.5h/u = 0.004\,[\mathrm{s}]$, so that $T_{\mathrm{sim}} = 70\Delta t = 0.28\,[\mathrm{s}]$ can be computed in $T_{\mathrm{comp}} = 1\,[\mathrm{s}]$ of computing time. It means that for such a mesh the computations go 1:4 slower than the physical process. Other approach to RTC is to circumvent the restriction of CFL < 1 characteristic of explicit methods [INLO12].

5.4 Flow simulations

Numerical simulations of several flows involving moving bodies are shown in Figures (16)-(20). In all cases (except for the case of the example in Section (5.4.3)) the flows represent a body moving inside a square or cubic cavity of length side $1\,[\mathrm{m}]$. In order to circumvent the restriction of periodic boundaries intrinsic to the FFT solver, a thin layer (2.5% of the square or cubic domain side length) is defined as a fixed body. In all cases the color corresponds

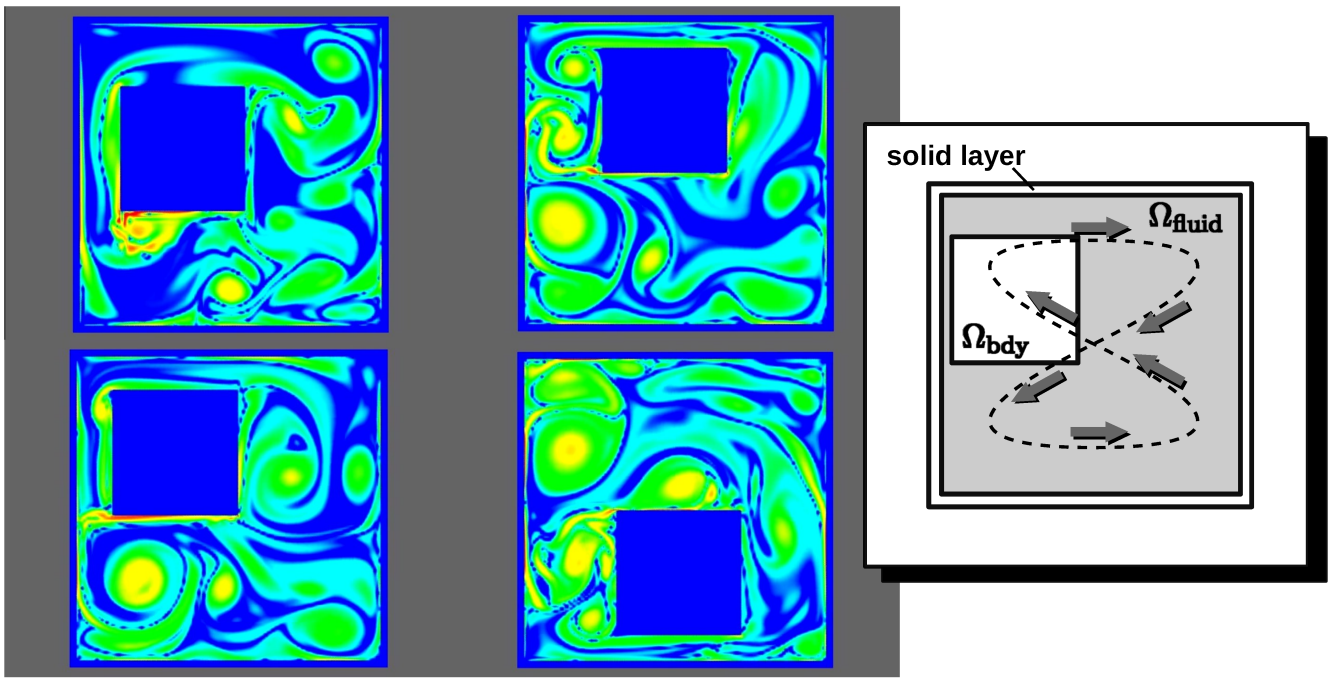


Figure 16: Colormap of $\log_{10}(|\omega|)$ for a square of side $L_s = 0.4$[m] moving in a square domain of side $L = 1$[m]. The square moves forming a Lissajous *8*-shaped curve.

to $\log_{10}(|\omega|)$, i.e. the absolute magnitude of the vorticity vector $\omega = \nabla \times \mathbf{u}$ in logarithmic scale. This quantity helps in the visualization of boundary layers, since the magnitude of vorticity has variations of several orders of magnitude there at high Reynolds numbers. In 2D cases the mesh was 128×128 and in 3D cases $128 \times 128 \times 128$. In all cases the side of the domain (square in 2D, cube in 3D) was $L = 1\,[\mathrm{m}]$ and kinematic viscosity was $\nu = 6.33 \times 10^{-5}\,[\mathrm{m}^2/\mathrm{s}]$.

5.4.1 Square moving in curved trajectory

The body is a square of side $L_s = 0.4\,[\mathrm{m}]$, and the center of the body (x_c, y_c) describes an *8*-shaped Lissajous curve, described by:

$$\begin{aligned} x_c &= \frac{L}{2} + A\cos(2\omega t) y_c = \frac{L}{2} + A\cos(\frac{\pi}{2} + \omega t) \\ \omega &= 1[\mathrm{s}^{-1}],\ A = 0.2[\mathrm{m}] \end{aligned} \tag{59}$$

As the body displaces fluid high levels of vorticity can be observed at the vertices. As the simulation progresses large vortices remain rotating in the fluid with long filamentary vorticity layers that are a characteristic 2D feature (they are unstable in 3D).

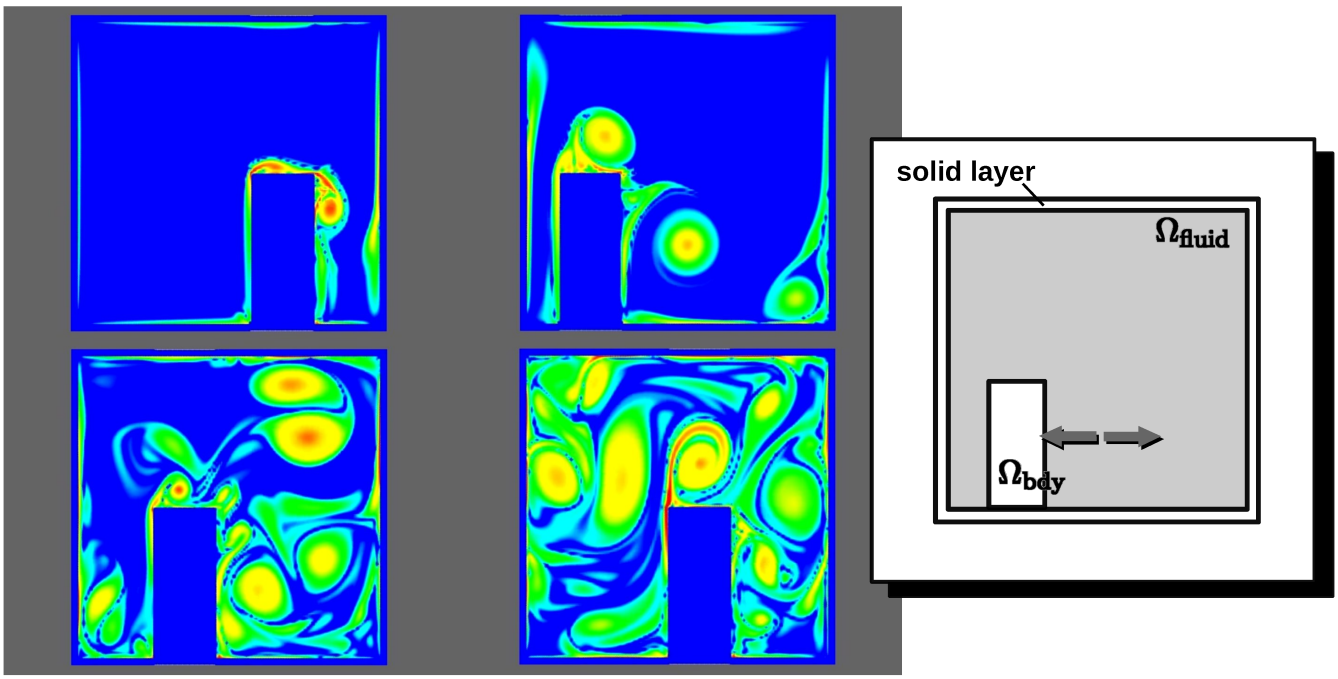


Figure 17: Colormap of $\log_{10}(|\omega|)$ for a rectangle sliding on the bottom of the domain.

5.4.2 Moving rectangular obstacle

The body is a rectangle of height $H = 0.5\,[\text{m}]$ and width $W = 0.2\,[\text{m}]$. An harmonic horizontal displacement as follows:

$$\begin{aligned} x_c &= (L/2) + A\cos(\omega t) \\ \omega &= 1[\text{s}^{-1}],\ A = 0.3[\text{m}] \end{aligned} \tag{60}$$

is imposed. As the body displaces fluid a large concentration of vorticity is observed in the upper corner of the body, with characteristic trailing filamentary vortex layers that detach from the corners.

5.4.3 Square moving vertically with mean horizontal flow

In this example the exterior boundary of the computational domain is not at rest, but rather it is intended to generate a mean flow that impinges on the body. This freestream flow is obtained with a layer of width 0.025[m] at the left and right sides were a positive x velocity of $u = 1\,[\text{m/s}]$ is imposed. Periodic boundary conditions are imposed in the vertical y direction. The body is a square of side $L_s = 0.4\,[\text{m}]]$, the center of the body (x_c, y_c) is centered in the x direction and experiences an harmonic vertical movement:

$$\begin{aligned} y_c &= (L/2) + A\cos(\omega t) \\ \omega &= 0.5[\text{s}^{-1}],\ A = 0.2\,[\text{m}] \end{aligned} \tag{61}$$

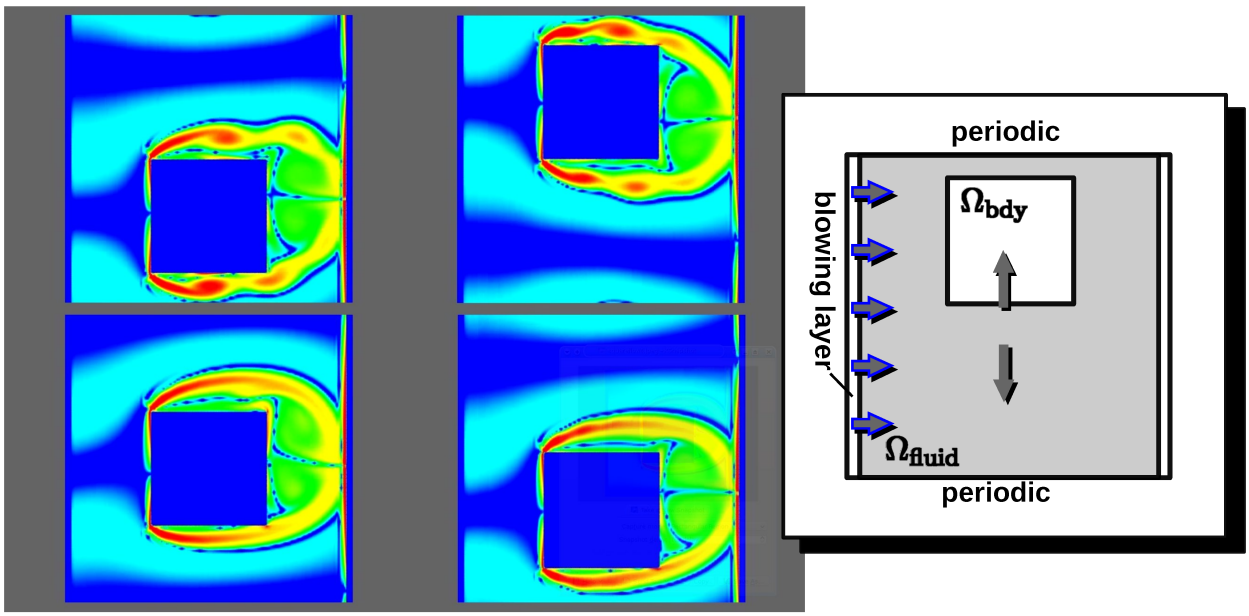


Figure 18: Colormap of $\log_{10}(|\omega|)$ for a square body performing harmonic motion in the vertical direction with a cross flow in the horizontal direction.

An accelerating boundary layer is formed at the left side facing the fluid stream. The boundary layer accelerate towards the corners and detach there. If the vertical movement were at a constant velocity then the flow would be equivalent to a fixed body with an impinging stream at an angle of attack. A notable feature of the flow is that when the body reaches the extreme positions in the y direction the vortex layers become unstable and start shedding vortices, whereas when the body is moving the vortex layer stabilizes.

5.4.4 Moving cube

This is a 3D case. The center (x_c, y_c, z_c) of a cube of side $L_s = 0.4\,[\mathrm{m}]$ is describing a Lissajous 8-shaped figure in the $z = 0.66\,[\mathrm{m}]$ plane, as follows:

$$\begin{aligned} x_c &= L/2 + A\cos(\omega t) \\ y_c &= L/2 + A\cos(\frac{\pi}{2} + 2\omega t) \\ z_c &= 0.66[\mathrm{m}] \\ \omega &= 2[\mathrm{s}^{-1}], A = 0.4\,[\mathrm{m}] \end{aligned} \tag{62}$$

This is similar to the case Section (5.4.1) but 3D. The large filamentary vortex layers are no more present, but instead there is a large amount of small eddies characteristic of a 3D flow.

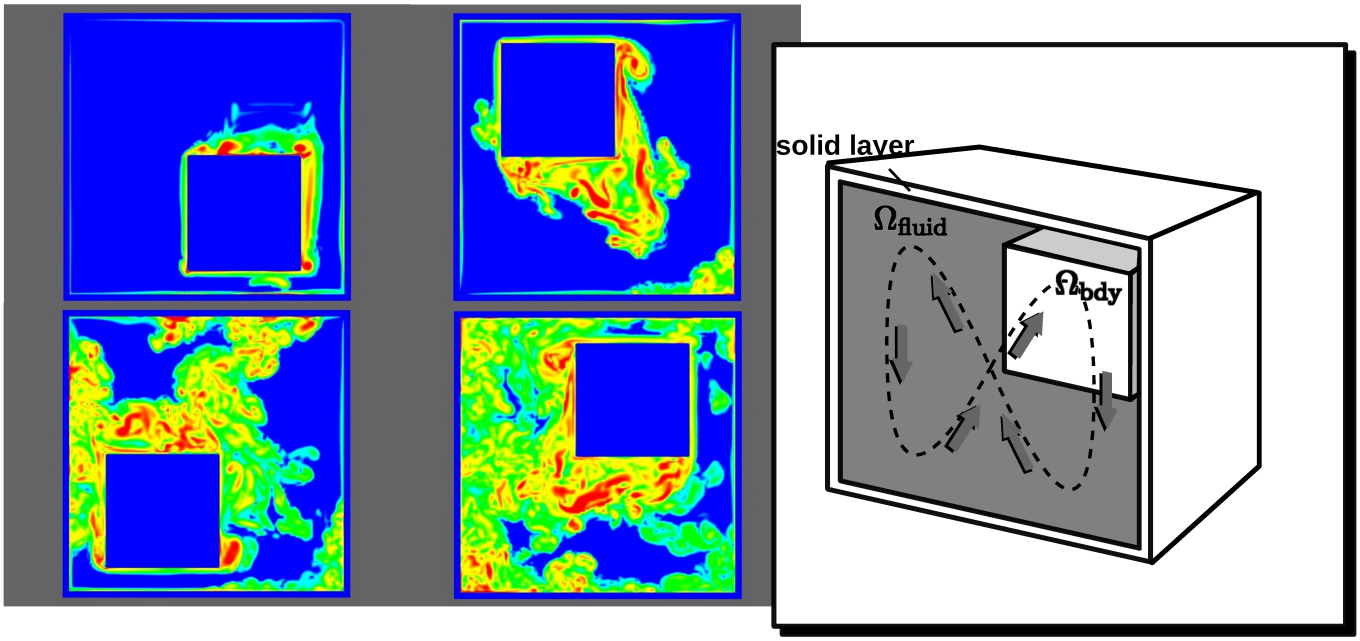


Figure 19: Colormap of $\log_{10}(|\omega|)$ for a cube moving in a Lissajous *8*-shaped curve.

5.4.5 Falling block

The body is a parallelepiped block of dimensions $L_x = L_z = 0.6\,[\text{m}]$, $L_y = 0.2\,[\text{m}]$. The center of the body is initially at $(x_c, y_c, z_c) = (0.4125, 0.95, 0.5)\,[\text{m}]$ and starts falling vertically with a velocity of $1\,[\text{m/s}]$. As the body falls it displaces a large quantity of fluid that forms a turbulent region expanding from both sides of the block.

6 Conclusions

We presented a new method called Accelerated Global Preconditioning for solving the incompressible Navier-Stokes equations with moving bodies. The algorithm is based on a pressure segregated, staggered grid, Finite Volume formulation and uses a FFT solver for preconditioning the CG solution of the Poisson problem. Theoretical estimates of the condition number of the preconditioned Poisson problem are given, and several numerical examples are presented validating these estimates. The algorithm is specially suited for implementation on GPU hardware. The condition number of the preconditioned Poisson equation does not degrade with refinement. The algorithm allows computing 3D problems in real time on moderately large meshes for many problems of practical interest in the area of Computational Fluid Dynamics.

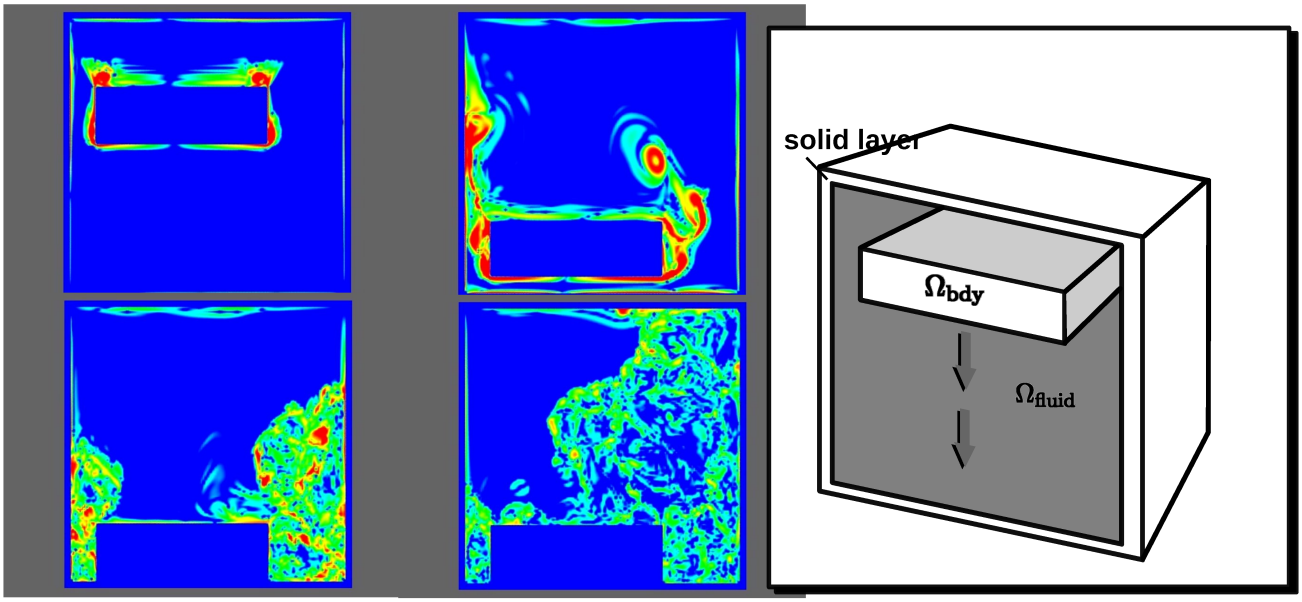


Figure 20: Colormap of $\log_{10}(|\omega|)$ for a falling block.

7 Acknowledgment

This work has received financial support from:

- Consejo Nacional de Investigaciones CientÃ■ficas y TÃl'cnicas (CONICET, Argentina, PIP 5271/05),
- Universidad Nacional del Litoral (UNL, Argentina, grant CAI+D 2009-65/334),
- Agencia Nacional de PromociÃşn CientÃ■fica y TecnolÃşgica (ANPCyT, Argentina, grants PICT-1506/2006, PICT-1141/2007, PICT-0270/2008), and
- European Research Council (ERC) Advanced Grant, Real Time Computational Mechanics Techniques for Multi-Fluid Problems (REALTIME, Reference: ERC-2009-AdG).

The authors made extensive use of *Free Software* as GNU/Linux OS, GCC/G++ compilers, Octave, and *Open Source* software as VTK among many others. In addition, many ideas from these packages have been inspiring to them.

8 References

[APB07] S. Adams, J. Payne, and R. Boppana. Finite difference time domain (FDTD) simulations using graphics processors. *HPCMP Users Group Conference*, 0:334–338, 2007.

[BG09] Nathan Bell and Michael Garland. Implementing sparse matrix-vector multiplication on throughput-oriented processors. In *SC '09: Proceedings of*

the Conference on High Performance Computing Networking, Storage and Analysis, pages 1–11, New York, NY, USA, 2009. ACM.

[CCLW11] A. Corrigan, F.F. Camelli, R. Löhner, and J. Wallin. Running unstructured grid-based CFD solvers on modern graphics hardware. *International Journal for Numerical Methods in Fluids*, 66(2):221–229, 2011.

[CLT07] K. Crane, I. Llamas, and S. Tariq. Real-time simulation and rendering of 3D fluids. *GPU Gems*, 3:633–675, 2007.

[ELD08] Erich Elsen, Patrick LeGresley, and Eric Darve. Large calculation of the flow over a hypersonic vehicle using a GPU. *J. Comput. Phys.*, 227(24):10148–10161, 2008.

[ETK+08] Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. Stable, circulation-preserving, simplicial fluids. In *SIGGRAPH Asia '08: ACM SIGGRAPH ASIA 2008 courses*, pages 1–11, New York, NY, USA, 2008. ACM.

[Far11] R. Farber. *CUDA application design and development*. Morgan Kaufmann, 2011.

[FJ98] M. Frigo and S.G. Johnson. FFTW: an adaptive software architecture for the FFT. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 3, pages 1381–1384. IEEE, 1998.

[FJ12] M. Frigo and S.G. Johnson. FFTW: fastest fourier transform in the west. In *Astrophysics Source Code Library, record ascl: 1201.015*, volume 1, page 01015, 2012.

[GSMY+08] Dominik Goddeke, Robert Strzodka, Jamaludin Mohd-Yusof, Patrick McCormick, Hilmar Wobker, Christian Becker, and Stefan Turek. Using GPU's to improve multigrid solver performance on a cluster. *Int. J. Comput. Sci. Eng.*, 4(1):36–55, 2008.

[IGLF06] Geoffrey Irving, Eran Guendelman, Frank Losasso, and Ronald Fedkiw. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Trans. Graph.*, 25(3):805–811, 2006.

[INLO12] S. Idelsohn, N. Nigro, A. Limache, and E. Oñate. Large time-step explicit integration method for solving problems with dominant convection. *Computer Methods in Applied Mechanics and Engineering*, 217-220:168–185, 2012.

[Kel95] C.T. Kelley. *Iterative methods for linear and nonlinear equations*. Society for Industrial Mathematics, 1995.

[KWBH09] A. Klöckner, T. Warburton, J. Bridge, and J.S. Hesthaven. Nodal discontinuous Galerkin methods on graphics processors. *Journal of Computational Physics*, 228(21):7863–7882, 2009.

[Leo79] BP Leonard. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Comput. Methods Appl. Mech. Eng.*, 19(1):59–98, 1979.

[LMU+09] Miguel Lastra, José M. Mantas, Carlos Urena, Manuel J. Castro, and José A. García-Rodríguez. Simulation of shallow-water systems using graphics processing units. *Math. Comput. Simul.*, 80(3):598–618, 2009.

[MCP+09] Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. Energy-preserving integrators for fluid animation. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–8, New York, NY, USA, 2009. ACM.

[MCPN08] Jeroen Molemaker, Jonathan M. Cohen, Sanjit Patel, and Jonyong Noh. Low viscosity flow simulations for animation. In *SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 9–18, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[MRDI12] F. Mossaiby, R. Rossi, P. Dadvand, and S. Idelsohn. OpenCL-based implementation of an unstructured edge-based finite element convection-diffusion solver on graphics hardware. *International Journal for Numerical Methods in Engineering*, 89:1635 1651, 2012.

[NBGS08] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming with cuda. *ACM Queue*, 6(2):40–53, 2008.

[Pes02] C.S. Peskin. The immersed boundary method. *Acta numerica*, 11(0):479–517, 2002.

[PNS06] R.R. Paz, N. Nigro, and M. Storti. On the efficiency and quality of numerical solutions in CFD problems using the Interface Strip Preconditioner for domain decomposition. *International Journal for Numerical Methods in Fluids*, 52:89–118, 2006.

[PS05] R.R. Paz and M. Storti. An Interface Strip Preconditioner for Domain Decomposition Methods Application to Hydrology. *International Journal for Numerical Methods in Engineering*, 62(13):1873–1894, 2005.

[RGBVC08] P. Rinaldi, C. García Bauza, M. Vénere, and A. Clausse. Paralelización de autómatas celulares de aguas superficiales sobre placas gráficas. In Alberto Cardona, Mario Storti, and Carlos Zuppa, editors, *Mecánica Computacional Vol. XXVII*, volume XXVII, pages 2943–2957, 2008.

[RRB+08] S. Ryoo, C.I. Rodrigues, S.S. Baghsorkhi, S.S. Stone, D.B. Kirk, and W.W. Hwu. Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, pages 73–82. ACM, 2008.

[SDP+06] M. Storti, L. Dalcín, R.R. Paz, A. Yommi, V. Sonzogni, and N. Nigro. A Preconditioner for the Schur Complement Matrix. *Advances in Engineering Software*, 37:754–762, 2006.

[TS09] Julien C Thibault and Inanc Senocak. CUDA implementation of a Navier-Stokes solver on multi-GPU desktop platforms for incompressible flows. In AIAA, editor, *47th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition (Disc 1)*, pages 1–15, 2009.

[WLL04] Enhua Wu, Youquan Liu, and Xuehui Liu. An improved study of real-time fluid simulation on GPU: Research articles. *Comput. Animat. Virtual Worlds*, 15(3-4):139–146, 2004.

[WWZ12] Xingshi Wang, Chu Wang, and Lucy Zhang. Semi-implicit formulation of the immersed finite element method. *Computational Mechanics*, 49:421–430, 2012.

Título:

GPGPU implementation of the BFECC algorithm for pure advection equations

Contribución:

Escritura del artículo, desarrollo e implementación GPGPU del método numérico

Firma del director de tesis:

GPGPU implementation of the BFECC algorithm for pure advection equations

Santiago D. Costarelli[1], Mario A. Storti[1], Rodrigo R. Paz[1],
Lisandro D. Dalcin[1], Sergio R. Idelsohn[1,2,3]

[1]*CIMEC, INTEC - Universidad Nacional del Litoral y CONICET, Santa Fe, Argentina.*

[2]*Institució Catalana de Recerca i Estudis Avançats (ICREA), Barcelona, Spain*

[3]*International Center for Numerical Methods in Engineering (CIMNE), Technical University of Catalonia (UPC), Gran Capitán s/n, 08034 Barcelona, Spain*

Abstract

In the present work an implementation of the Back and Forth Error Compensation and Correction (BFECC) algorithm specially suited for running on General-Purpose Graphics Processing Units (GPGPUs) through Nvidia's Compute Unified Device Architecture (CUDA) is analyzed in order to solve transient pure advection equations. The objective is to compare it to a previous explicit version used in a Navier-Stokes solver fully written in CUDA. It turns out that BFECC could be implemented with unconditional stable stability using Semi-Lagrangian time integration allowing larger time steps than Eulerian ones.

Keywords: GPGPU; CUDA; BFECC; Semi-Lagrangian; Level-Set; Navier-Stokes

Contents

1 Introduction

In recent years GPGPU's (General-Purpose Graphics Processing Units) are being used in HPC (High Performance Computing), specially for problems that can be solved with Cellular Automata (CA) algorithms. In particular, a great effort is being oriented to exploit the great computing power of this hardware on fluid mechanics problems.

In previous works [SPD+13, MCPN08, CPDS11] a Fractional Step solver for the Navier-Stokes (NS) equations was presented. All steps in the solver were explicit, except for the projection step, which includes the solution of a Poisson equation, which is performed with a Fast Fourier Transform (FFT) technique. Explicit methods fall in the category of CA algorithms and then can be implemented very efficiently on GPGPUs. The momentum equations were solved using a stabilization technique called Quadratic Upwinded Interpolation for Convection Kinematics (QUICK) [Leo79]. The performance of the solver was limited because QUICK performs the computation using a large stencil and is limited by the CFL (Courant-Friedrichs-Lewy) condition. In this article an alternative algorithm for the solution of the momentum equations based on Back and Forth Error Compensation and Correction (BFECC, [SC91, CSP+13]) is explored. This solver is more efficient since allows for larger CFL numbers; its stability is restricted only by the diffusion term. The algorithm was implemented on Nvidia's GPGPUs using the Compute Unified Device Architecture (CUDA)[NBGS08, Far11].

This article is an extended version of the presentation *"A Numerical Algorithm for the Solution of Viscous Incompressible Flow on GPU's"* presented at HPCLatAm 2012 [CSP+12].

2 Governing equations

The equations being solved are the classical NS equations for incompressible viscous fluid flows:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} &= -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f} \\ \nabla \cdot \mathbf{u} &= 0 \end{aligned} \tag{1}$$

where $\mathbf{u}$ is the velocity field, p the pressure field, ρ the density (constant), ν the kinematic viscosity (constant) and $\mathbf{f}$ a body force per unit volume. These equations are intended to be solved with several combination of boundary and initial conditions.

Considering $\mathbf{w}_0$ as an approximation to the solution of $\mathbf{u}$ at time $t = n\Delta t$, one can obtain the solution at $t + \Delta t$ performing successively the following operations [Sta99]:

- *Force:* Add force terms:

$$\mathbf{w}_1(\mathbf{x}, t) = \mathbf{w}_0(\mathbf{x}, t) + \Delta t\ \mathbf{f}(\mathbf{x}, t) \tag{2}$$

- *Advection*: the BFECC method following the characteristic field is used. This results in an unconditionally stable step. This is going to be explained on the following sections.

- *Diffusion*: The diffusion equation:

$$\frac{\partial \mathbf{w}}{\partial t} = \nu \nabla^2 \mathbf{w} \tag{3}$$

  is solved in the interval $[t, t + \Delta t]$, with $\mathbf{w}(t) = \mathbf{w}_2$, $\mathbf{w}(t + \Delta t) = \mathbf{w}_3$. If periodic boundary conditions are applied, this step can be solved implicitly with an efficient Fast Fourier Transform (FFT) based solver, avoiding the restriction on the Fourier number:

$$\mathrm{Fo} = \nu \Delta t / h^2 < \mathrm{Fo}_{\mathrm{crit}} \tag{4}$$

  typical for explicit methods. $\mathrm{Fo}_{\mathrm{crit}}$ is a nondimensional constant depending on the spatial dimension and discretization stencil. For the standard second order Poisson discretization used in this paper $\mathrm{Fo}_{\mathrm{crit}} = 1/(2n_d)$, where n_d is the number of spatial dimensions.

- *Projection:* the resulting velocity field is projected onto the divergence free space. This is achieved solving a Poisson equation:

$$\mathbf{w}_4(\mathbf{x}, t) = \mathbf{w}_3(\mathbf{x}, t) - \nabla p \tag{5}$$

  where p is defined as the solution of [CMM90]:

$$\begin{aligned} \nabla^2 p &= \nabla \cdot \mathbf{w}_3 \ \text{ in } \Omega \\ \nabla p \cdot \mathbf{n} &= \mathbf{w}_3 \cdot \mathbf{n} \ \text{ in } \partial\Omega \end{aligned} \tag{6}$$

  $\Omega \subset \mathbb{R}^n$ being the domain on a n-dimensional space, $\partial\Omega$ its boundary and $\mathbf{n}$ the outward normal to ∂D.

2.1 The method of characteristics

The scope of this article is to focus on the solution of the advection step of the exposed solver. Let's consider for the moment a scalar field F that is being advected by the velocity

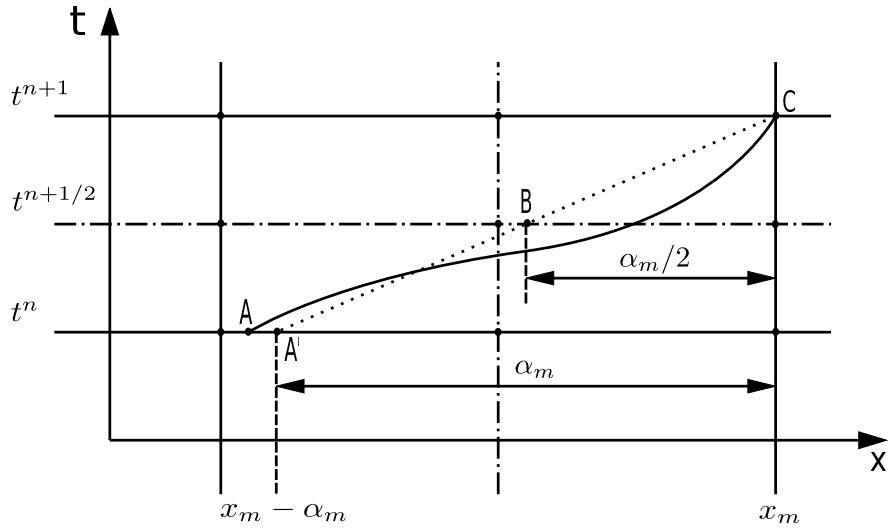


Figure 1: Solid line is a trajectory that connects A and C points. Dashed line $\overline{A'C}$ is an approximation based on using the velocity field as a first order predictor.

field $\mathbf{u}$:

$$\frac{D_{(m)}F}{Dt} = \frac{\partial F}{\partial t} + \mathbf{u} \cdot \nabla F = 0 \tag{7}$$

where $D_{(m)}(\cdot)/Dt$ stands for a material derivative, i.e. following fluid particles. An arbitratry dimensional space can be considered, in 3D $\mathbf{x} = (x, y, z)$. Even if the method of characteristics can be explained abstractly (without spatial discretization), to fix ideas a simple cartesian grid with constant mesh size h is going to be used.

The approach for solving this kind of equations follows that presented on [SC91]. Considering Equation (7) in 1D (extensions to higher dimensions are trivial), a splitting reveals that:

$$\frac{\partial F}{\partial t} + \frac{dx}{dt}\frac{\partial F}{\partial x} = 0 \tag{8}$$

where:

$$\frac{dx}{dt} = u(x, t) \tag{9}$$

that can be solved leading to an ordinary differential equation for the particle position x. Once the trajectories of the particles have been determined, the solution of Equation (8) is obtained by simply stating that the value of F is constant throughout the trajectory [BC92]. This last equation relates, in particular, points A and C of Figure (1). The objective is to track back the trajectory passing through C at time $t + \Delta t$ to the point defined as A at time t. This should be done following the trajectory (solid curve); but in the discrete case this is approximated using the velocity field as a first order predictor of the previous position (dashed curve); reaching point A'.

Components of the velocity field on positions located at the mid interval of the actual computational grid are required (i.e. $\mathbf{u}(\mathbf{x}, t^{n+\frac{1}{2}})$). One option is a second order extrapolation of the form:

$$\mathbf{u}(\mathbf{x}, t^{n+\frac{1}{2}}) = \frac{1}{2}\left(3\mathbf{u}(\mathbf{x}, t^{n}) - \mathbf{u}(\mathbf{x}, t^{n-1})\right) \tag{10}$$

Then, Equation (9) can be integrated to $O(\Delta t^2)$ [Rob81] solving for $\boldsymbol{\alpha} = \Delta\mathbf{x}$ such that:

$$\frac{\Delta\mathbf{x}}{\Delta t} = \frac{\boldsymbol{\alpha}}{\Delta t} = \mathbf{u}(\mathbf{x} - \frac{\boldsymbol{\alpha}}{2}, t^{n+\frac{1}{2}}) \tag{11}$$

whose solution can be obtained by a Fixed-Point iteration:

$$\boldsymbol{\alpha}^{(k+1)} = \Delta t\ \mathbf{u}(\mathbf{x} - \frac{\boldsymbol{\alpha}^{(k)}}{2}, t^{n+\frac{1}{2}}) \tag{12}$$

taking $\boldsymbol{\alpha}^0 = \mathbf{0}$ on the first iteration $k = 0$. It was found that using two iterations was typically sufficient; this coincides with the results reported by [SC91].

A sufficient condition for convergence of the iterative procedure is shown in [PBS85], and basically states that Δt must be smaller than the reciprocal of the maximum absolute value of the velocity gradient components, i.e. for the 2D case :

$$\Delta t < \left[\max\left(|u_{,x}|, |u_{,y}|, |v_{,x}|, |v_{,y}|\right)\right]^{-1} \tag{13}$$

where u and v are the velocity components on x and y direction respectively). Having the value of $\boldsymbol{\alpha}$, Equation (7) is easily solved by stating that the value of F is constant along the approximated trajectory $\overline{A'C}$, i.e.:

$$\frac{F(\mathbf{x}, t^{n+1}) - F(\mathbf{x} - \boldsymbol{\alpha}, t^{n})}{\Delta t} = 0 \tag{14}$$

Trilinear (in 3D) spatial interpolation is used when evaluating $\boldsymbol{\alpha}$ and when solving Equation (14). Linear interpolation for $\boldsymbol{\alpha}$ and tricubic spline interpolation for F was proposed in [SC91] in order to reduce numerical damping. Tricubic spline interpolation has also the property of exactly conserving mass for solenoidal flows. Analysis of the stability properties of the semi-Lagrangian advection scheme [BM82] shows that it is possible to integrate Equation (14) for CFL numbers greater than one; where $\text{CFL} = u_{\max}\Delta t/h$, and $u_{\max}$ being the maximum velocity in the flow field.

2.2 Back and Forth Error Compensation and Correction

Following [KLLR07] let's suppose that there exists some operator $L(.,.)$ that performs an approximate solution to the advection operator, i.e.:

$$F^{n+1} = L(\mathbf{u}, F^{n}) \tag{15}$$

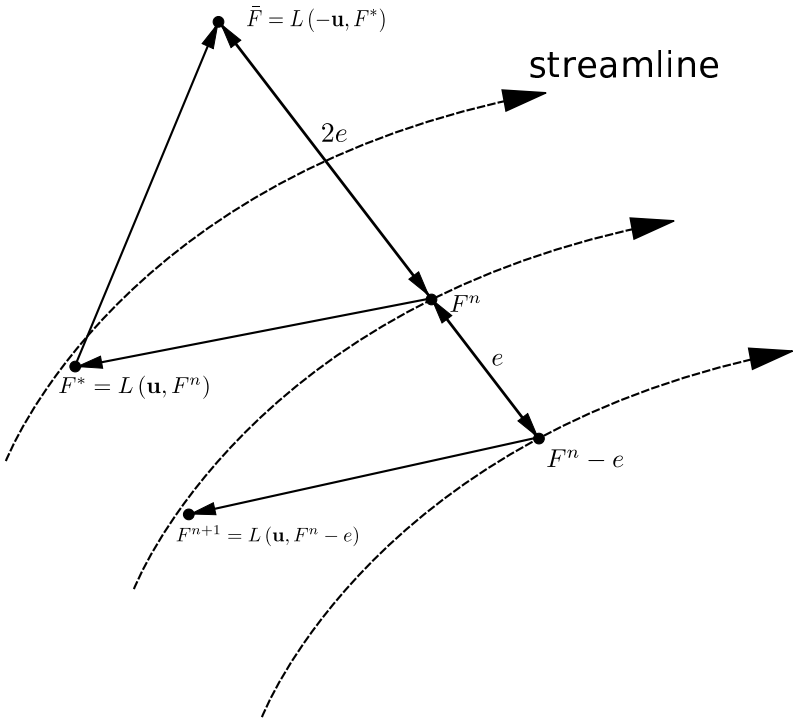


Figure 2: Schematic BFECC operation over a streamline field and using $L(.,.)$ as the advection operator for the scalar field F.

In the continuum case a forward operation $L(-\mathbf{u}, \cdot)$ exactly compensates the backward operation $L(\mathbf{u}, \cdot)$ (see Figure (2)), i.e. after applying a backward and forward operation the same initial field is obtained. (Note that this is valid only for the pure advection case, with no diffusion). Of course, this is no longer true for the discrete operators due to numerical errors. Lets say that the backward operation introduces an error denoted as e. If this error is purely dissipative, then the forward operation will introduce the same error, so after the backward and forward steps a field $\bar{F} = F^n + 2e$. Then, an explicit expression for e can be readily obtained as:

$$e = -\frac{1}{2}\left(F^n - \bar{F}\right) \tag{16}$$

This error can be subtracted (this is called as *error compensation*, hence the name of the method) from F^n and then the field is advected. This method has been proven to be second order accurate in both, time and space [SFK+08].

Considering the advection operator $L(.,.)$ as the Semi-Lagrangian one explained on Section (2.1), BFECC is defined as follows:

$$F^* = L\left(\mathbf{u}, F^n\right)$$

$$\bar{F} = L\left(-\mathbf{u}, F^*\right)$$

$$F^* = F^n + \left(F^n - \bar{F}\right)/2$$

$$F^{n+1} = L\left(\mathbf{u}, F^{*}\right).$$

In this way the order of accuracy of the Semi-Lagrangian scheme can be raised from one to two at the expense of computing three times the advection operator [SFK+08].

As there is no strong evidence on the time dependence of e a variation of the algorithm was proposed in [SFK+08]. After the first two steps, forward and backward, the error is computed and used as a correction directly for the backward solution $L\left(\mathbf{u}, F^{n}\right)$ instead of applying a new forward step with a modified field. In this article the standard BFECC is used, since the modified version introduced previously performed worst (in terms of accuracy) in the cases studied.

The proposed scheme does not preserve positivity, i.e. it does not satisfy the Discrete Maximum Principle [SFK+08], i.e. overshoots/undershoots may be observed at sharp discontinuities. A basic correction was proposed in the mentioned reference and tested on the transport of a 1D square signal. This correction was implemented by the authors, but the improvement obtained was not worth for the extra computational effort needed.

3 Numerical examples: scalar transport

In typical solid-fluid applications F is used for representing the region occupied by each phase, the fluid occupying the region Ω where $F \geq 0$ and the other the complement.

3.1 Unbiased Level-Set contouring algorithm (ULCA)

First of all, given an initial region Ω the level set function F must be computed. The algorithm used in this article follows that presented on [CFA01]. It turns out that even with coarse grids (i.e. 50×50) the numerical error obtained by the algorithm developed differs at most by 1%.

3.2 Error indicator based on the L1-norm computation of the characteristic function

One way to assess the quality of the discrete transport algorithm is to compute the area (area in 2D, volume in 3D) occupied by one fluid, i.e. the $F \geq 0$ region, for instance, since for a solenoidal field this area should be preserved. So the error indicator would be:

$$E_a = ||\Omega^n| - |\Omega^0|| \tag{17}$$

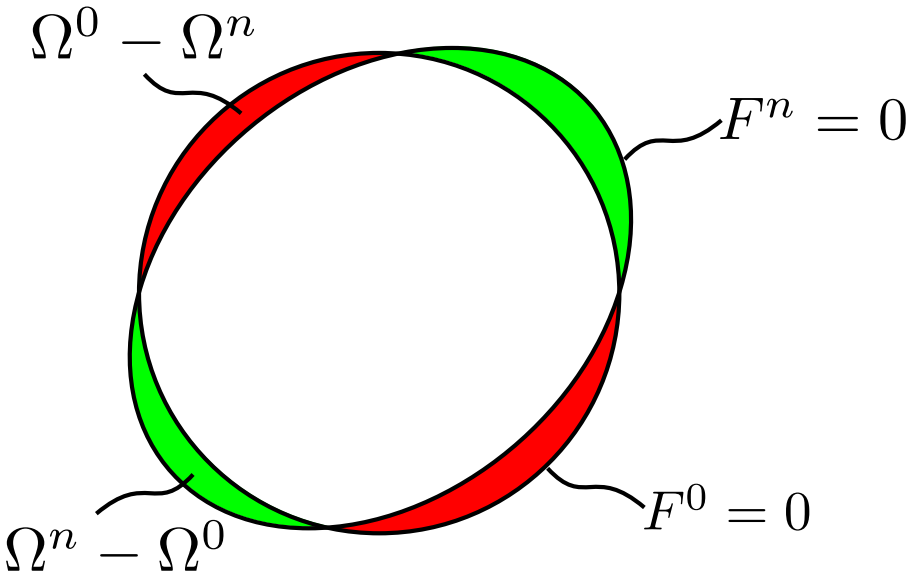


Figure 3: Definition of the L_1 error E in the level set indicators.

In this equation the absolute value $|\cdot|$ for sets stands for the area (measure) of the set. Ω^0 (Ω^n) is the region where the initial (final) level set function is nonnegative $F^0 \geq 0$ ($F^n \geq 0$).

But checking only area changes may be misleading, because the transport algorithm may distort the region keeping the same area. If the velocity field is a pure rotation, then after a full rotation the geometry F^n should be the same as the initial F^0. Then the improved error indicator E (see [SF99]) consists basically in measuring the area where the indicator functions $H(F)$ for the initial and final configuration do not coincide, i.e. the sum of the red and green areas in Figure (3):

$$\begin{aligned} E &= \int_\Omega \frac{1}{L} \left| H(F^0) - H(F^n) \right| d\Omega \\ &= |\Omega^n - \Omega^0| + |\Omega^0 - \Omega^n| \end{aligned} \tag{18}$$

where the Heaviside function $H(.)$ is defined as $H(\phi) = 1$ if $\phi \leq 0$ and $H(\phi) = 0$ otherwise. Note that in the second line of Equation (18) the subtraction operator stands for set difference. This E indicator is null only if the regions are coincident. Note that if the absolute value is removed in the definition of E with the H differences then we would get the area difference, i.e.:

$$\begin{aligned} E_a &= ||\Omega^n| - |\Omega^0|| \\ &= \left| \int_\Omega (H(F^0) - H(F^n)) d\Omega \right| \end{aligned} \tag{19}$$

The error indicator E can be computed numerically by subintegration, i.e.:

- Refine the computational grid into smaller subcells, i.e. each cell is divided in $n_{\text{sc}} \times n_{\text{sc}}$ subcells (typically n_{sc} =10).

- Interpolate bilinearly the values of F^0 and F^n over this finer mesh.
- Compute the L_1-norm as:

$$\begin{aligned} E &= \int_\Omega \frac{1}{L} \left| H(F^0) - H(F^n) \right| d\Omega \\ &\approx \frac{\Omega_c}{L} \sum_{i,j=1}^{N} \left| H(F_{ij}^0) - H(F_{ij}^n) \right| \end{aligned} \tag{20}$$

  where L is the expected perimeter, N is the amount of subcells per dimension, and Ω_c is the subcell area.

For velocity fields that are not rigid body rotations, the benchmark can be extended in the following way. The initial level set function F^0 is advected under the velocity field $\mathbf{u}$ for a certain time $T/2$ to a deformed state $F^{n/2}$. Then this distorted geometry is advected back with the velocity field reversed ($\mathbf{u} \to -\mathbf{u}$). Again, the same region should be recovered and the error indicator E can be used.

3.3 Performance measurement

The efficiency of the algorithm is assessed by computing the rate at which cells are processed, using the following data:

- N_{it}: number of iterations performed
- N_{cells}: number of cells computed on each time step
- t_{total}: time (in seconds) taken to perform one complete BFECC iteration

with $N_{cells} = (N_{dim})^d$, d being the number of dimensions and N_{dim} the amount of grid points per direction (only structured cartesian constant-mesh-size grids are considered). So the computing rate is computed as:

$$\text{Rate} = \frac{N_{it} \times N_{cells}}{t_{total}} \left[\frac{\text{cells}}{\text{sec}} \right] \tag{21}$$

3.4 2D examples

3.4.1 Zalesak's disk

This test consists in the advection of a region composed of a circle with a slot [Zal79].

- The computational domain is $\Omega \subset \mathbb{R}^2 : [0;100] \times [0;100]$;

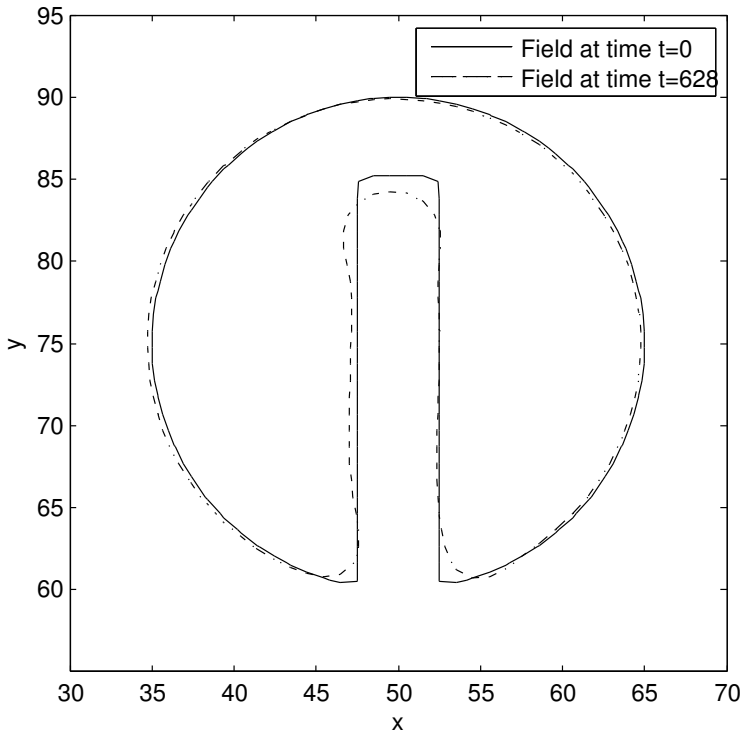


Figure 4: Zalesak's disk results after one full revolution with 100 grid point per direction and $CFL = 4.9$.

- The advected region is a circle centered at $(50; 75)$ with a radius of 15 and a slot of width 5 and height 25.
- The velocity field is a rigid body rotation around the center of the domain with a period of 628 time units:

$$\begin{aligned} u &= (\pi/314)(50 - y) \\ v &= (\pi/314)(x - 50) \end{aligned} \tag{22}$$

The initial field and the error indicator (see Section (3.2)) after one revolution are shown on Figure (4) and Table (1). The CFL number was 4.9. For reference the results obtained with a particle level set method [ELF05] are shown.

The results obtained by BFECC are similar in accuracy to those obtained by the particle method, but far more accurate than those obtained by other simple level set methods (see [ELF05]).

	Grid Cells	Initial Area	Area Loss [%]	L_1-error
	Theoretical	581.85	-	-
One revolution	50	574.38	4.98 (3.09)	0.85 (0.434)
	100	581.14	0.78 (1.07)	0.26 (0.181)
	200	580.92	0.11 (0.22)	0.06 (0.105)
Two revolutions	50	574.38	4.66 (2.66)	1.75 (0.610)
	100	581.14	4.02 (1.01)	0.52 (0.206)
	200	580.92	0.34 (0.22)	0.12 (0.103)

Table 1: Zalesak's disk. Error indicator values obtained with the present BFECC code. In parentheses numerical results obtained by [ELF05] using Particle Level-Set method and Semi-Lagrangian time integration.

3.4.2 Single vortex

While Zalesak's disk test is a good indicator of numerical diffusion in an interface-capturing method, it does not test the ability to preserve small scale properties of the fluid flow. The single vortex benchmark proposed in [PAB+97] tests this property of tearing flows.

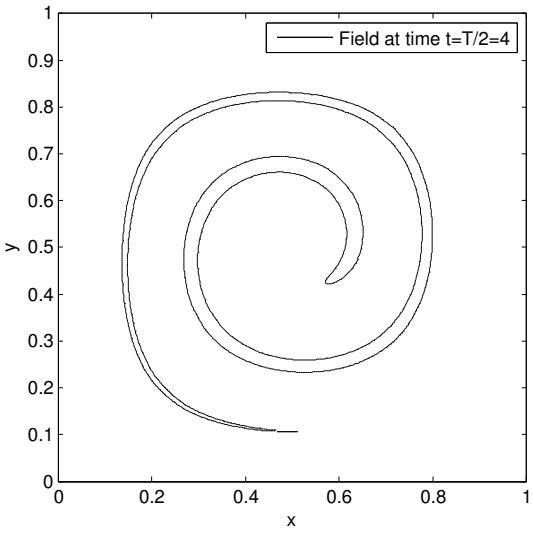
The velocity field is defined by the stream function:

$$\psi(\mathbf{x}) = \frac{1}{\pi}\sin^2(\pi x)\sin^2(\pi y)\cos\left(\frac{\pi t}{T}\right) \tag{23}$$

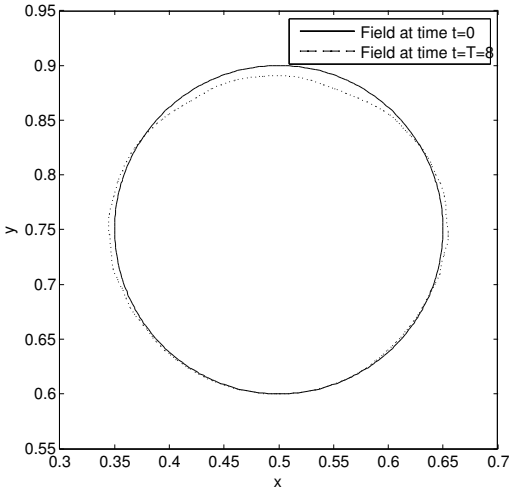
so, the velocity components are:

$$\begin{aligned} u &= \psi_{,x} = \sin^2(\pi x)\sin(2\pi y)\cos\left(\frac{\pi t}{T}\right) \\ v &= -\psi_{,y} = -\sin^2(\pi y)\sin(2\pi x)\cos\left(\frac{\pi t}{T}\right) \end{aligned} \tag{24}$$

The test considers a unit square computational domain with a circle of radius 0.15 placed at $(0.5; 0.75)$. The initial conditions, the maximum stretching and the results after the forward and backward steps with $T = 8$, $N = 256$ grid points per dimension and $CFL = 4.9$ are shown on Figure (5). The error indicator (see Section (3.2)) is shown in Table (2).



(a) Field at time t = T/2 = 4.



(b) Field at time t = T = 8.

Figure 5: Single vortex test using 256 grid points per direction and $CFL = 4.9$.

3.5 3D examples

3.5.1 Volume preservation computation

In this test the volume preservation of the scheme [EFFM02] is checked. The volume is computed as:

$$\int_{\Omega} H(\phi) d\mathbf{x} = h^3 \sum_{i,j=1}^{N} \tilde{H}\left(\phi_{i,j}\right) \tag{25}$$

Grid Cells	Initial Area	Area Loss [%]	L1-error
Theoretical	0.0707	-	-
64	0.0443	37.09 (1.81)	0.033 (0.00300)
128	0.0652	7.66 (0.71)	0.014 (0.00100)
256	0.0696	1.44 (0.35)	0.003 (0.00059)

Table 2: Single vortex test. Area loss and error indicator values obtained with the present BFECC code. In parentheses numerical results obtained using Particle Level-Set method and Semi-Lagrangian time integration [EFFM02].

where $\tilde{H}$ is a regularized version of the Heaviside function:

$$\tilde{H}(x) = \begin{cases} 0, & \text{if } x < -\epsilon \\ \frac{1}{2} + \frac{x}{2\epsilon} + \frac{1}{2\pi} sin\left(\frac{\pi x}{\epsilon}\right), & \text{if } -\epsilon \leq x \leq \epsilon \\ 1, & \text{if } x > \epsilon \end{cases} \tag{26}$$

and $\epsilon = 1.5h$.

3.5.2 Zalesak's Disk

This benchmark is the natural extension of the 2D version (see Section (3.4.1)) to 3D. The domain $\Omega \subset \mathbb{R}^3 : [0;100] \times [0;100] \times [0;100]$ and 100 grid points per direction. The disk is now a sphere centered at $(50;75;50)$ with radius 15 and a slot of width 5 and height 25. The velocity field is rigid body rotation centered at $z = 50$, is

$$\begin{aligned} u &= (\pi/314)(50 - y), \\ v &= (\pi/314)(x - 50), \\ w &= 0. \end{aligned} \tag{27}$$

The level set after one complete revolution is shown in Figure (6). The numerical results obtained are shown on Table (3) being the accuracy comparable to that achieved with particle methods.

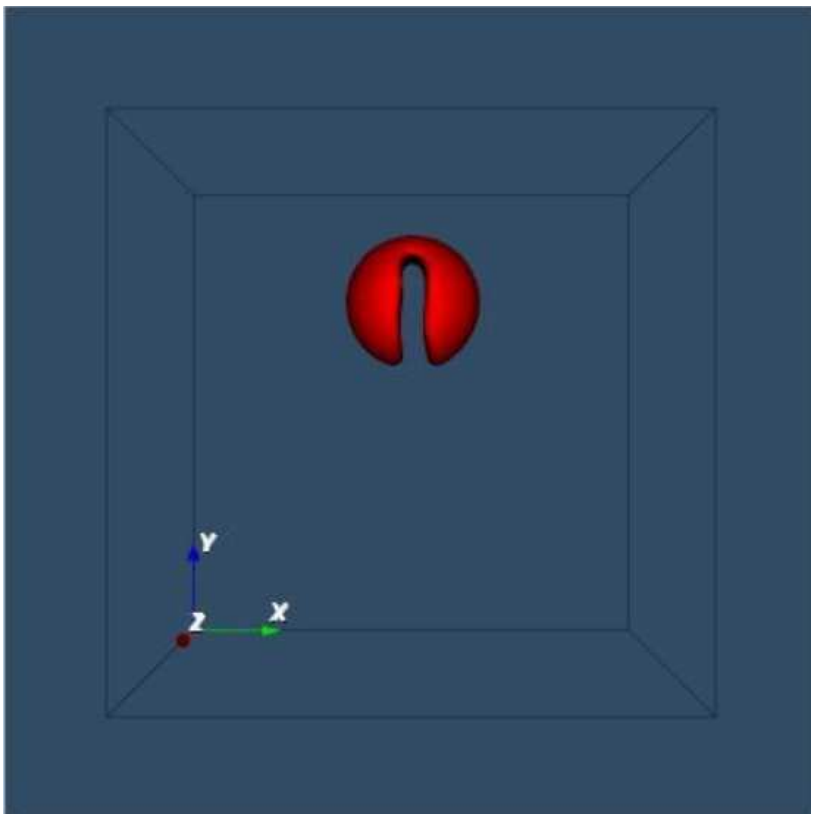


Figure 6: 3D Zalesak's disk after one complete revolution with 100 points per dimension and $CFL = 4.9$.

3.5.3 Deformation field

First proposed by LeVeque [LeV96] this is a 3D test combining deformations on x-y and x-z planes. The velocity field is given by:

$$\begin{aligned} u &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos\left(\frac{\pi t}{T}\right) \\ v &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi x)\cos\left(\frac{\pi t}{T}\right) \\ w &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos\left(\frac{\pi t}{T}\right) \end{aligned} \tag{28}$$

modulated in time, being the half period $T = 3$. The domain is a unit cube and the advected region is initially a sphere of radius 0.15 placed at $(0.35; 0.35; 0.35)$. Since the velocity field is reversed for $t > T/2$ the geometry at the final position $T = 3$ should be the same as for the initial position $t = 0$. The numerical results are shown in Table (4). Figure (7) shows the field at maximum stretching $t = T/2 = 1.5$, and the final position at time $t = T = 3$. This test is particularly difficult due to small scale structures that can not be resolved, leaving a trailing tail behind the sphere not present in the initial condition. However the results obtained with the present algorithm are much better than those reported with comparable

Cells per dimension	Initial Volume	Volume Loss [%]
100	11176	0.51 (2.3 [EFFM02])
128	11106	0.27 (0.9 [MDHZ08])

Table 3: Volume loss for the Zalesak's disk problem with the present method. In parentheses results obtained with Particle Level-Set method [EFFM02] and Semi-Lagrangian time integration [MDHZ08].

algorithms [EFFM02, MDHZ08].

Cells per dimension	Initial Volume	Volume Loss [%]
100	0.014237	3.76 (2.6 [EFFM02])
128	0.014197	4.49 (1.8 [MDHZ08])

Table 4: Volume loss for the 3D deformation field case with the proposed BFECC algorithm. In parentheses values obtained with Particle Level-Set [EFFM02] and Semi-Lagrangian time integration [MDHZ08].

4 Applications to Navier-Stokes Equations

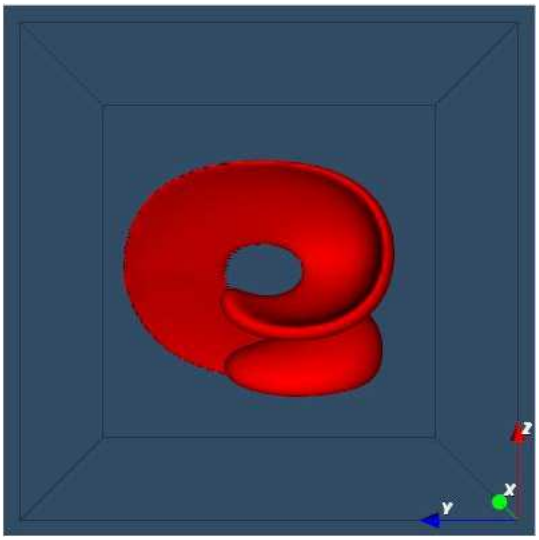
4.1 Solving Momentum equations by BFECC and Semi-Lagrangian Time Integration

The extension of the application of the method of characteristics with BFECC described in Section (2.1) to the advection term in the NS equations will be described. The material derivative is used to approximate the advection term as follows:

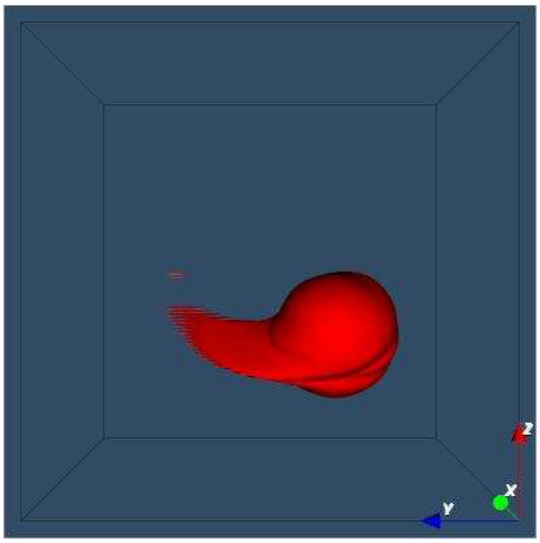
$$\frac{D_{(m)}\mathbf{u}\left(\mathbf{x};t\right)}{Dt} = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}^{*} \cdot \nabla \mathbf{u} = \mathbf{0} \tag{29}$$

where $\mathbf{u} = \{u, v, w\}$ are the velocity components on x, y and z directions. $\mathbf{u}^{*}$ is an intermediate (predicted) velocity field.

Staggered grids are used to prevent uncoupling of the velocity and pressure fields [MCPN08]. This must be taken into account in the application of the method of characteristics, as explained in Section (2.1), since velocities are now located in non coincident grids. Considering for instance pure transport of the x-velocity vector alone, the



(a) Field at time t = T/2 = 1.5.



(b) Field at time t = T = 3.

Figure 7: 3D deformation field test using 128 points per direction and $CFL = 4.9$.

discrete equation is:

$$\left(\frac{D_{(m)}u\left(\mathbf{x};t\right)}{Dt}\right)_{i+1/2,j,k} = 0 \tag{30}$$

then:

$$\boldsymbol{\alpha}^{(l+1)} = \Delta t\ \mathbf{u}^{*}\left(\mathbf{x} - \boldsymbol{\alpha}^{(l)}/2; t + \Delta t\right)_{i+1/2,j,k} \tag{31}$$

where l is the iteration index in the fixed point algorithm.

The explicit expressions of each component of $\boldsymbol{\alpha}$ are as follows:

$$\alpha_x^{(l+1)} = \Delta t \; u^*_{i+1/2-\alpha_x^{(l)}/2,j-\alpha_y^{(l)}/2,k-\alpha_z^{(l)}/2} \tag{32}$$

$$\alpha_y^{(l+1)} = \Delta t \; (v^*)^c_{i+1/2-\alpha_x^{(l)}/2,j-\alpha_y^{(l)}/2,k-\alpha_z^{(l)}/2} \tag{33}$$

$$\alpha_z^{(l+1)} = \Delta t \; (w^*)^c_{i+1/2-\alpha_x^{(l)}/2,j-\alpha_y^{(l)}/2,k-\alpha_z^{(l)}/2} \tag{34}$$

where $()^c$ stands for *co located* fields, i.e.:

$$(v^*)^c_{i,j+1/2,k} = \frac{\Delta t}{4}\{v^*_{i+1-\alpha_x^{(l)}/2,j+1/2-\alpha_y^{(l)}/2,k-\alpha_z^{(l)}/2} + \tag{35}$$

$$v^*_{i-\alpha_x^{(l)}/2,j+1/2-\alpha_y^{(l)}/2,k+1-\alpha_z^{(l)}/2} + \tag{36}$$

$$v^*_{i+1-\alpha_x^{(l)}/2,j-1/2-\alpha_y^{(l)}/2,k-\alpha_z^{(l)}/2} + \tag{37}$$

$$v^*_{i-\alpha_x^{(l)}/2,j-1/2-\alpha_y^{(l)}/2,k-\alpha_z^{(l)}/2}\}. \tag{38}$$

Finally, after convergence of the fixed point iteration ($\boldsymbol{\alpha}^{(l)} \to \boldsymbol{\alpha}$):

$$u^{n+1}_{i+1/2,j,k} = u^n_{i+1/2-\alpha_x/2,j-\alpha_y/2,k-\alpha_z/2} \tag{39}$$

The same scheme is applied for the transport of the other velocity components v and w.

4.2 Poisson equation for pressure

The projection step (as described in Section (2)) is solved using the Conjugate Gradient (CG) iterative solver with a FFT preconditioning. A complete analysis in this subject can be found in [SPD+13]. Of course, the number of iterations is a key point for the performance of the algorithm. The values reported in the numerical simulations have been obtained with tolerance ($O(10^{-3})$) which requires typically 3 iterations. The FFT is computed with the CUFFT implementation that is included in CUDA [Nvi10].

4.3 Numerical Examples

4.3.1 2D lid-driven cavity

This is a classical internal flow test in a square domain of side length L. The shear velocity $u = 1$, $v = 0$ is imposed at the top, whereas all the other sides are imposed the non-slip condition ($u = v = 0$). The numerical results obtained at Re $= 1000$ (taking L as characteristic length) are shown on Figure (8), compared with the results of Ghia et.al. [GGS82]. The performance obtained, measured in [Mcells/s] (i.e. seconds of

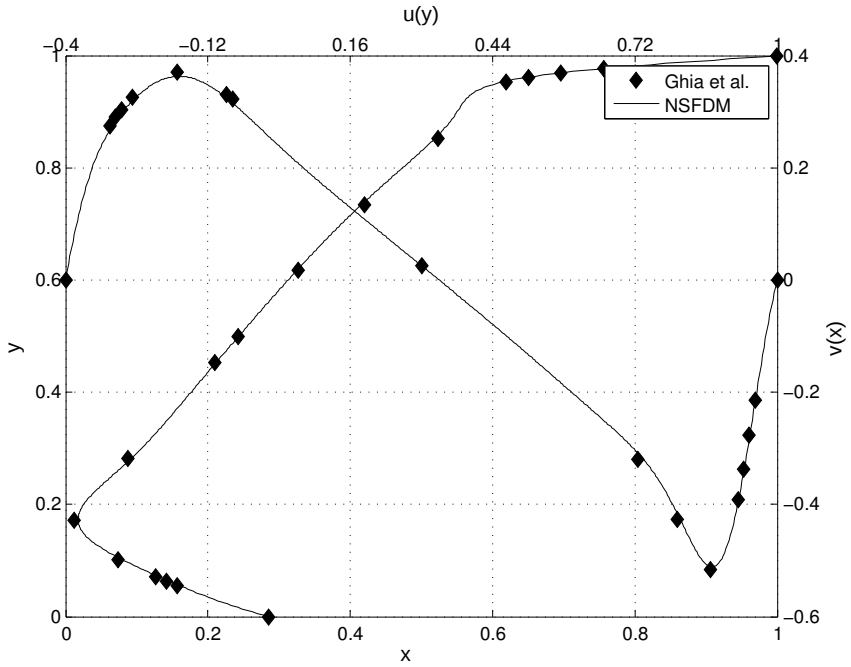


Figure 8: Lid driven square cavity. Results obtained at Re = 1000 using a grid of 512×512.

	Simple	Double
64×64	0.62	0.61
128×128	2.50	2.22
256×256	9.09	7.14
512×512	25.00	16.66

Table 5: Lid driven square cavity at Re = 1000. Performance measured in [Mcells/sec] on a NVIDIA GTX 580.

computation to compute one million cells), is shown on Table (5). In this case stability is diffusion-controlled (see Equation (4)), so the CFL is effectively reduced to CFL < 0.48. This is not reflected in the rates of the tables because they are expressed in cells/sec, but it reduces the efficiency of the code, since the rate of computation is proportional to CFL.

4.3.2 2D flow past circular cylinder

This is a homogeneous stream of unperturbed velocity u_∞ impinging on a cylinder of diameter D (see [Ach72, Ros54, MK01, RIMNMG+13]). The length L and height H of the computational domain are related $L = H = 15D$. This relation was chosen in order to minimize the adverse effects of boundary conditions on the numerical results. The body

is represented as a staircase geometry. Non-slip boundary conditions are imposed on the solid boundary.

The far field velocity $\mathbf{u}_\infty = (u_\infty, 0)$ is fixed on the inlet, top and bottom boundaries with $u_\infty = 1$. At the downstream boundary a Neumann-type boundary condition for the velocity is specified that corresponds to zero viscous stress vector [Mit01]. The kinematic viscosity is adjusted in order to reach a predefined Reynolds number (Re_D).

The most important physical quantities are the drag and lift forces $F_{x,y}$, and the vortex shedding frequency f. The corresponding nondimensional quantities are the drag and lift coefficients, and the Strouhal number:

$$C_{d,l} = \frac{F_{x,y}}{\frac{1}{2}\rho {u_\infty}^2 D}, \qquad \mathrm{St} = \frac{fD}{u_\infty} \tag{40}$$

To compute drag and lift a momentum balance over a control surface enclosing a square region containing the cylinder is used [LP00]. Since the control surface is close to the body, the error from not considering the inertia terms in the fluid inside the control surface can be neglected. The vortex shedding frequency is computed by counting the number of periods (crossing by zero) in lift history.

The results obtained at $Re_D = 1000$ are shown on Table (6), and the performance obtained is shown on Table (7).

Table 6: Flow past cylinder at $Re_D = 1000$.

		C_d	C_l	St
2D	Present formulation	1.56	1.3	0.211
	PFEM-2 [RIMNMG+13]	1.639	1.63	0.2475
	FEM [MK01]	1.48	1.36	0.21
3D	Experimental	1.00 [Ach72]		0.21 [Ros54]
	Present formulation	1.021	0.533	0.183
	PFEM-2 [RIMNMG+13]	1.16	0.2 to 0.3	0.185
	OpenFOAM [RIMNMG+13]	1.22	0.5	0.195

In this case, the stability of the scheme is controlled by advection, not diffusion, so a CFL up to 5 can be used. A time history of the coefficients is shown on Figure (9).

4.3.3 3D lid-driven cavity

This is the natural extension of the 2D lid driven square cavity example in Section (4.3.1) to 3D. Non-slip condition $u = v = w = 0$ is imposed at all the walls, except at the top, where

Table 7: Flow past cylinder at $Re_D = 1000$. Performance measured in [Mcells/sec] on a NVIDIA GTX 580.

		Simple	Double
2D	256×128	5.0	4.5
	512×256	16.7	11.1
	1024×512	33.3	20.0
3D	$64 \times 256 \times 256$	20.0	7.7

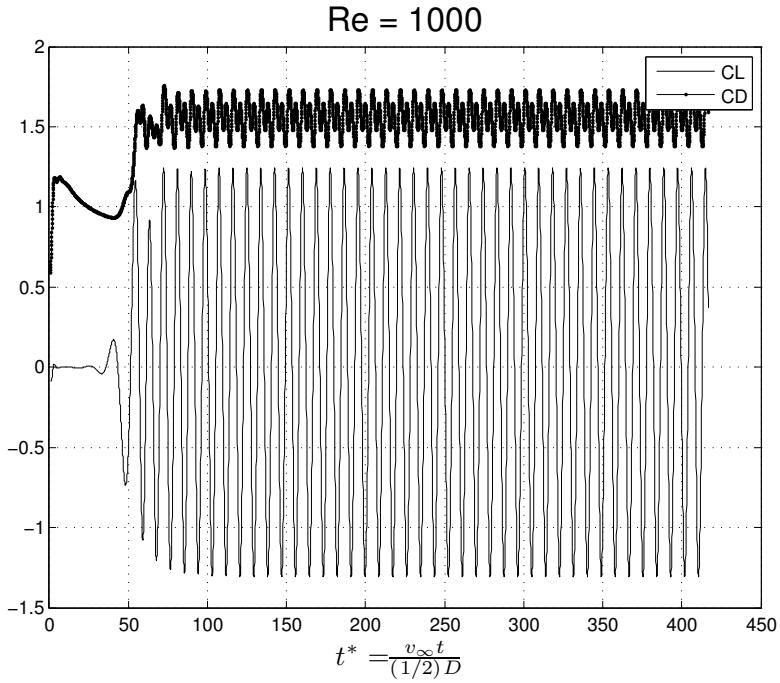


Figure 9: Results obtained at Re $= 1000$ using a grid of 1024×512.

$u = 1$, $v = w = 0$. The numerical results obtained at Re $= 1000$ are shown on Figure (10), and the performance obtained is shown on Table (8).

The performance of the code for performing a *real time computation*, that is computing the simulation of the physical process at the same as the physical system evolves (see [SPD+13]) will be analyzed. From Table (8), the performance obtained for the $128^3 \approx 2\,[\text{Mcells}]$ mesh is about $20\,[\text{Mcells/sec}]$, so that 10 time steps per second of computing time can be performed. As the time step for this case is $\Delta t = 0.01\,[\text{s}]$ it can be seen that $0.1\,[\text{s}]$ of simulation can be performed in $1\,[\text{s}]$ of computation. In this case also, for the finest meshes the stability constraint is diffusion-controlled, i.e. the critical time step is given by Equation (4).

Simple	[Mcell/sec]	[Mcell/sec]
Cells	QUICK	BFECC
$64 \times 64 \times 64$	29.09	12.38
$128 \times 128 \times 128$	75.74	18.00
$192 \times 192 \times 192$	78.32	17.81
Double	[Mcell/sec]	[Mcell/sec]
Cells	QUICK	BFECC
$64 \times 64 \times 64$	15.9	5.23
$128 \times 128 \times 128$	28.6	7.29
$192 \times 192 \times 192$	30.3	7.52

Table 8: Cubic cavity. Computing rates for the whole NS solver (one step) in [Mcell/sec] obtained with the BFECC and QUICK algorithms on a NVIDIA GTX 580. 3 Poisson iterations were used.

4.3.4 3D flow past circular cylinder

The cylinder case has been also run in the 3D case, with periodic boundary conditions along the cylinder axis direction. Performance results obtained at $Re_D = 1000$ are shown on Table (6) and time history of the coefficients is shown on Figure (11). The performance obtained is shown on Table (7). In this case stability is advection-controlled and $\Delta t = 0.023\,[\mathrm{s}]$ was used. The mesh has 7.1 Mcells and $0.23\,[\mathrm{s}]$ of simulation can be performed in $1\,[\mathrm{s}]$ of computation.

4.4 Performance analysis

The performance obtained by the present algorithm for advancing one time step in the NS solver (the four stages as described in Section (2)) in the case of the 3D cubic cavity is shown on Table (8). As a reference, this performance is compared with a previous implementation of the solver using QUICK [SPD+13]. Details of the two solvers follows:

- QUICK: this version of the momentum advection uses a high performance algorithm that makes use of shared memory, but has the drawback of a tight CFL condition, namely CFL<0.58 for the advection operator. For the NS solver the condition is even somewhat tighter, CFL<0.5 (see [CPDS11]).

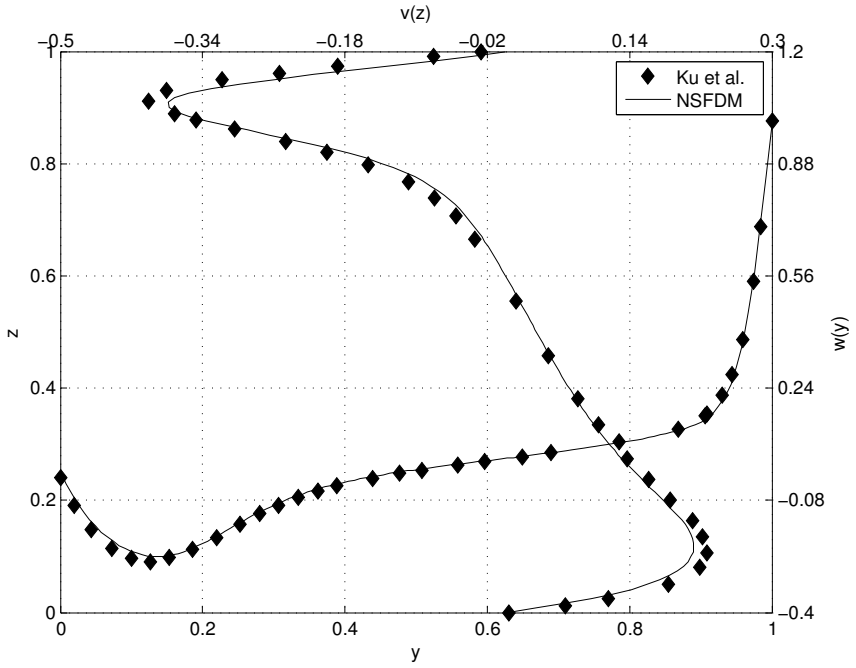


Figure 10: Results obtained at Re $= 1000$ using a grid of $128 \times 128 \times 128$.

- BFECC: the solution of the momentum equations using the BFECC algorithm described in this article. The CUDA implementation uses global memory. Note that while the great advantage of this implementation is that allows for CFL > 1, the cost of tracking the trajectories is proportional to the CFL number, and in addition this has the drawback that restricts the possibility of optimization, due to the spreading in memory accesses. Nevertheless, it will be shown that this algorithm is faster than the QUICK version.

The CUDA kernel proposed is memory bound due to many small accesses from global memory. This results in poor performance achievement but with the advantage of being unconditional stable, in other words, the time step can be chosen without any additional constraints. Even if semi-Lagrangian integration allows CFL > 1, another stability limit can be reached, namely the stability of the diffusion step (viscosity term). If this step is solved explicitly then the Fourier number Equation (4) is restricted to $\text{Fo} < \text{Fo}_{\text{crit}}$. In terms of CFL this restriction can be rewritten as:

$$\text{CFL} < \text{Fo}_{\text{crit}}\text{Re}_h \tag{41}$$

where Re_h is the cell size based Reynolds number. Normally, in many fluid mechanics computations of practical interest the global Re number is very high, and a mesh as fine as possible so as to have a $\text{Re}_h = O(1)$ would be desirable. However that would be

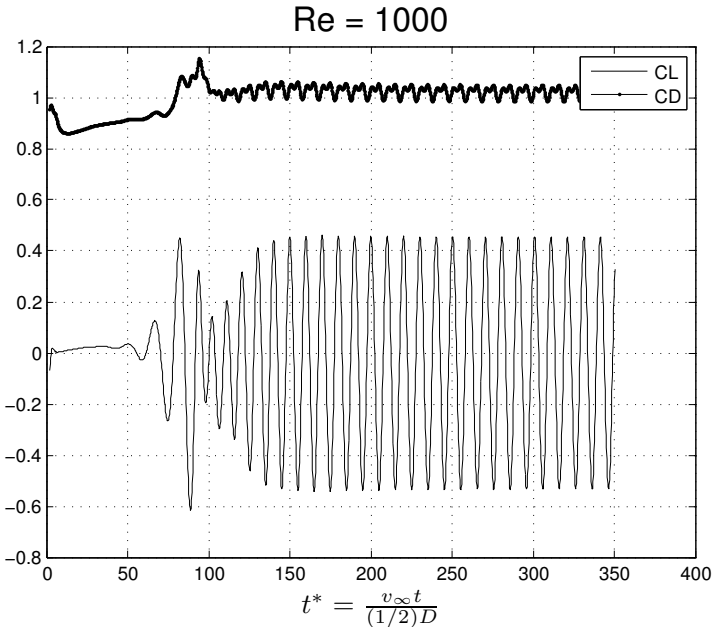


Figure 11: Results obtained at $Re_D = 1000$ using a grid of $96 \times 384 \times 192$.

computationally too expensive and usually $\mathrm{Re}_h > 1$. But on the other hand Re_h can not be too large, otherwise the small scale features of the flow are lost.

This restriction is overcome by solving the diffusion equation implicitly with a high performance FFT-based pure transient diffusion equations solver. With this modification the NS solver would be unconditionally stable, except for nonlinear instabilities.

Regarding the performance results shown in Table (8), it can be seen that the computing rate of QUICK is at most 4x faster than that of BFECC. So BFECC is more efficient than QUICK whenever used with CFL $>$ 2, being the critical CFL for QUICK 0.5. The CFL used in our simulations is typically CFL $\approx$ 5, provided that the stability is not diffusion-controlled, see Equation (4) and, thus, at this CFL the BFECC version runs 2.5 times faster than the QUICK version.

As a reference, the QUICK algorithm was implemented in CPU using the GNU g++ compiler (optimization flags `-O3 -funroll-loops`), obtaining a rate of 3.5 Mcell/sec on an Intel i7-3820@3.47 GHz (Sandy Bridge microarchitecture) for large 3D meshes (above 1 Mcell), i.e. 8.6 times slower with respect to the GPU(QUICK) version. Note that this speedup obtained on the GPU is close to the 8x speedup factor obtained for the FFT [SPD+13]. This is normal, because for the QUICK implementation a large part of the computing time is spent in the Poisson step. The BFECC(GPU) is only 2.15 times faster than the QUICK(CPU) version in Mcells/sec, but taking into account that the CFL is 10 times larger, the overall speedup is 21.5, i.e. BFECC(GPU) is 21.5 times faster than QUICK(CPU)

in computing one second of the same physical process.

Note that the computing rate highly increases with mesh size in the 2D case, both for the lid driven square cavity (see Table (5)) and the cylinder flow (see Table (7)), whereas it is almost constant or increases slightly in 3D cases. The reason for this is that 2D meshes have much fewer cells; in some cases the finest 2D cell has as much cells as the coarsest 3D one. (For instance the 512^2 and 64^3 meshes have the same number of cells.) Note that the same behaviour is observed for the GPU(QUICK) version and the CUFFT implementation itself [SPD+13].

A version of BFECC using shared memory is currently under development by performing the advection at CFL > 1 with repeated applications of advection substeps at CFL < 1. This split in smaller CFL numbers needs higher interpolation schemes, otherwise the dissipation is higher. So, on one hand, more efficient (in terms of Mcell/sec due to the GPGPU capabilities) algorithms may be developed but, on the other hand, high resolution interpolation schemes need to be introduced [SFK+08].

5 Conclusions

A GPGPU implementation of the BFECC algorithm with Nvidia's CUDA platform was tested against two classical benchmarks: Zalesak's disk and LeVeque's deformation field, in two and three dimensions. The proposed algorithm is more efficient than the explicit version using QUICK. Currently the performance is limited by the poor pattern access of the global memory, and also BFECC requires three access per time step as compared with QUICK with requires only one, but this is compensated by the higher CFL numbers that can be achieved. Numerical results are presented for the full Navier-Stokes solver, having a better performance than the QUICK version for CFL > 2.

6 Acknowledgements

This work has received financial support of:

- Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT, Argentina, grants PICT-1141/2007, PICT-0270/2008, PICT-2492/2010),
- Universidad Nacional del Litoral (UNL, Argentina, grants CAI+D 2009-65/334, CAI+D-2009-III-4-2) y

- European Research Council (ERC) Advanced Grant, Real Time Computational Mechanics Techniques for Multi-Fluid Problems (REALTIME, Reference: ERC-2009-AdG, Dir: Dr. Sergio Idelsohn).

Also we use some development tools under Free Software like GNU/Linux OS, GCC/G++ compilers, Octave, and *Open Source* software like VTK, among many others.

7 References

[Ach72] Elmar Achenbach. Experiments on the flow past spheres at very high reynolds numbers. *Journal of Fluid Mechanics*, 54(3):565–575, 1972.

[BC92] David Bleecker and George Csordas. *Basic partial differential equations*. CRC Press, 1992.

[BM82] JR Bates and A McDonald. Multiply-upstream, semi-lagrangian advective schemes: Analysis and application to a multi-level primitive equation model. *Monthly Weather Review*, 110(12):1831–1842, 1982.

[CFA01] Rachel Caiden, Ronald P Fedkiw, and Chris Anderson. A numerical method for two-phase flow consisting of separate compressible and incompressible regions. *Journal of Computational Physics*, 166(1):1–27, 2001.

[CMM90] Alexandre Joel Chorin, Jerrold E Marsden, and Jerrold E Marsden. *A mathematical introduction to fluid mechanics*, volume 3. Springer, 1990.

[CPDS11] Santiago Costarelli, R Paz, L Dalcin, and M Storti. Resolución de las ecuaciones de navier-stokes utilizando cuda. *Mecánica Computacional*, 30:2979–3008, 2011.

[CSP+12] Santiago Costarelli, Mario Storti, Rodrigo Paz, Lisandro Dalcín, and Sergio Idelsohn. A numerical algorithm for the solution of viscous incompressible flow on gpu's. *V Latin American Symposium on High Performance Computing HPCLatAm 2012*, 2012.

[CSP+13] Santiago D Costarelli, Mario A Storti, Rodrigo R Paz, Lisandro D Dalcin, and Sergio A Idelshon. Solving 3d viscous incompressible navier-stokes equations using cuda. *Proceedings of HPCLatAm*, pages 69–79, 2013.

[EFFM02] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational physics*, 183(1):83–116, 2002.

[ELF05] Douglas Enright, Frank Losasso, and Ronald Fedkiw. A fast and accurate semi-lagrangian particle level set method. *Computers & structures*, 83(6):479–490, 2005.

[Far11] Rob Farber. *CUDA application design and development*. Elsevier, 2011.

[GGS82] UKNG Ghia, Kirti N Ghia, and CT Shin. High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of computational physics*, 48(3):387–411, 1982.

[KLLR07] ByungMoon Kim, Yingjie Liu, Ignacio Llamas, and Jarek Rossignac. Advections with significantly reduced dissipation and diffusion. *IEEE transactions on visualization and computer graphics*, 13(1), 2007.

[Leo79] Brian P Leonard. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Computer methods in applied mechanics and engineering*, 19(1):59–98, 1979.

[LeV96] Randall J LeVeque. High-resolution conservative algorithms for advection in incompressible flow. *SIAM Journal on Numerical Analysis*, 33(2):627–665, 1996.

[LP00] Ming-Chih Lai and Charles S Peskin. An immersed boundary method with formal second-order accuracy and reduced numerical viscosity. *Journal of computational Physics*, 160(2):705–719, 2000.

[MCPN08] Jeroen Molemaker, Jonathan M Cohen, Sanjit Patel, and Jonyong Noh. Low viscosity flow simulations for animation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 9–18. Eurographics Association, 2008.

[MDHZ08] Xing Mei, Philippe Decaudin, Baogang Hu, and Xiaopeng Zhang. Real-time marker level set on gpu. In *Cyberworlds, 2008 International Conference on*, pages 209–216. IEEE, 2008.

[Mit01] S Mittal. Computation of three-dimensional flows past circular cylinder of low aspect ratio. *Physics of Fluids*, 13(1):177–191, 2001.

[MK01] S Mittal and V Kumar. Flow-induced vibrations of a light circular cylinder at reynolds numbers 10 3 to 10 4. *Journal of sound and vibration*, 245(5):923–946, 2001.

[NBGS08] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, 2008.

[Nvi10] CUDA Nvidia. Cufft library, 2010.

[PAB+97] Elbridge Gerry Puckett, Ann S Almgren, John B Bell, Daniel L Marcus, and William J Rider. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. *Journal of Computational Physics*, 130(2):269–282, 1997.

[PBS85] J Pudykiewicz, R Benoit, and A Staniforth. Preliminary results from a partial lrtap model based on an existing meteorological forecast model. *Atmosphere-ocean*, 23(3):267–303, 1985.

[RIMNMG+13] Sergio Rodolfo Idelsohn, Norberto Marcelo Nigro, Juan Marcelo Gimenez, Riccardo Rossi, and Julio Marcelo Marti. A fast and accurate method to solve the incompressible navier-stokes equations. *Engineering Computations*, 30(2):197–222, 2013.

[Rob81] André Robert. A stable numerical integration scheme for the primitive meteorological equations. *Atmosphere-Ocean*, 19(1):35–46, 1981.

[Ros54] Anatol Roshko. On the development of turbulent wakes from vortex streets. 1954.

[SC91] Andrew Staniforth and Jean Côté. Semi-lagrangian integration schemes for atmospheric models—a review. *Monthly weather review*, 119(9):2206–2223, 1991.

[SF99] Mark Sussman and Emad Fatemi. An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow. *SIAM Journal on scientific computing*, 20(4):1165–1191, 1999.

[SFK+08] Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. An unconditionally stable maccormack method. *Journal of Scientific Computing*, 35(2):350–371, 2008.

[SPD+13] Mario A Storti, Rodrigo R Paz, Lisandro D Dalcin, Santiago D Costarelli, and Sergio R Idelsohn. A fft preconditioning technique for the solution of incompressible flow on gpus. *Computers & Fluids*, 74:44–57, 2013.

[Sta99] Jos Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128. ACM Press/Addison-Wesley Publishing Co., 1999.

[Zal79] Steven T Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of computational physics*, 31(3):335–362, 1979.

Título:

An Embedded Strategy for the Analysis of Fluid Structure Interaction Problems

Contribución:

Escritura del artículo, desarrollo e implementación GPGPU del método numérico

Firma del director de tesis:

An Embedded Strategy for the Analysis of Fluid Structure Interaction Problems

Santiago D. Costarelli[1,2], Luciano Garelli[1], Marcela A. Cruchaga[3], Mario A. Storti[1,2], Ronald Ausensi[3] and Sergio R. Idelsohn[1,4,5]

[1]*Centro de Investigación de Métodos Computacionales - CIMEC, CONICET-UNL, Santa Fe, Argentina* http://www.cimec.org.ar

[2]*Facultad de Ingeniería y Ciencias Hídricas. UN Litoral. Santa Fe, Argentina,* http://fich.unl.edu.ar

[3]*Departamento de Ingeniería Mecánica, Universidad de Santiago de Chile (USACH) / Santiago de Chile, Chile*

[4]*Institució Catalana de Recerca i Estudis Avançats (ICREA), Barcelona, Spain*

[5]*International Center for Numerical Methods in Engineering (CIMNE), Technical University of Catalonia (UPC), Gran Capitán s/n, 08034 Barcelona, Spain,* http://www.cimne.upc.edu

Abstract

A Navier-Stokes solver based on Cartesian structured finite volume discretization with embedded bodies is presented. Fluid structure interaction with solid bodies is performed with an explicit partitioned strategy. The Navier-Stokes equations are solved in the whole domain via a Semi-Implicit Method for Pressure Linked Equations (SIMPLE) using a colocated finite volume scheme, stabilized via the Rhie-Chow discretization. As uniform Cartesian grids are used, the solid interface usually do not coincide with the mesh, and then a second order Immersed Boundary Method is proposed, in order to avoid the loss of precision due to the staircase representation of the surface. This fact also affects the computation of fluid forces on the solid wall and, accordingly, the results in the fluid-structure analysis. In the present work, first and second order approximations for computing the fluid forces at the interface are studied and compared. The solver is specially oriented to General Purpose Graphic Processing Units (GPGPU) hardware and the efficiency is discussed. Moreover, a novel submerged buoy experiment is also reported. The experiment consists of a sphere with positive buoyancy fully submerged in a cubic tank, subject to harmonic displacements imposed by a shake table. The sphere is attached to the bottom of the tank with a string. The position of the buoy is determined from video records with a Motion Capture algorithm. The obtained amplitude and phase curves allow a precise determination of the added mass and drag forces. Due to this aspect the experimental data can be of interest for comparison with other fluid-structure interaction codes. Finally, the numerical results are compared with the experiments, and allows the confirmation of the numerically predicted drag and added mass of the body.

Keywords: Graphics Processing Units; Incompressible Navier-Stokes; Embedded methods; Added mass; Fluid structure interaction; Fully submerged buoy experiments

Contents

1 Introduction

Nowadays several engineering and scientific applications require extensive use of Fluid-Structure Interaction (FSI) analysis. Energy generation, ship maneuvering, offshore oil platforms, buoy structures, are some of the applications in the field of ocean engineering. FSI is also applied in other areas, like aeronautics, hydraulics, automotive, and many other fields. Due to this, several works can be found in the literature dealing with numerical algorithms for modeling FSI problems [PF01, BHK+11, TBT12, TT11, WL04,

HZB12, HGC+14, BSTL11, RIO+11, RMSF11]. To this end the formulations require the solution of two different fields, the fluid and the structure, and they must be coupled through their common interface.

In the monolithic approaches [HT06, IOPC06, IMLO08, TT11, RMSF11], both fields are solved simultaneously, leading to a unique set of equations for both the fluid and structure fields. This approach tends to be computationally expensive, due to the size of the combined problem, and also usually they have dissimilar time and length scales, and consequently the strategy to solve each field are different. An alternative approach to avoid these drawbacks is to use a partitioned strategy, i.e. to solve each one of the structural and fluid subproblems separately, and couple them explicitly or iteratively. Of course, when using this approach care must be taken to assess the stability and precision of the scheme. To compute these kind of problems with acceptable accuracy in reasonable computational times, it is extensively used High Performance Computing (HPC) techniques like parallel processing in distributed and shared memory architectures, hybrid, and recently coprocessor boards like General Purpose Graphic Processing Units (GPGPU) or Many Integrated Core (MIC) [CCLW11, CCLM12, VHRS14, RC13, TT12, LDB14]. Each of these architectures condition the underlying algorithm.

In [SPD+13, CSP+14] a numerical strategy to deal with immersed rigid bodies with imposed motion has been presented. The scheme is based on a staggered Finite Volume Method (FVM) and velocity and pressure are computed with a Fractional Step Method (FSM). Advection terms are stabilized with Quadratic Upstream Interpolation for Convective Kinematics (QUICK), and the Poisson equation is solved with the Fast Fourier Transform (FFT). The FFT solver is very efficient when the whole domain is filled with a single fluid material. In cases where there are immersed bodies, the Poisson equation must be solved only on the fluid part, and the FFT solver can not be directly applied. In [SPD+13] an iterative scheme to solve the Poisson equation with immersed bodies using the FFT on the whole domain as a preconditioner has been presented. In [CSP+14] QUICK is replaced by the Method of Characteristics combined with the Back and Forth Error Compensation and Correction (BFECC) technique so as to extend the CFL (Courant-Friedrich-Levy) range of stability increasing the performance of the algorithm. These algorithms were designed for an efficient implementation on GPGPU hardware, and very high cell computing rates were obtained. The treatment of the embedded body surfaces were staircase (i.e. first order approximation, as is also reported in [RMSF11] using a different technique). However, correct results for drag and lift values have been obtained in the flow past cylinder

benchmark at Re $= 1000$. Nevertheless, in the fluid-structure problem proposed in this work the results obtained with the described methodology denotes an overestimation of the drag force, which is consistent with a low-order surface representation.

As it is well reported in the literature, the level-set technique is effectively used to describe moving boundaries and interfaces. The existing algorithms can be basically classified as geometric (for instance Fast Marching [Set96]) or PDE based [PMO+99, CMOS97, HLOZ97]. The geometric algorithms give an accurate definition of the signed distance level set function. However they are computationally expensive, and difficult to parallelize on the GPGPU. On the contrary, PDE based algorithms, e.g those including BFECC techniques [DL03, DL07, CSP+14], are comparative less expensive, very GPGPU friendly, and strictly $O(N)$. Nevertheless, their computation introduce cumulative errors that corrupt the solution of the level set function loosing precision on the representation of the body. To solve this drawback, PDE versions can be combined with geometric algorithms, e.g. reinitialization, redistancing in a narrow strip around interfaces or the Particle Level-Set Method [EFFM02].

To overcome the low accuracy obtained with the staircase first order scheme used in [SPD+13, CSP+14] a new numerical methodology is proposed in the present work comprising a second order scheme for a solid embedded technique based on a colocated Finite Volume Method stabilized with the Rhie-Chow approximation, including a level set description of the solid. The velocity-pressure coupling is dealt with the SIMPLE algorithm. The combination of all these features leads to a numerical scheme that can be efficiently implemented in the GPGPU architecture.

In order to verify this novel numerical scheme, comparisons with a well established numerical technique and also with experimental results have been performed.

The FSI strategy proposed in [GPS10, SGP12, SNPD09] will be used in the present work as a reference. It is based on the Finite Element Method (FEM) and includes an Arbitrary Lagrangian-Eulerian (ALE) technique. The fluid discrete equations are written using a SUPG+PSPG approach [Tez91]. The fluid mesh is boundary fitted to the actual position of the solid, and the nodes are relocated using a minimum element distortion strategy [LNST07]. Mesh movement is taken into account with an ALE technique. Fluid structure interaction is performed with a staggered scheme [SNPD09]. The coupling can be iterated in order to reach a strong coupled solution. A predictor can be used for the solid position for increasing the precision of the algorithm when a fully explicit temporal coupling is used. This approach has the advantage that allows for a very good

representation of the flow in the boundary layer of the body surface, but for very large displacements of the body the mesh could collapse in the gap between the solid and the tank walls [LNST07].

In the present work we propose an experiment designed to capture the effects of fluid forces on a rigid body. Specifically, it attempts to describe added mass and drag forces on a sphere in motion. From the experiment we determine the amplitude and phase of oscillation of a submerged sphere with positive buoyancy. These results are compared with the numerical ones and it turned out to be very sensitive to the drag forces, and then it is a challenging benchmark for FSI codes.

The remaining of this work is organized as follows. Section (2) describes the experimental setup and procedure. The governing equations are summarized in Section (3). A reduced analytic model is described in (4). Section (5) presents the proposed numerical scheme. Numerical results and comparisons are shown in Section (6). Specific aspects of the GPGPU implementation and efficiency are discussed in Section (7). The conclusions are presented in Section (8).

2 Experimental setup

The experiment presented in this article consists of a sphere of silicone immersed in a cubic tank filled with water, see Figure (1). The sphere has a diameter of $D = 0.10$ m and a density $\rho = 366.69$ kg/m^3, i.e. lighter than water. A thread of 0.16 m attaches the sphere to the bottom of the tank. The acrylic tank of section 0.38 m x 0.38 m and 0.40 m of height has two covers at its top that inhibit free surface evolution. The tank is mounted over a vibrating table Shake-Table II and it is subject to oscillatory motion at different imposed frequencies.

It is important to mention that a heavy pendulum could be used also for the experiment. However in that case the influence of the fluid forces would be smaller compared with the inertia of the body. In the present case the added mass is 1.47 times the mass of the buoy. If the experiment would consist in a heavy pendulum the added mass would represent at most 0.54 of the mass of the buoy and, accordingly, the incidence of the fluid forces would be negligible. For that reason we have chosen a positive buoyant body.

The objective is to study the sphere motion developed under different imposed conditions. To this end, the experiments are recorded with an AOS High Speed Camera and the video frames are analyzed using a Motion Capture technique. The camera is

Figure 1: Experimental layout.

positioned on a perpendicular distance of 2.25 m from the tank, and a lens of 35 mm is used. A resolution of 800×800 pixels is utilized, in which the reservoir can be fully

appreciated. A frame rate of 120 frames per second is enough to capture the movement of the sphere. Resulting in a precision of 0.5 mm in space and 0.008 s in time. With this configuration approximately 2000 frames can be recorded covering a time of 16.7 s.

The experimental buoy amplitude A_{buoy} and phase φ (time shift between imposed displacement and sphere center of mass position, converted to degrees) are obtained during the time-periodic regime. The imposed table displacements have an amplitude of $A_{\text{tank}} = 0.02\text{m}$ and frequencies f in the range $[0.5 - 1.5]$ Hz, being 0.9 Hz the analytic primary resonance frequency of the system.

It should be mentioned that a second camera positioned in alignment with the table motion was used to register 3D behavior. From these videos planar (x, y) motion was detected up to resonance frequency. After that frequency a slight deviation of the sphere from the middle plane is observed. Nevertheless, the planar motion is recovered approximately at frequency 1.2 Hz. This experimental observation validates the use of a planar model in the reported range of frequencies.

Figure (2) shows the evolution of the center mass of the sphere and the imposed motion where amplitudes and phases are indicated.

The experimental results are summarized in Sections (4) and (6.2) together with analytic and numerical responses, respectively.

The flow can be characterized by the Keulegan-Carpenter number (relating the drag and inertia forces in oscillating bluff bodies) $K = U_{\max}/(fD)$, where $U_{\max}$ is the maximum body velocity, and the Stokes number $\beta = D^2\rho f/\mu$ (inversely related to the viscous forces). Based on the experimental results K varies from 0.4 to 5.6 and β ranges from 5000 to 15000, see Section (6.2).

2.1 Motion Capture Technique

The developed Motion Capture technique is based on a minimization of the position captured between two consecutive images.

Assume that the body is initially at 2D position $x(t)$ and in the next frame moves to position $x(t') = x(t) + a$. In general a is a set of parameters describing a homography transformation, for simplicity in the present description only translations are considered. If the pixel intensity is represented as a function $f(x)$ then we have at time t':

$$f(x, t') = f(x - a, t) \tag{1}$$

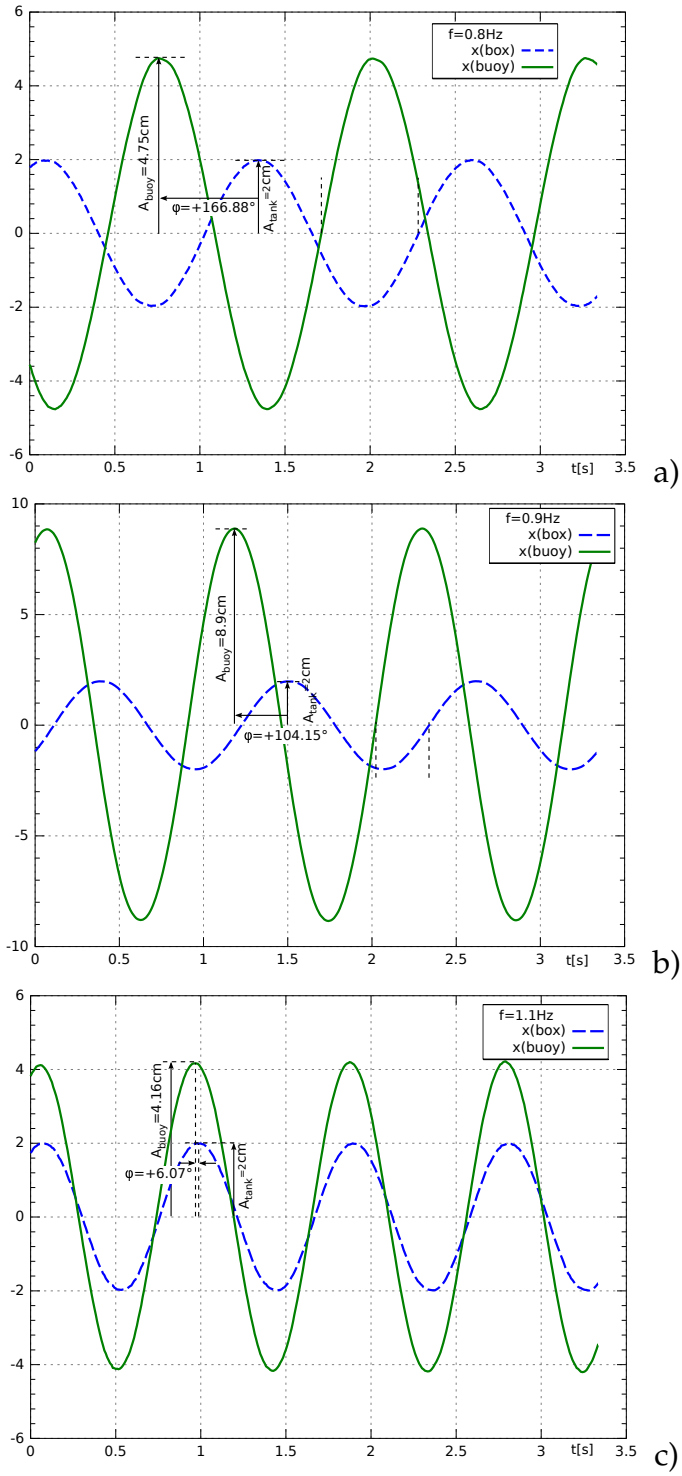


Figure 2: Sphere and tank motion during time-periodic regime at imposed frequencies of a) 0.8 Hz, b) 0.9 Hz, and c) 1.1 Hz ($A_{\text{tank}} = 20\text{mm}$).

so given the pixel intensity at the reference position $f(x,t)$ and at other time $f(x,t')$ we can find the translation a by minimization of the L_2 norm of the deviation functional:

$$\Phi(a) = \int |f(x,t') - f(x-a,t)|^2 \, d\Omega \tag{2}$$

The advantage of using the L_2 norm is that the minimization process leads to a smooth

gradient of the functional, and then can be solved for a by the Newton-Raphson technique. As the pixel intensity is a piecewise constant function, a monotone bicubic interpolation is performed, so that the functional has second order continuous derivatives, and the Newton-Raphson scheme can be applied. Figure (3) shows three frames with the positions of the sphere and the anchor point determined by the Motion Capture algorithm as a white box.

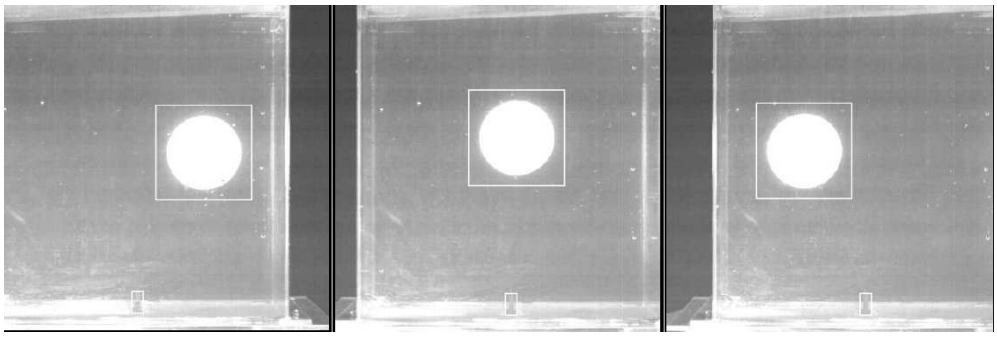


Figure 3: Motion capture detection of the sphere and anchor position. Detected positions are shown as white boxes.

3 Governing equations

Fluid-solid interaction problems require to solve flow and solid fields. We model the problem in the context of continuum mechanics and we assume that the fluid flow is described using the incompressible Navier-Stokes equations while conservation of momentum will be applied to the rigid body.

3.1 Fluid flow

The Navier-Stokes equations for a Newtonian fluid reads:

$$\rho\left(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}\right) + \nabla p - \nabla\cdot\left(2\mu\mathbf{ffl}(\mathbf{u})\right) = \rho\mathbf{f} \text{ in } \Omega, \tag{3}$$

$$\nabla\cdot\mathbf{u} = 0 \text{ in } \Omega \tag{4}$$

for time $t \in [0, T]$, with T a final time, where Ω is the fluid domain, $\mathbf{u}$ is the fluid velocity, $\mathbf{f}$ is the body force, ρ is the fluid density, p is the pressure, μ is the dynamic viscosity, and $\mathbf{ffl}(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + \nabla^T\mathbf{u})$ is the strain rate tensor. Moreover, the subindex t denotes time derivative and ∇ is the gradient operator.

The boundary conditions at the contour of the domain Γ are

$$\mathbf{u} = \mathbf{u}_D \qquad \text{on } \Gamma_D\,, \tag{5}$$

$$\boldsymbol{\sigma} \cdot \mathbf{n} = \mathbf{t} \qquad \text{on } \Gamma_{\mathbf{t}}\,, \tag{6}$$

where velocities are given for the Dirichlet part Γ_D, e.g. over solid walls, while $\Gamma_{\mathbf{t}}$ admits Neumann boundary conditions. Note that $\Gamma = \Gamma_D \cup \Gamma_{\mathbf{t}}$ and $\Gamma_D \cap \Gamma_{\mathbf{t}} = \emptyset$.

3.2 Rigid body governing equations

In the present analysis the sphere motion is reduced to a one degree of freedom (DOF) equation for the model of a submerged buoy sketched in Figure (4). We assume that the string is straight, non-extensible, and the center of the sphere is aligned with the string. Also we assume that the sphere does not rotate about the axis of the string, and that the movement of its center is restricted to the main plane of movement (plane $x - y$, see Figure (1)).

Hence, the body movement is reduced to one DOF, namely the angular deflection of the center of the sphere θ taken positive in clockwise direction, and the corresponding angular momentum equation, written in a non-inertial reference system fixed to the tank, reads as:

$$I_s \ddot{\theta} = m_s g L \sin\theta - m_s a_{tank} L \cos\theta + T_{\text{fl}} \tag{7}$$

where I_s is the momentum of inertia of the sphere with respect to the anchor point, m_s is the mass of the buoy, g is the gravity acceleration, a_{tank} is the time depending acceleration of the tank, L is the distance from the center of gravity of the sphere to the anchor point, and T_{fl} encompasses the torque for all the forces of the fluid on the sphere (positive clockwise), including added mass, drag forces, and buoyancy due to gravity and inertial effects (see Section (4)). Note that Equation (7) takes into account the inertial force on the solid due to the tank acceleration.

In the coupled fluid-solid model, the fluid forces are computed from the pressure and velocity fields given by solving the system of Equation (3)) and Equation (4) according to the numerical strategies presented in Section (5).

4 Approximate analytic solution

With the aim to estimate the resonant frequency, added mass and drag coefficient, Equation (7) can be approximated with the following reduced analytic model:

$$\begin{aligned} I_s\ddot{\theta} &= m_s g L \sin\theta - m_s a_{tank} L \cos\theta + T_{bg} + T_{ba} + T_a + T_d \\ &= m_s g L \sin\theta - m_s a_{tank} L \cos\theta \\ &\quad - m_{\text{fl}} g L \sin\theta + m_{\text{fl}} a_{tank} L \cos\theta - m_a L^2 \ddot{\theta} - 1/2 \rho_f |v|^2 \,\text{sign}(v)\, C_d A_{\text{r}} L \end{aligned} \tag{8}$$

where $v = L\dot{\theta}$ is the tangential velocity , T_{bg} and T_{ba} are the torques generated by buoyancy due to gravity and tank acceleration respectively, T_a is the torque due to the added mass effect, and T_d the torque produced by the drag force. Standard expressions have been assumed for the drag and added mass forces. In addition m_{fl} is the mass of fluid displaced by the sphere, m_a is the added mass, $A_{\text{r}} = \pi D^2/4$ is the projected area of the sphere, and C_d is the drag coefficient. The added mass is usually made non-dimensional by defining the added mass coefficient $C_a = m_a/m_{\text{fl}}$.

Under a small displacement assumption $\theta \ll 1$ a closed expression for the resonance frequency can be obtained:

$$2\pi f_n = \sqrt{\left(\frac{m_{\text{fl}} - m_s}{m_s + m_a}\right)\frac{g}{L}} \tag{9}$$

Analytic, numerical, and experimental values for C_d and C_a are reported in the literature. However C_d is usually reported for a steady state free stream condition. With respect to the added mass, it is well known the value given by potential flow (one half of the displaced mass for a sphere). The present analysis is far from those ideal situations, i.e. the velocity is not constant and the domain of analysis is restricted to the tank (not a free stream condition).

Hence, the added mass and drag coefficients are unknown for the case of study and they are determined by minimizing the error between the experimental and analytic values of the curve of maximum sphere displacement vs. frequency during forced time-periodic regime, see Figure ((5)).

Given C_d and m_a, Equation (8) is solved with a standard ODE discretization technique with an imposed tank motion of amplitude $A_{\text{tank}} = 0.02$m and frequency f in range 0.5 to 1.5 Hz, as reported in Section (2). From this analysis the maximum sphere displacement during forced time-periodic regime are determined and plotted versus imposed frequency. The added mass and drag coefficients that best fit the experimental curves are $C_a = 0.54$ and $C_d = 0.25$. The added mass coefficient agree with those reported in [NLD07].

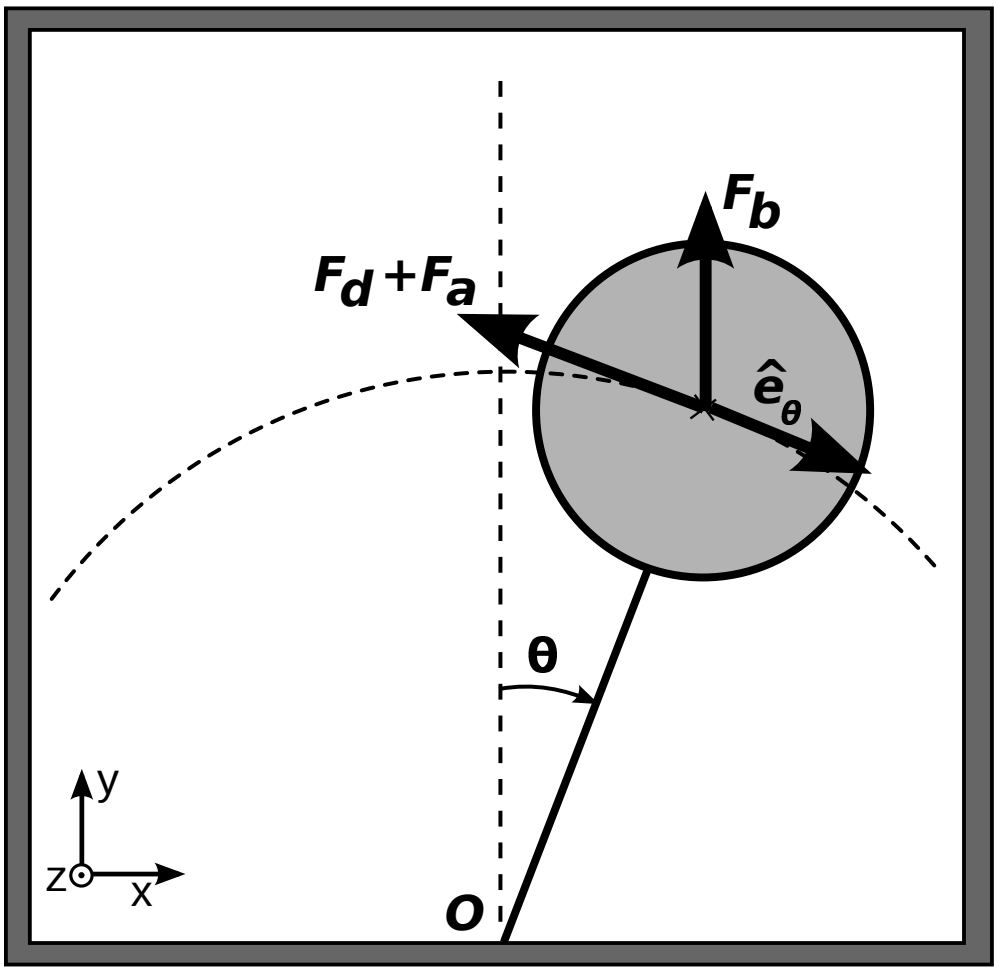


Figure 4: Sketch of fluid forces acting on a submerged buoy. Simplified 1 DOF model.

In addition, considering resonance conditions, i.e. $f = 0.9\,\text{Hz}$ and buoy amplitude $A_{\text{buoy}} = 0.089\,\text{m}$, the resulting characteristic numbers are $K = 5.59$ and $\beta = 9000$, for which the coefficients reported in the experimental work [S+76] are C_a in range $[0.45, 0.55]$ and C_d in range $[0.25, 0.45]$. Note that in [S+76] the fluid is not confined, and harmonic linear motion is considered.

The phase response of the system computed with the analytic model using the fitted values ($m_a = 0.54 m_{\text{fl}}$, $C_d = 0.25$) is also shown in Figure ((5)), as a further verification.

It is important to mention the influence on the system response (see Figure ((5))) of the different terms present in the reduced equation (8). In that equation, the most difficult fluid terms to accurately compute are the drag force, the added mass, and the buoyancy term, in such order. In embedded schemes, like the one presented in this work, it is particularly difficult to capture the drag, because of the staircase effect on the solid-fluid interface.

At low frequencies ($f < 0.8\text{Hz} < \text{f}_\text{n}$), the dominant term is the gravity buoyancy one that acts as the restitutive force, while the added mass and drag term (damping) do not

play an important role. At large frequencies ($f > 1.0\text{Hz} > \text{f}_\text{n}$), the added mass is the dominant term, so if the numerical response fails to adjust the experimental one in that regime, it means that the added mass effect is not well captured. Finally, for frequencies near resonance ($0.8\text{Hz} < f < 1.0\text{Hz}$) the restitutive and added mass forces cancel each other out, and the maximum amplitude is mainly governed by the drag force. Due to this aspect, it is an important test for embedded methods to properly describe this curve near resonance. On the other hand the resonant frequency directly depends on the added mass (according to Equation (9)). In the present analysis the resonant frequency computed with the reported parameters is $f_n = 0.91\text{Hz}$.

The phase response curve, also reported in Figure ((5)), gives relevant information in order to compare the numerical results. In this system the driving force is proportional to the net buoyancy of the sphere. As the sphere has positive net buoyancy, it experiments a force in the same direction of the tank acceleration. Note that if the sphere would have a net negative buoyancy, then the driving force would be in the opposite direction to the tank acceleration. As the tank experiments harmonic motion, the acceleration is always in opposite phase to the tank displacement.

For low frequencies, driving force is in phase with the sphere displacement, and in opposite phase with the tank displacement. On the other hand, for high frequencies, the driving force is in opposite phase with the sphere displacement, and consequently in phase with the tank displacement. At resonance, the sphere displacement is near at a phase of $\pi/2$ with the tank displacement. These aspects are also illustrated in Figure (2). Note that this behavior is opposite to the expected one for a heavy (negative net buoyancy) pendulum.

It is important to remark that the $C_d = 0.23$ found from the numerical model is close to the experimental one $C_d = 0.25$, while both being far from the standard value of $C_d = 0.47$.

5 Finite volume fluid-structure interaction strategy with second order embedded bodies

The second order treatment of immersed boundaries will follow [MY98, FVOMY00]. However, this technique is hard to combine staggered grids and Fractional Step schemes. Due to this aspect a new formulation is proposed, including a Rhie-Chow interpolation [RC83, VM07] for avoiding staggered grids and SIMPLE algorithm [PS72]

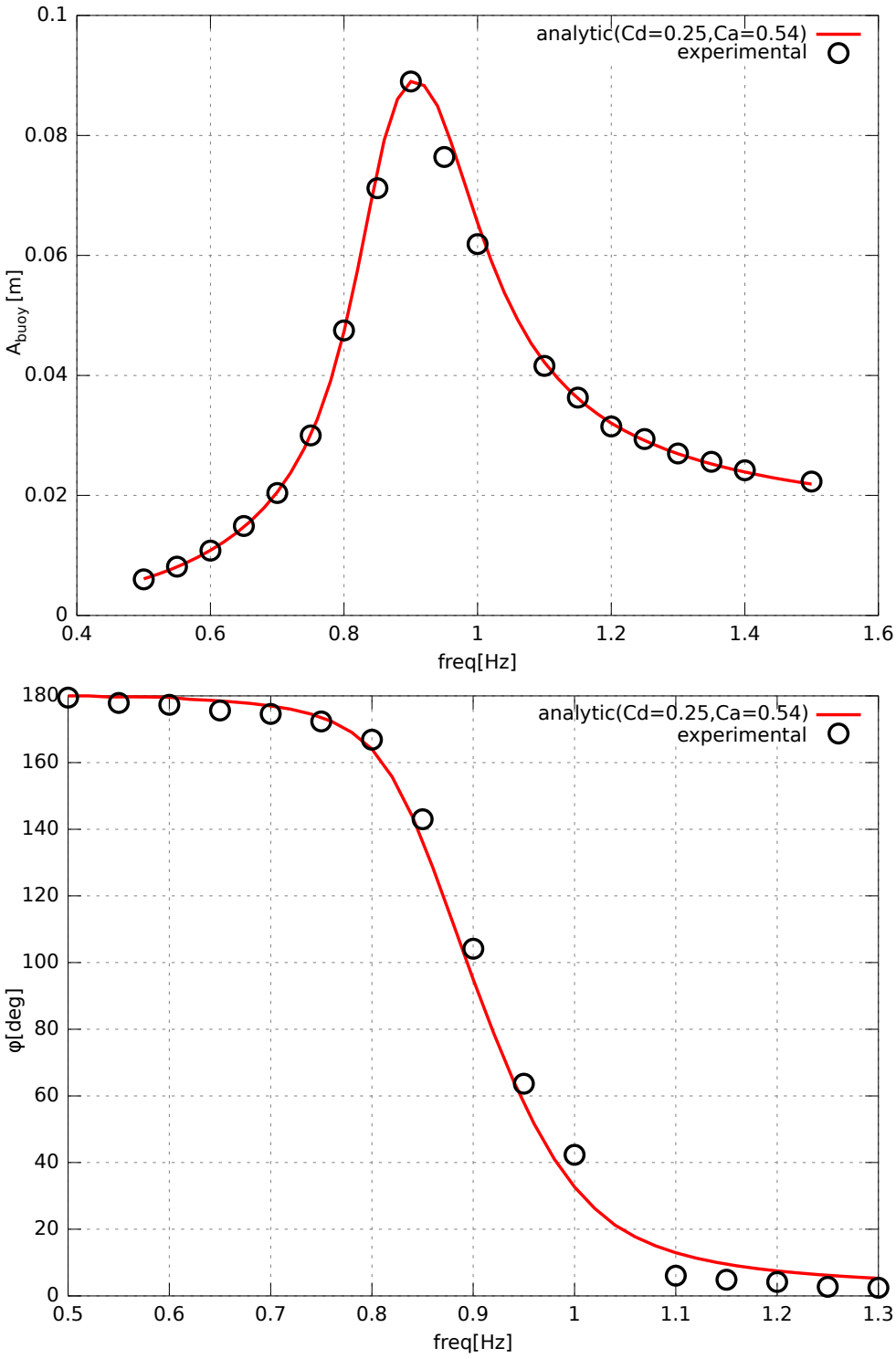


Figure 5: Analytic vs. experimental results.

for iterating the pressure-velocity coupling. This algorithm will be referred afterwards as FVM/IBM algorithm.

The complete algorithm reads as follows,

Algorithm 1: SIMPLE + RHIE-CHOW + IBM

1: Data initialization: $u^0, p^0, q^0, \dot{q}^0$

2: **for** $n \leftarrow 0$, to $n = N_{\text{step}} - 1$ **do**

3: $u^* \leftarrow u^n$, $p^* \leftarrow p^n$

4: **for** $k \leftarrow 0$, to $n = N_{\text{simple}} - 1$ **do**

5: $w \leftarrow u^*$, $f \leftarrow 0$

6: **for** $l \leftarrow 0$, to $n = N_{\text{IBM}} - 1$ **do**

7: $w \leftarrow Gp^* + R(w) + f$

8: $\bar{w} \leftarrow \Pi_{\text{BC}}(w, q_S^n, \dot{q}_S^n)$

9: $f' \leftarrow \bar{w} - Gp^* - R(w)$

10: $\epsilon_f \leftarrow \|f - f'\| / \|f^{l=0}\|$

11: $f \leftarrow f'$, $w \leftarrow \bar{w}$

12: If $\epsilon_f < \text{tol}_f$ break

13: **end for**

14: $u \leftarrow \bar{w}$

15: Solve $A(p - p^*) = A'p^* + Du$ for p

16: $\epsilon_u \leftarrow \|u - u^*\| / \|u - u^n\|$

17: $u^* \leftarrow \alpha_u u + (1 - \alpha_u)u^*$

18: $p^* \leftarrow p^* + \alpha_p p$

19: If $\epsilon_u < \text{tol}_u$ break

20: **end for**

21: $u^{n+1} \leftarrow u^*$, $p^{n+1} \leftarrow p^*$

22: Compute the tractions of the fluid onto the solid t_{fl}

23: Compute the new solid position q_S^{n+1} and velocity $\dot{q}_S^{n+1}$

24: Update level set function

25: **end for**

For each time step n there are two nested loops, the most external one (k-index) is for the SIMPLE iteration that enforces compressibility through iterating the momentum and continuity equation. The inner loop over index l is the proposed method for imposing the solid body boundary condition with second order accuracy (Immersed Boundary Method, IBM). It first solves the momentum equation to update the intermediate velocity field w. As this velocity field does not necessarily satisfy the boundary conditions the projection operator Π_{BC} is applied and produces a modified new velocity field $\bar{w}$ that is enforced to satisfy the BC condition. This new $\bar{w}$ does not satisfy the momentum equation, so compensating forces f that act only in the fictitious fluid inside the solid are computed.

Now the momentum equations is solved again including this compensating forces f and this procedure is iterated up to convergence.

In addition, the generalized coordinates and velocities q_S^{n+1} and $\dot{q}_S^{n+1}$ are computed in the present case according to Equation (7). In this case, the generalized coordinates is simply the rotation angle θ.

The discrete operators in the algorithm are as follows. G is the gradient. R englobes the remaining terms in the right hand side of the momentum equation (diffusion, convection, and body force term). A is the Laplace operator and A' is Rhie-Chow contribution to the Poisson equation that acts as stabilization term for the compressibility constraint. D is the divergence operator.

In [SPD+13] the FFT solver was used as a preconditioning for a Conjugate Gradient (CG) iteration. In this work, the solid is occupied by a fictitious fluid, hence the Poisson equation is posed in the whole domain, then the CG converges in just one iteration, i.e. the FFT is a direct solver for that equation. This is an additional advantage of this algorithm to run on GPGPUs.

5.1 The boundary condition projection operator

In this section we give the details for the computation of the projection operator Π_{BC} in the previous algorithm. The purpose of the projection operator is to extend the fluid velocity field to the solid region, such that the velocity at the body surface is the prescribed one [MY98]. This prescribed velocity may come from an imposed body motion or else from the rigid body velocity field determined by the fluid-structure interaction.

The body is described by a level set function ψ [OF06, CBSD14] such that by convention $\psi = 0$ defines the body surface, and $\psi < 0$ ($\psi > 0$) identifies the solid (fluid) region. For the present case of study ψ is computed analytically from the position of the center of the sphere.

Considering the situation in Figure (6), the black (white) filled dots represent cell centers that belong to the solid (fluid) region. The velocity values in the fluid dots are known, and Π_{BC} must extrapolate the values to solid dots, preserving the boundary condition at the body surface. Taking as an example the solid dot S, and B being the boundary point closest to S then:

$$x_S = x_B - dn_B \tag{10}$$

where d is the distance between B and S and n_B is the normal to the surface at B. The

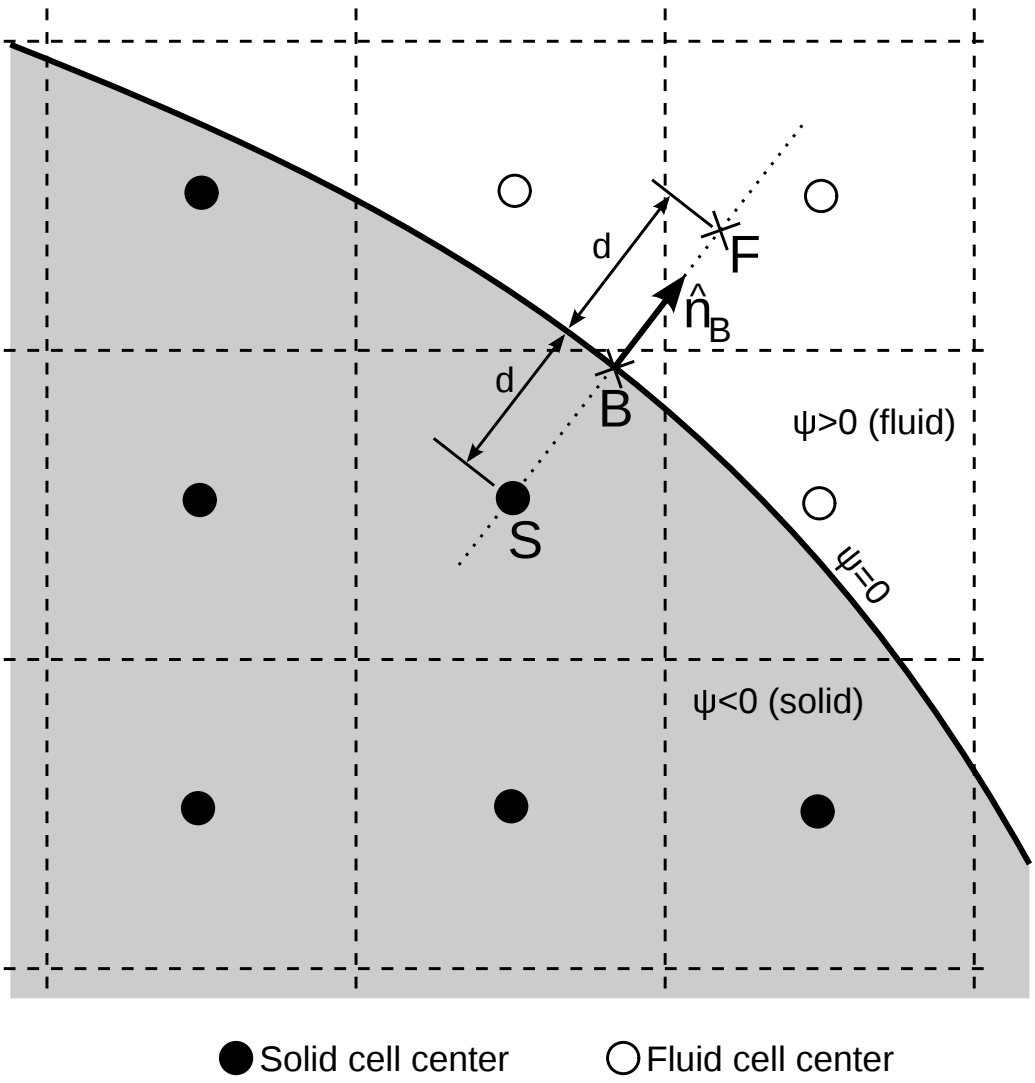


Figure 6: Projection operator Π_{BC} nomenclature.

normal to the surface can be computed as:

$$n_B = \left(\frac{\nabla\psi}{\|\nabla\psi\|}\right)_B \approx \left(\frac{\nabla\psi}{\|\nabla\psi\|}\right)_S \tag{11}$$

Let F be the mirror point of S with respect to the body surface, i.e. F lies along the projection of the normal that passes through S, at a distance d from the boundary, i.e.:

$$x_F = x_B + dn_S \tag{12}$$

The velocity u_F at x_F is obtained by interpolation from the velocity values at the nearest cell centers (indicated as dots in the figure). From this u_F value an extrapolated velocity u_S^f at point x_S is computed to ensure the boundary velocity value u_B at x_B. Note that this u_S^f velocity field inside the solid is fictitious. A simple linear extrapolation leads to:

$$u_S^f = 2u_B - u_F \tag{13}$$

The proposed technique has the advantage that the implementation on the GPGPU avoids branch divergence, since all the computations are straightforward, without involving conditionals, and preserves data locality.

The level-set function used in this work was computed analytically due to the simplicity of the geometry. For complex geometries a more general approach must be used paying special attention to the implementation of the level set in order not to degrade the global efficiency of the GPGPU algorithm. To this end, algorithms mixing PDE based and geometric algorithm could be used.

6 Numerical results

The present section reports the simulations addressed to assess the capabilities of the proposed second order finite volume formulation applied to fluid structure interaction with immersed rigid bodies. To this end a 2D numerical test with prescribed body motion is presented with the object to verify the numerical response with a Finite Element technique [GPS10, SGP12, SNPD09]. This example also consists in a reduced model of the experiment reported in Section (2), where no FSI is computed and is restricted to evaluate the forces induced by the fluid flow. In the second example a 3D model of the experiment reported in Section (2) is analyzed taking into account the FSI as described in the present work, resulting in an accurate representation.

The techniques that will be used in the modeling will be referred as: FVM/IBM2 for the second order algorithm presented in this article and detailed in Section (5), FVM/IBM1 for the first order technique reported in [SPD+13, CSP+14], and FEM/ALE for the reference scheme described in [GPS10, SGP12, SNPD09].

6.1 2D model with prescribed rigid body motion

As it was already mentioned, the end of this example is to assess the computation of the forces only induced by the fluid flow obtained with the proposed immersing technique. This problem enables us to perform a comparison with a well reported code based on the FEM/ALE technique. The geometrical setup and physical parameters try to mimic the conditions of the experimental case. It consists in a square cavity of side $a = 0.38$m, with a cylindrical body of diameter $D = 0.1$m, centered at $y_c = 0.21$m from the bottom, with an imposed harmonic horizontal motion of amplitude $A_i = 0.05$ m and frequency

$f_i = 0.9$ Hz at the center. Note that this example does not involve FSI since the body motion is imposed, but it helps to assess the computation of the fluid forces on the body using different numerical techniques.

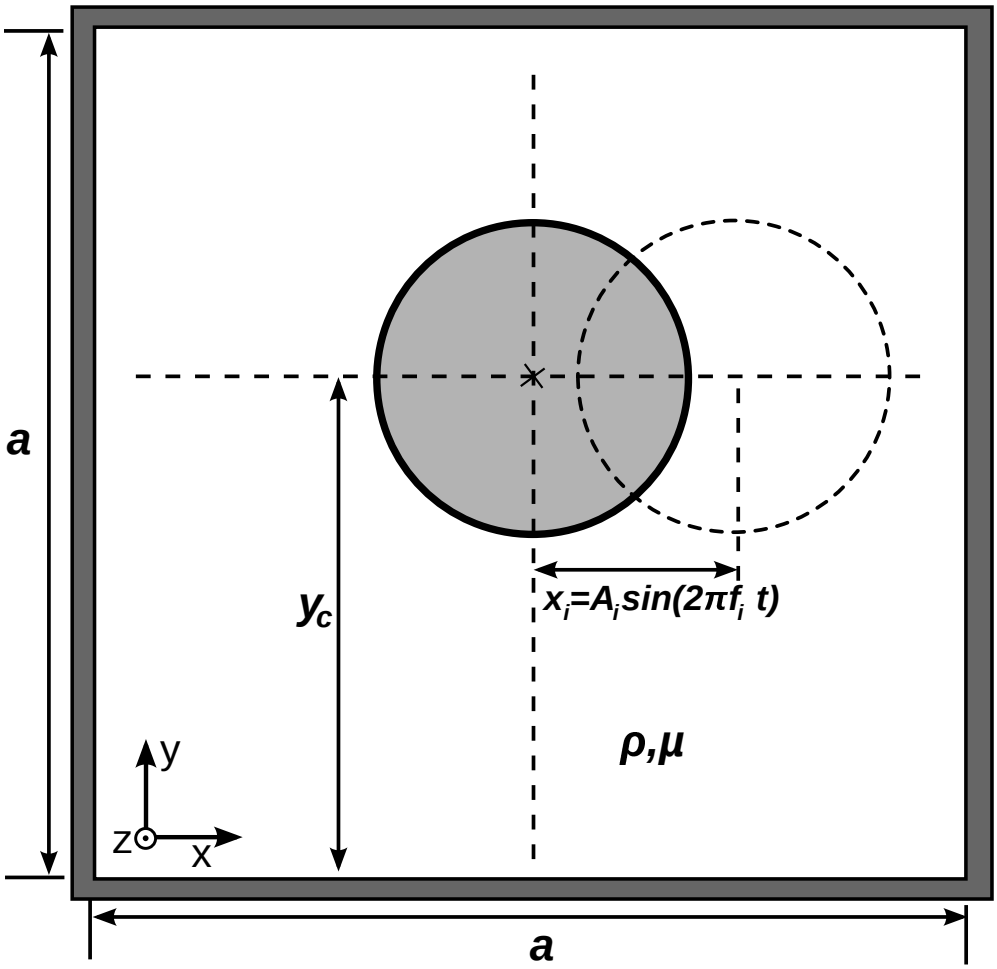


Figure 7: Oscillating cylinder example. Geometrical description.

Figure (7) sketches the computational domain and Figure (8) shows the boundary fitted meshes that have been used in the FEM/ALE analysis. The coarse mesh is composed of 16384 nodes with a mesh size of 1 mm in the normal direction on the body skin, and the fine mesh has 65536 nodes with a normal mesh size of 0.3 mm on the body skin.

For the FVM/IBM code a homogeneous discretization of 512 cells in each direction is adopted resulting in a 262144 total cells. The mesh size in this case is 0.75 mm, so it is comparable with the FEM meshes.

The material properties used in the analysis are: fluid density $\rho = 1000$ kg/m^3, two dynamic viscosities are considered $\mu = 0.05$ kg/ms and $\mu = 0.001$ kg/ms. This kind of flow is characterized in the literature [BDGO85] by the following non-dimensional numbers: the Keulegan-Carpenter number $K = U_{\max}/(f_i D) = 2\pi A_i/D$, where $U_{\max} = 2\pi f_i A_i$ is the maximum body velocity, and the Stokes number $\beta = D^2 \rho f_i/\mu$. In this example

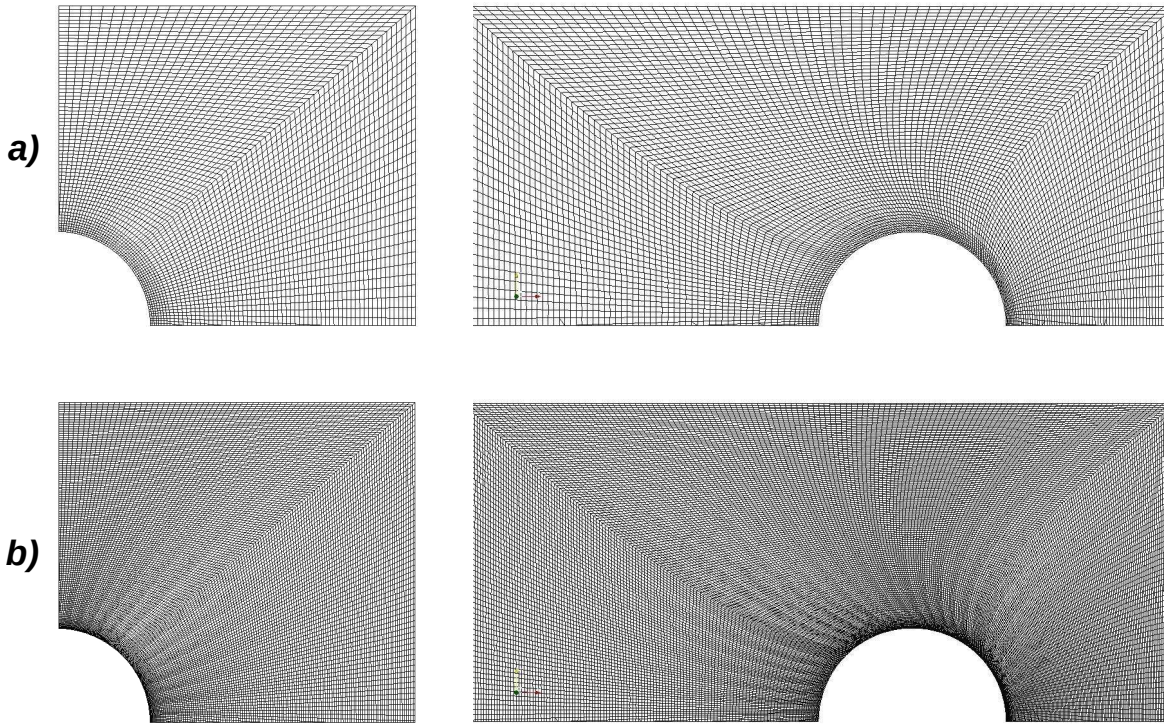


Figure 8: FEM meshes used in the computation. a) Coarse mesh, left: original (one quarter of the mesh is shown), and right: deformed mesh at maximum displacement (upper half mesh). b) Idem for the fine mesh.

constant Keulegan-Carpenter is considered $K = 3.14$ and β takes the values 180 and 9000 for the previous two different viscosities, and the related Reynolds number based on the maximum velocity ($\mathrm{Re} = U_{\max} D\rho/\mu = K\beta$) are Re = 565.5 and 28274 respectively. Note that even if the achieved Re is high, it must not be directly assimilated to a constant velocity flow at the same Re, since the flow conditions vary during a typical period. This could promote that full turbulence is not developed in short periods of time, for instance in [BDGO85] the boundary layer is assumed to be laminar for at least $\beta = 1665$. Taking these considerations into account no turbulence model was included. The satisfactory match between our numerical results and experimental observations reported in the literature [S$^+$76, Sar86, BDGO85] could be considered as an indirect evidence that this is an appropriate assumption considering the mesh refinement used.

For the FEM/ALE computation the time step used is $\Delta t = 0.002$ s, meanwhile for the FVM/IBM is $\Delta t = 0.0004$ s, and $N_{\mathrm{IBM}} = 1$ iterations were used for the IBM loop, and the tolerance for SIMPLE algorithm was $\mathrm{tol}_u = 10^{-4}$ (typically 3 iterations of the SIMPLE loop).

The computed horizontal fluid forces on the body are shown in Figure (9) for both β.

Horizontal imposed motion (x_i) is also included (scaled by 7.5 for readability) in order to visualize the time shift of the forces, which is an indicator of the drag. As it can be seen for $\beta = 180$ the forces computed are almost coincident. The forces for $\beta = 9000$ present differences, but in fact at such a Reynolds number, the temporal evolution is not deterministic. Vortices are shed randomly and then, fluctuations in the force are produced. Due to this fact, and with the end to produce a proper comparison, the work done by the fluid on the body is computed as:

$$W(t) = \int_0^T Fv \, \mathrm{d}t \tag{14}$$

with $v = 2\pi f_i A_i \cos(2\pi f_i t)$ is the horizontal imposed velocity and F is the horizontal fluid forces reported in Figure (9). This work is shown in Figure (10). From these plots we can compute the power dissipated by the fluid in a period $\bar{P}_n = (W((n+1)T) - W(nT))/T$. In Table (1) the average of $\bar{P}_n$ over a certain number of cycles is presented. Another possibility is to compute the average slope of $W(t)$ by linear regression over the same number of cycles. This value is also reported in the table, and it is shown in Figure (10) with black straight lines. Differences between such values represent the fluctuations in the forces. The average dissipation can be made non-dimensional in terms of the drag coefficient C_d. Assuming that the force is $F = -m_a\dot{v} - 1/2\rho_f v|v| \, C_d D$, and replacing this expression in (14) we obtain:

$$\begin{aligned} \bar{P} &= -\frac{1}{T}\int_0^T 1/2\rho_f D {A_i}^3 (2\pi f_i)^3 C_d |\cos(2\pi f t)|^3 \mathrm{d}t \\ &= \frac{16}{3}\pi^2 f_i^3 \rho_f D {A_i}^3 C_d \end{aligned} \tag{15}$$

from which the C_d coefficient can be computed in terms of the average dissipation. Note that the added mass force does not produce dissipation. Moreover for this problem the added mass force is much larger than the drag one and hence, it is difficult to disaggregate both components of the force. Then, the dissipation rate is a good indicator of the drag force, which in turn is the force component more difficult to obtain accurately in the computation. These C_d values are also reported in Table (1).

Average dissipation and drag coefficients at $\beta = 180$ obtained with FEM/ALE exhibit mesh independency. The corresponding values computed with FVM/IBM2 are close to them. The FEM/ALE results obtained at $\beta = 9000$ varies with mesh resolution (element size around the body skin), the FVM/IBM2 values are in between confirming that the technique predicts well the fluid forces on the body.

Although there is not experimental data for this specific configuration, there is available data for oscillating cylinders in an infinite domain. Even if it is not actually the same

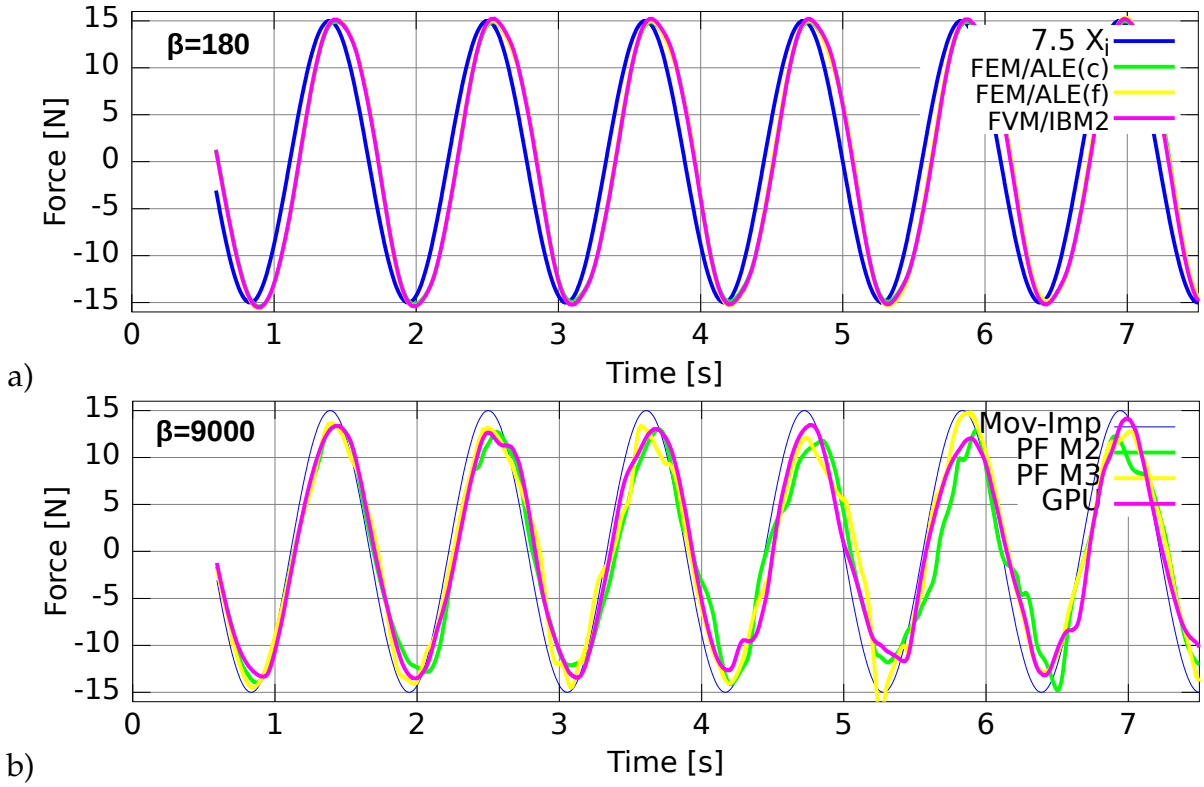


Figure 9: Horizontal fluid forces on the body computed for $K = 3.14$, and a) $\beta = 180$, b) $\beta = 9000$.

	$\beta = 180$				$\beta = 9000$			
	$\bar{P}$[1]	C_d [1]	$\bar{P}$[2]	C_d [2]	$\bar{P}$[1]	C_d [1]	$\bar{P}$[2]	C_d [2]
FVM/IBM2	0.770	1.604	0.770	1.604	0.445	0.926	0.445	0.926
FEM/ALE(c)	0.752	1.565	0.776	1.616	0.543	1.132	0.503	1.047
FEM/ALE(f)	0.760	1.583	0.781	1.626	0.361	0.752	0.406	0.845

Table 1: Average dissipation and drag coefficient, computed with: [1] average of cycle work, [2] work linear regression. (c) and (f) stands for coarse and fine meshes respectively.

configuration the values of C_d and C_a are comparable. Figure (11) shows the drag coefficient C_d computed using the FVM/IBM2 and FEM/ALE (fine mesh) versus Stokes number β together with experimental data from several sources [S+76, Sar86, BDGO85] for Keulegan-Carpenter number $K = 3.14$. The numerical predictions reasonable match the experiments. It can be seen that the numerical C_d values are slightly larger, probably because in the numerical test the flow is confined increasing shear stresses. In addition

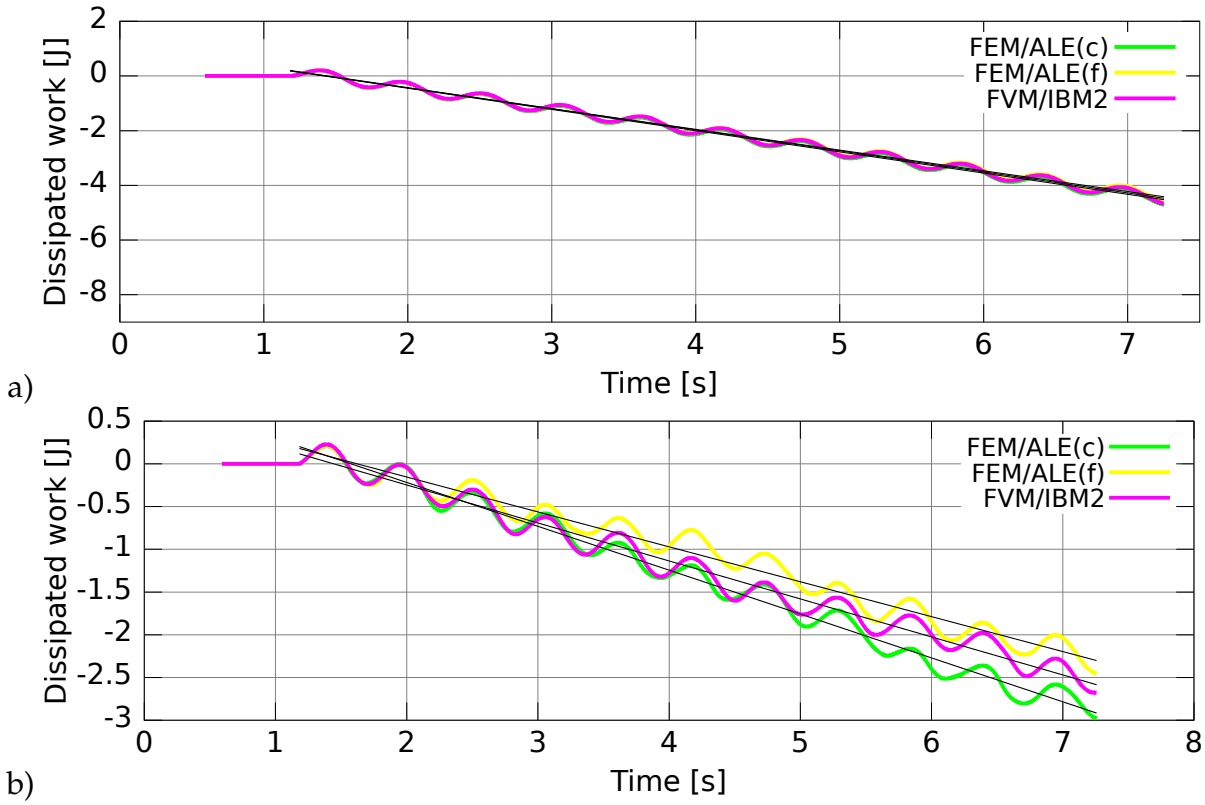


Figure 10: Work done by the fluid forces on the cylinder for $K = 3.14$, and a) $\beta = 180$, b) $\beta = 9000$. Straight lines are the linear regression used for the computation of $\bar{P}$ and C_d.

the computed added mass coefficient is $C_a = 0.995$ for $\beta = 9000$ and the total force coefficient [Sar86] results in $C_f = (0.375{C_d}^2 + \pi^4(1 + C_a)^2/(2K^2))^{0.5} = 4.478$, matching the experimental value 4.443 reported in that reference. From these results we can conclude that the model assumptions reasonably describe the physics of the present problem, as it was previously mentioned.

Colormaps of vorticity at several times for $\beta = 180$ are shown in Figure (12). The first row corresponds to $t = 0.8$ s from the start (i.e. $0.72\,T$, near the maximum displacement to the left). The remaining five rows are plotted $T/8$ apart from $6.5\,T$, representing a half cycle from the centered position to the extreme left position and back to the center. Note that for the first two columns computed with FEM/ALE, the body is shown in white because the mesh is boundary fitted. On the other hand, the third column, corresponding to FVM/IBM2 computations, the body has a predominantly blue color that represents the fictitious flow inside the body, which is mainly at rest, except for a thin layer close to the skin.

Similar snapshots for $\beta = 9000$ are shown in figure (13). In this case the snapshots can

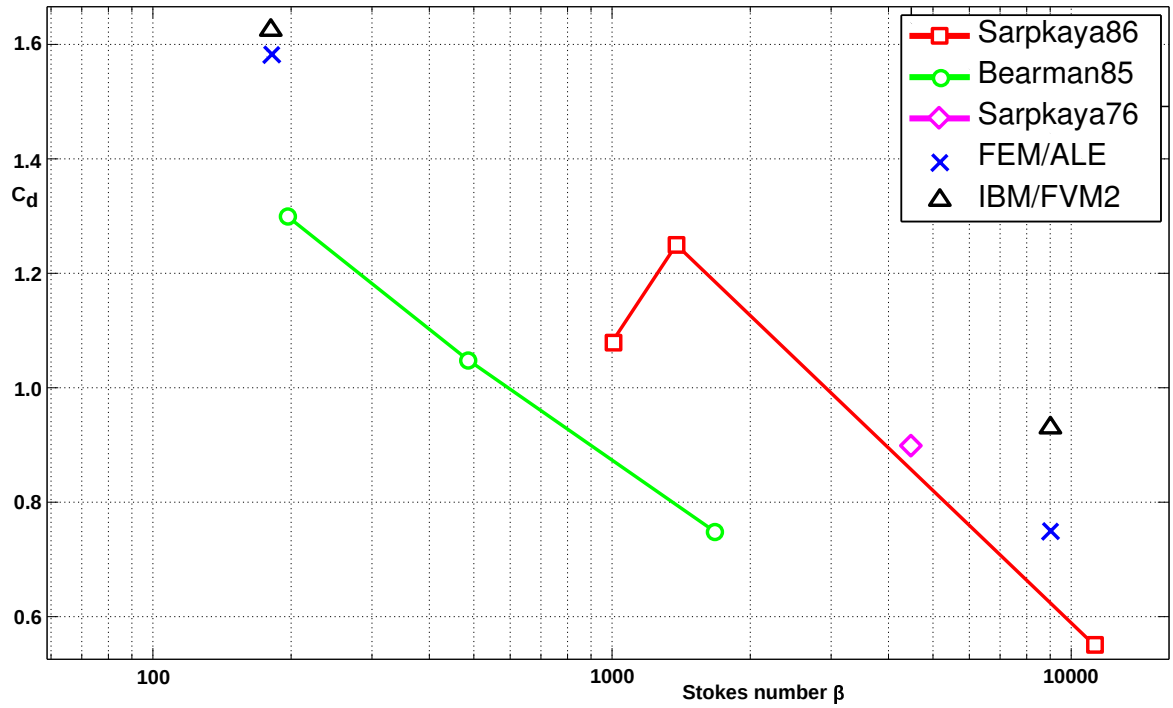


Figure 11: Drag coefficient C_d vs Stokes number β for Keulegan-Carpenter number $K = 3.14$. Numerical (present work) and experimental data from [S+76, Sar86, BDGO85]

not be directly compared because of the chaotic behavior of vortices at this high β, even between the two FEM/ALE meshes. However the qualitative aspect of the flow is similar and the quantitative values of force, dissipated power, and corresponding drag coefficients reported in Table (1) are coincident.

It is observed in these snapshots that the main mechanism of vortex generation is the shedding of vortex pairs, as it was reported in experimental works for this range of Keulegan-Carpenter number [LD02, LHL10].

6.2 Modeling the experiment

3D simulations able to describe the experiment presented in Section (2) are carried out using the proposed second order embedding technique. To evaluate the precision of this technique the results were compared with experimental data and with those obtained using the finite volume first order embedding scheme [SPD+13].

It is remarkable that in this case the tank motion is imposed to the model by using a non-inertial frame of reference attached to the tank, and so the effect of the movement is introduced as an inertial force, i.e. resulting in $\mathbf{f} = -ge_y - a_{\text{tank}}e_x$ (see Equation 3).

The motion imposed to the tank is harmonic with amplitude $A_{\text{tank}} = 0.02$ m and

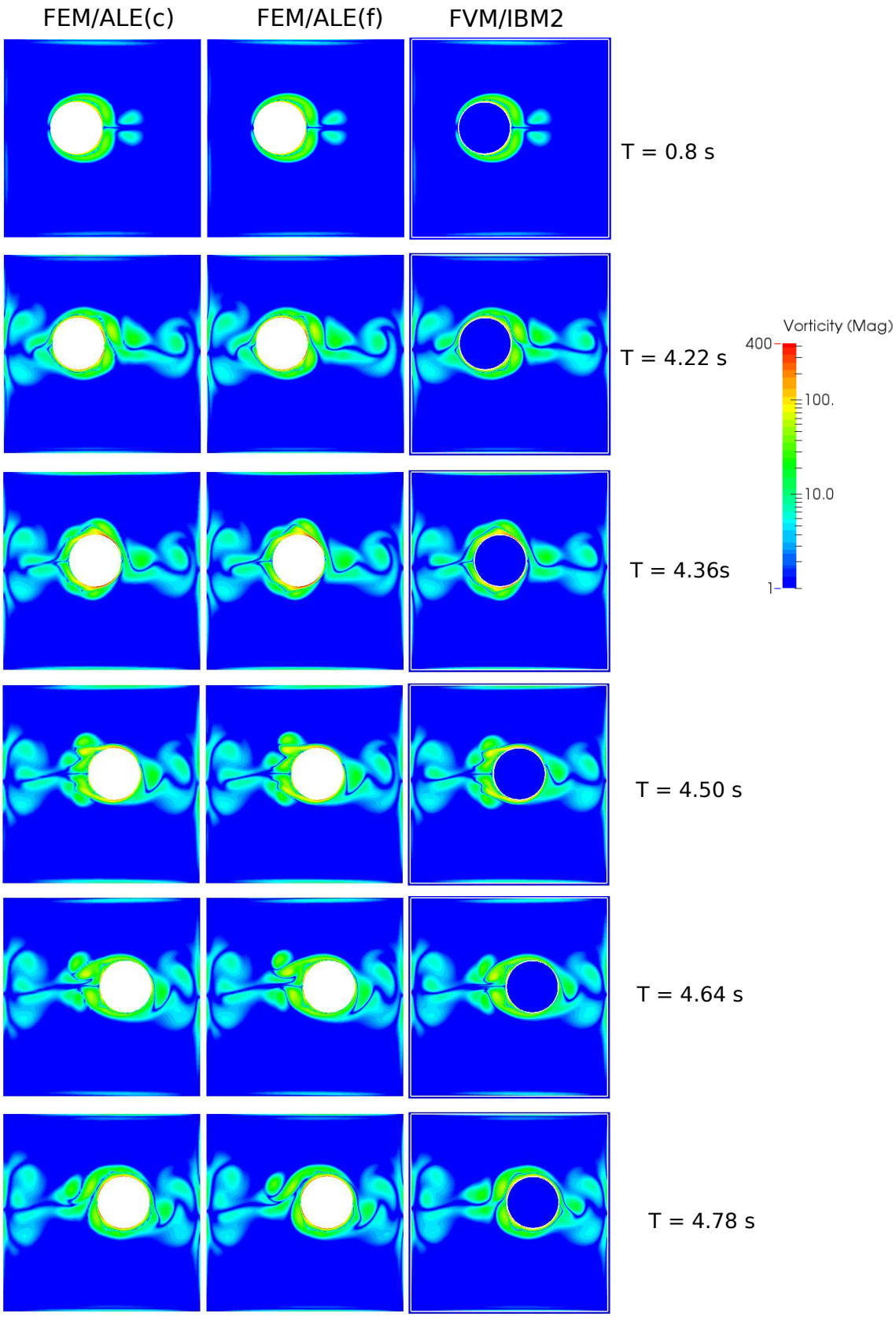


Figure 12: Colormaps of vorticity for the oscillating cylinder example at $\beta = 180$.

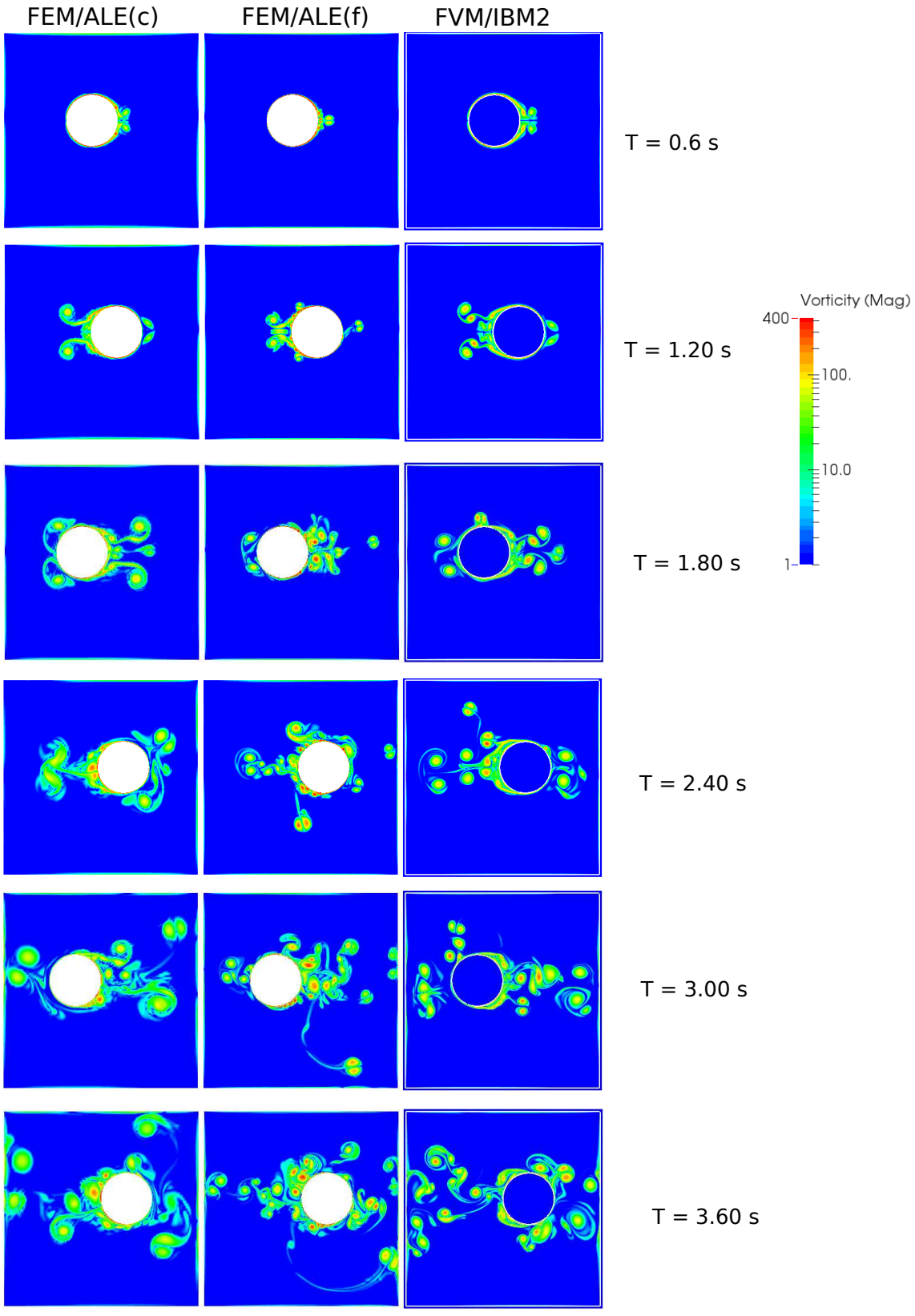


Figure 13: Colormaps of vorticity for the oscillating cylinder example at $\beta = 9000$.

frequencies in the range of 0.5 to 1.5 Hz.

A mesh convergence analysis is reported in Figure (14) including the predictions

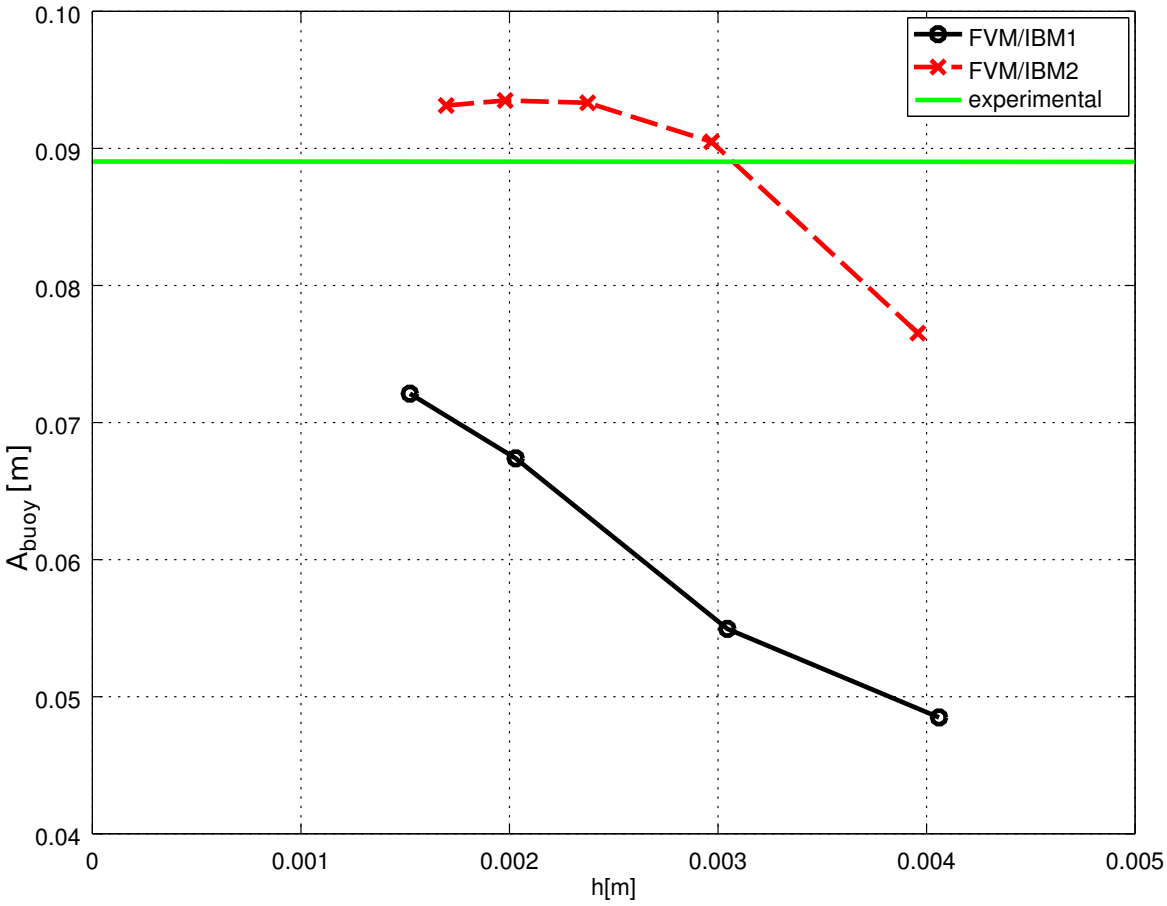


Figure 14: Mesh convergence analysis for first and second order FVM/IBM schemes at $f = 0.9$ Hz.

obtained for the amplitude at resonance frequency using both finite volume embedding schemes of first and second order. The resonant frequency condition is chosen because it is the more severe one.

The time step was chosen for FVM/IBM1 in order to have a CFL number lower than 0.2 being the critical value 0.5. The CFL for FVM/IBM2 was close to 0.1, being the critical value determined experimentally in the range 0.5-0.6. The tolerance for the solution of the Poisson equation in FVM/IBM1 was 10^{-6}. Meanwhile for the second order scheme presented in this work $N_{\text{IBM}} = 1$ and $N_{\text{simple}} = 3$, reaching typically an IBM convergence to $\epsilon_f = 10^{-3}$ and for the SIMPLE loop convergence to $\epsilon_u = 2 \times 10^{-3}$.

Linear convergence is observed for the first order (FVM/IBM1) scheme, while quadratic convergence is clearly obtained for the second order (FVM/IBM2) scheme. For the coarsest mesh (96 cells per side) an error of 45% with respect to the experimental value is obtained with FVM/IBM1, while an error of 14% is found with FVM/IBM2. For a finer mesh (160 cells per side) the errors are reduced to 14% and 5% respectively. These results confirm the good behavior of the proposed second order scheme.

A number of numerical simulations have been performed in order to test the influence on the results of the algorithmic parameters N_{IBM} and N_{simple}. The results are practically independent for $N_{\text{IBM}} \geq 3$. The parameter N_{simple} was tested for values 5, 10, and 15 showing almost independency for the finer meshes. For coarser meshes there is some dependence, but it does not degrade the quadratic convergence of the algorithm.

Figure (15) summarizes the results of amplitudes and phases obtained during time periodic regime at different imposed frequencies computed with the proposed second order FVM/IBM2 embedding technique in the finest mesh (192 cells per side, 7.08 Mcell total) showing an excellent agreement with experimental data. The amplitude at resonance, which is very sensitive to the drag force, and hence one of the most difficult parameters to reproduce, is 0.0931m. This value is 4.5% higher than the experimental one. One reason for this may be the fact that not all the DOFs of the system are taken into account. Generally speaking, in a mechanical system as more DOFs are added, greater is the possibility of the system to dissipate energy and then the amplitude would be smaller. Using the analytic reduced model presented in Section (4) these numerical results can be fitted to obtain $C_{d,\text{num}} = 0.23$. This value is very close to that one that fits the experimental data ($C_{d,\text{exp}} = 0.25$). The convergence parameters used have been $N_{\text{simple}} = 3$, $N_{\text{IBM}} = 1$, CFL$= 0.1$, $\Delta t = 2 \times 10^{-4}$ s.

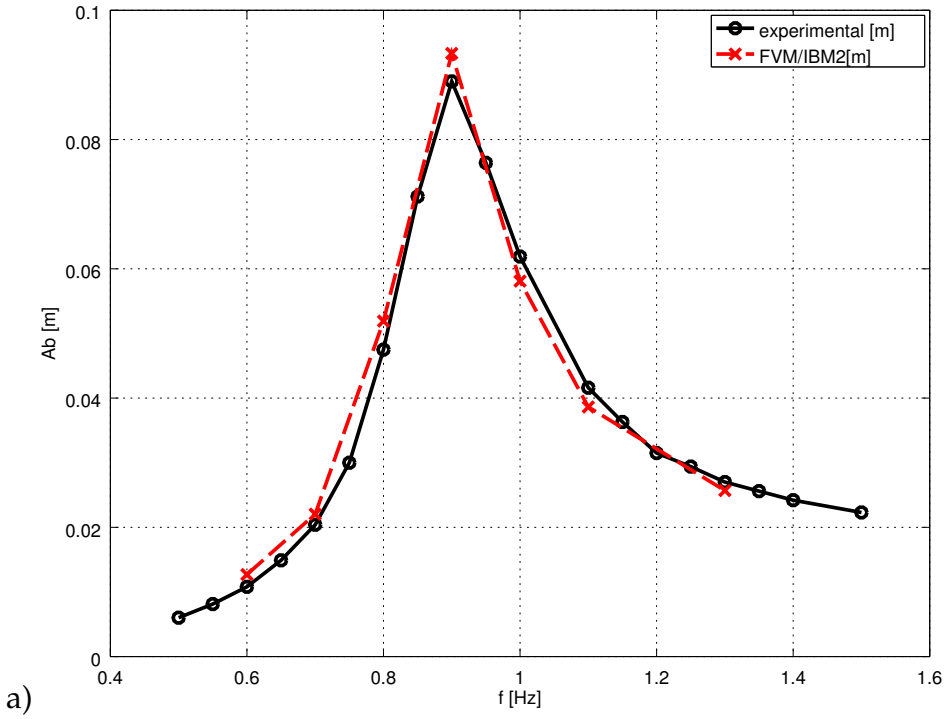
Several snapshots of vorticity isosurfaces for the simulation at $f = 0.9\,[\text{Hz}]$ are shown in Figure (16). The nine frames correspond to time positions $t = (9 + k/8)T$ with $0 \leq k \leq 8$ (approximately), that is the covering the whole 10-th cycle with intervals of $T/8$.

7 GPGPU implementation

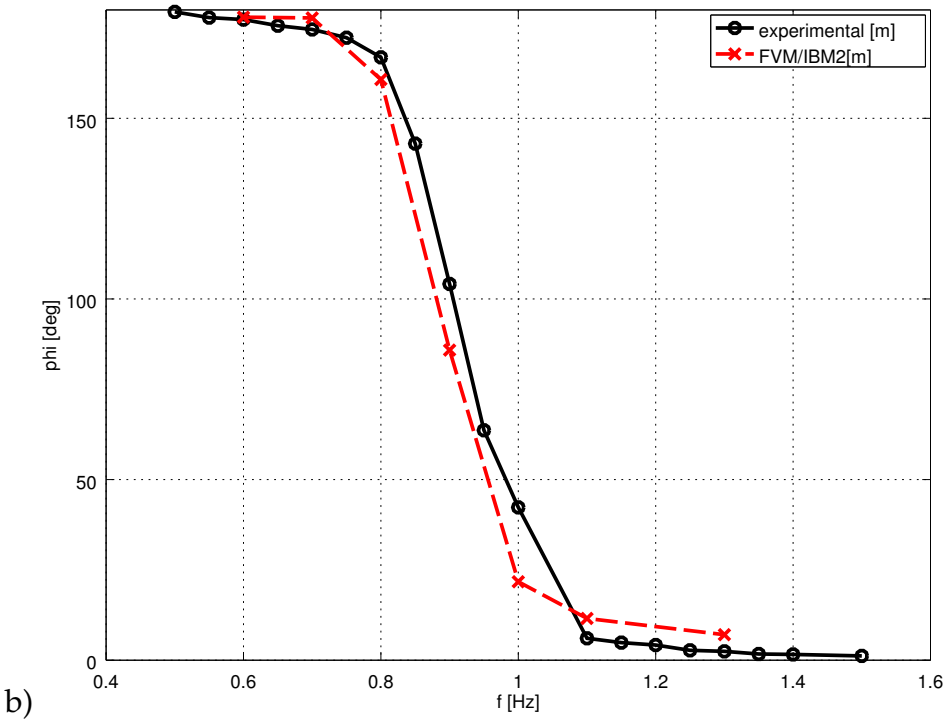
The main features of the implementation in GPGPU of the proposed algorithm are summarized in this section.

To improve the computational efficiency, the present GPGPU implementation encompasses the following aspects:

- The code is split in many small kernels. The following is a not exhaustive but includes the main kernels used in the code. (The lines refer to Algorithm 1).
  - Three kernels for the x, y, and z momentum equations (line 7).
  - Three kernels for the components of the compensating forces f' (line 9).



a)



b)

Figure 15: FVM/IBM2 vs. experimental results.

– Three kernels for the right hand side of the Poisson equation (line 15).

– One kernel for the FFT solution of the Poisson equation (line 15).

– Three kernels for the SIMPLE correction of velocities (line 17).

– One kernel for the SIMPLE correction of pressure (line 18).

– Three kernels for the computation of velocity gradients in order to compute the

a)

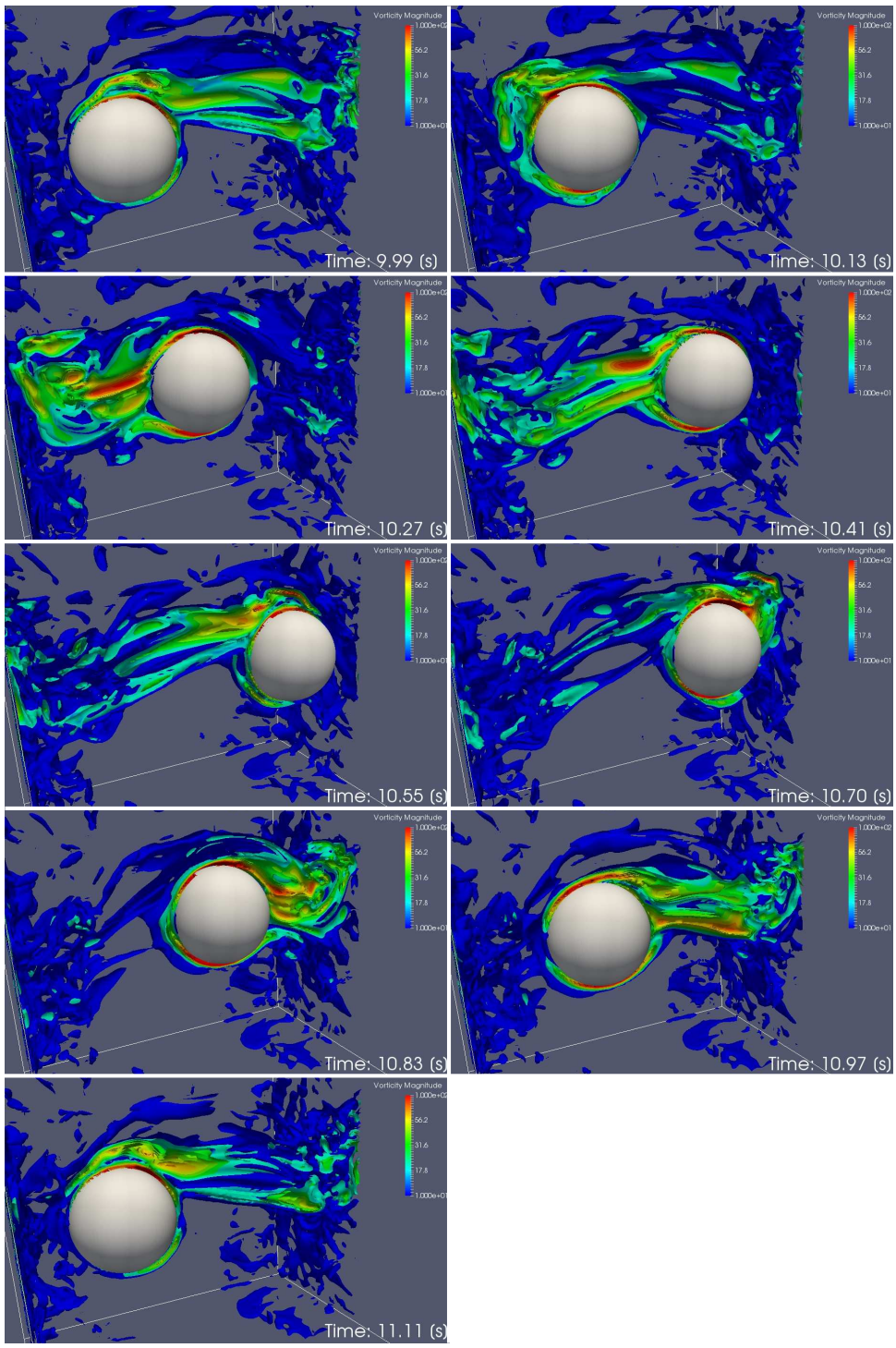


Figure 16: Isosurfaces of vorticity for results obtained with FVM/IBM2 from $t = 9T$ spaced at $T/8$ (approximately). Ten isosurfaces are shown for values equally spaced from $\omega = 10$ to $100\,\mathrm{s}^{-1}$, where ω is the norm of the vorticity vector.

viscous tractions (line 22).

  - One kernel for the update of the level set function (line 24).

- Global memory access (Gmem) is preferred over Shared memory (Smem).

- 3D CUDA blocks are used, with larger sizes aligned with the mesh direction along which the data is stored contiguously.
- In order to reduce the number of registers to the minimum some identical arithmetic operations are redundantly computed in all threads, instead of storing intermediate quantities in SM registers [CCLW11]. In practice this is done by replacing the intermediate variable by preprocessing macros.

With this improvements the code is speed up by a factor of 4x. Table (2) shows the relative computational times for each stage of the algorithm, on the NVIDIA K40c and GTX Titan Black cards, with meshes ranging from 160 to 224 cells per side, with 3 iteration for the SIMPLE loop and one iteration of the IBM loop.

Stage	IBM Loop	Poisson	Forces	Level Set	Other
Lines	6-13	15	22	24	
Time (%)	60%	35 %	4%	0.5%	0.5%

Table 2: Relative computing times for the different stages of the algorithm.

In cases where a complex geometry needs to be analyzed, the computational time related to solve the level-set equation with a PDE based algorithm can be estimated similar to one of the momentum kernels and, then, it could represent around 7% in the present context. Reinitialization of the level-set function will be needed, but estimating the frequency of reinitialization and its computing time the extra cost of the whole level-set computation can be estimated as an extra 20%. Notice that, the incidence of level-set computation on the global efficiency depends on the convergence parameters used for the IBM+SIMPLE algorithm. If tighter tolerances are enforced, more iterations of the IBM and SIMPLE loops are needed reducing the relative cost of the level-set computation. The same applies if some mechanism is used for extending the CFL range, for instance using a semi-Lagrangian approach [CSP+14].

7.1 Computing rates

The efficiency of the FVM/IBM2 algorithm was measured in the Nvidia Tesla K40c, and GTX Titan Black cards (CUDA version 6.5, Driver Version: 340.29). The K40c card has 2880

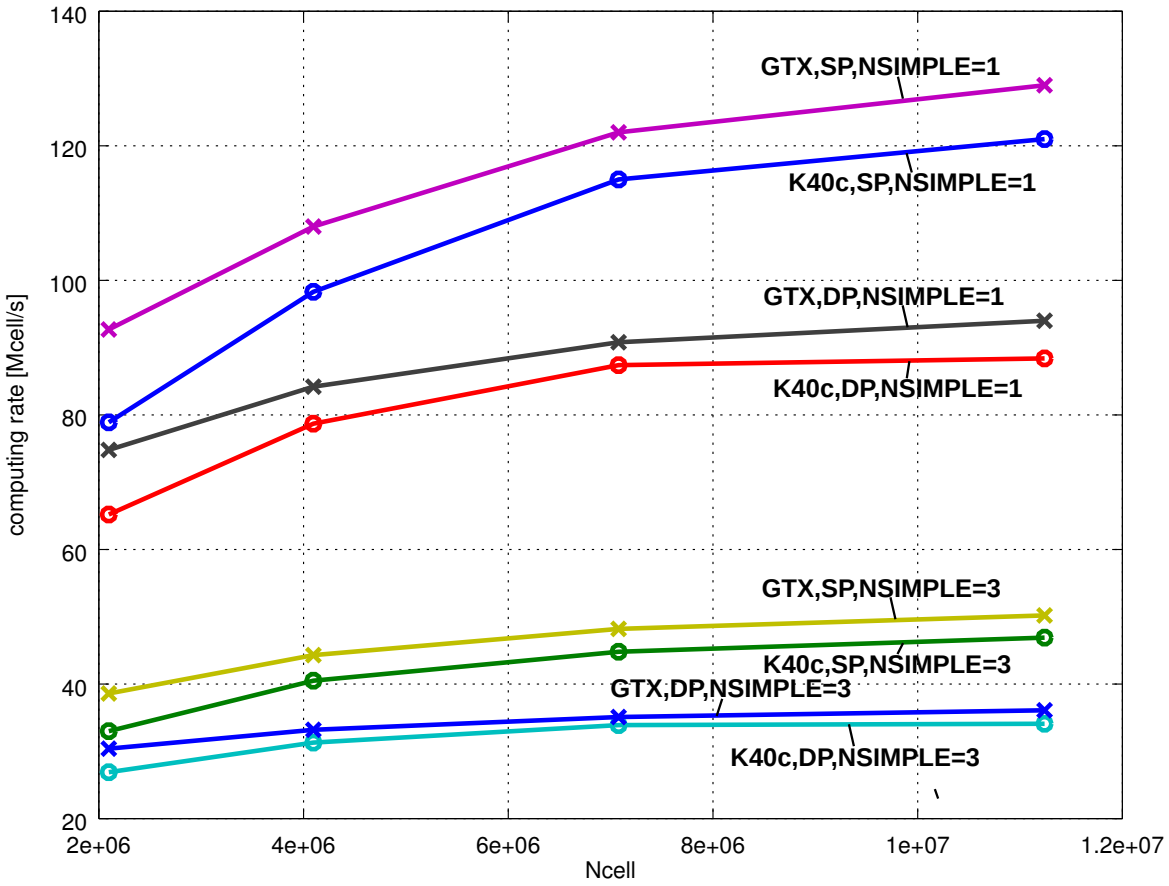


Figure 17: Computing rates on the Nvidia Tesla K40c, and GTX Titan Black.

CUDA cores @ 876 MHz, 12 GB ECC RAM, while the Titan Black has 2880 cores @ 980 MHz with 6GB RAM (no ECC). Figure (17) shows the computing rates [Mcell/s] as the number of cells processed per second for a whole time step. So this number is roughly inversely proportional to the number of IBM and SIMPLE iterations. In practice it has been found that $N_{\text{IBM}} = 1, N_{\text{simple}} = 3$ and CFL$= 0.1$ was accurate enough, and these are the parameters that where used in the examples. The computing rates are reported for meshes of 128, 160, 192, and 224 cells per side, using $N_{\text{IBM}} = 1$ and $N_{\text{simple}} = 1$ and 3, in single and double precision (SP/DP). Computing rates increase monotonically with mesh size, reaching 130 [Mcell/s] in the Titan Black for a mesh of $224^3 = 11.2$ Mcell with $N_{\text{simple}} = 1$ in SP and 94 [Mcell/s] in DP. The Titan Black outperforms the Tesla K40c both in SP and DP. If $N_{\text{simple}} = 3$ is used then the rates drop by a factor of roughly 3:1, i.e. 50.2 [Mcell/s] in SP and 36.1 [Mcell/s] in DP, both in the Titan Black.

To put these cell rates in context, they can be compared with the maximum rate attainable due to the memory bandwidth computed as $r_{\text{limit},b} = b/(mp)$ where b is the effective bandwidth between the GPGPU and the device memory, m is the number of scalars that must be accessed per cell in one iteration, and p is the precision in bytes.

Considering that in the present code $m = 66$ scalars are accessed, and $b = 234\,\mathrm{GB/s}$ for the K40c and $b = 228\,\mathrm{GB/s}$ for the GTX Titan Black, the corresponding maximum rates are limited by $r_{\mathrm{limit},b} = 886\,\mathrm{Mcell/s}$ and $r_{\mathrm{limit},b} = 863\,\mathrm{Mcell/s}$, for K40c and GTX Titan Black respectively in SP ($p = 4$). As the computing rates are well below these maximum bounds (15% or less) we conclude that the algorithm is not memory bound. On the other hand, the Nvidia profiler (nvprof) reports that the amount of floating point operations per cell is around 1320, then the code is performing 152 Gflop/s which is well below the raw computing power of the card (2.56 Tflop/s, not taking into account Mult-Add operations [Nvi10]), but is much higher than the Gflops/s reported in similar codes (see below). The analysis that was presented here is similar to the one presented in [LDB14], where a finite difference scheme over Cartesian structured meshes with constant step size is analyzed. That code uses artificial compressibility, first order staircase representation of the surfaces, but has the capability to do block structured refinement. The peak performance reported is 37 Mcell/s on a Tesla C2070 (1.3 Tflop/s SP). Based on an estimated number of flop per cell the estimated computing rate is 17.4 Gflop/s, one order of magnitude lower than the one reported in this work (but with a slower card). In [CCLW11] the authors report a computing rate of 50 Mcell/s (SP, tetrahedra) on a Nvidia Tesla 10 (1 Tflop/s SP), using an explicit compressible finite-volume code for unstructured meshes. This code is not directly comparable with ours; the following facts can be summarized. Computational cost on tetrahedra meshes must be scaled down by a factor of 5x, with respect to hexahedra meshes for roughly the same accuracy. Our code solves an FSI problem and solves implicitly for pressure, whilst their code is for compressible flows and fully explicit and so, as it is well known, fits better in the GPGPU architecture. The GTX Titan Black card is more recent and its raw computing power is around 4 to 5x times the one used in that references.

8 Conclusions

A formulation able to solve Fluid Structure Interaction based on Cartesian structured finite volume grids with embedded bodies has been presented. Particular emphasis has been done in the evaluation and well description of fluid forces on a rigid body. To this end, a second order numerical scheme has been developed based on a Rhie-Chow discretization for a colocated finite volume strategy, and a SIMPLE loop has been modified by including an operator to adjust the boundary conditions on the immersed body surface. The solver was specifically oriented to run in GPGPU architecture. Moreover, an experiment able to

detect the effects of drag and added mass forces has been designed. An sphere with positive buoyancy is attached to the bottom of a tank filled with water. The tank is subjected to controlled motion resulting in a system that acts as a pendulum. A Motion Capture algorithm has been implemented to extract the sphere position during motion from videos. The capabilities of the proposed formulation has been assessed by comparison of their numerical results with those computed using a robust FEM/ALE technique and a first order formulation previously reported in the literature. The results exhibit an excellent behavior when second order approximation is used. In particular, the simulations of the submerged buoy satisfactorily adjust the experimental data confirming that drag and mass added forces are well captured. The numerical model will be extended to include more DOFs in order to deal with a complete description of the problem.

9 Acknowledgment

The authors would like to thank Gilles Perrot and Raphaël Couturier of the FEMTO-ST Institute for their help on the code optimization on the GPGPU and also the Franche-Comté Meso-Centre.

The authors also thank the support given by research projects:

- Chilean Council for Scientific and Technological Research (CONICYT-FONDECYT 1130278);
- The Scientific Research Projects Management Department of the Vice Presidency of Research, Development and Innovation (DICYT-VRID) at Universidad de Santiago de Chile;
- Association of Universities: Montevideo Group (AUGM);
- Argentinean Council for Scientific Research (CONICET project PIP 112-20111-00978);
- Argentinean National Agency for Technological and Scientific Promotion, ANPCyT, (grants PICT-E-2014-0191, PICT-2014-0191);
- Universidad Nacional del Litoral, Argentina (grant CAI+D-501-201101-00233-LI);
- Santa Fe Science Technology and Innovation Agency (grant ASACTEI-010-18-2014); and

- European Research Council (ERC) Advanced Grant Real Time Computational Mechanics Techniques for Multi-Fluid Problems (REALTIME, Reference: ERC-2009-AdG).

The authors made extensive use of *Free Software* as GNU/Linux OS, GCC/G++ compilers, Octave, and *Open Source* software as VTK, ParaView, among many others.

10 References

[BDGO85] PW Bearman, MJ Downie, JMR Graham, and ED Obasaju. Forces on cylinders in viscous oscillatory flow at low Keulegan-Carpenter numbers. *Journal of Fluid Mechanics*, 154:337–356, 1985.

[BHK+11] Y. Bazilevs, M.C. Hsu, J. Kiendl, R. Wüchner, and K.U. Bletzinger. 3D simulation of wind turbine rotors at full scale. Part II: Fluid-structure interaction modeling with composite blades. *International Journal for Numerical Methods in Fluids*, 65(1-3):236–253, 2011.

[BSTL11] Joseph Baum, Orlando Soto, Fumiya Togashi, and Rainald Lohner. Numerical modeling of multi-phase, multi-material blast-structure interactions. In *41st AIAA Fluid Dynamics Conference and Exhibit*, 2011.

[CBSD14] Marcela Cruchaga, Laura Battaglia, Mario Storti, and Jorge D'Elía. Numerical modeling and experimental validation of free surface flow problems. *Archives of Computational Methods in Engineering*, pages 1–31, 2014.

[CCLM12] Andrew Corrigan, Fernando Camelli, Rainald Löhner, and Fernando Mut. Semi-automatic porting of a large-scale Fortran CFD code to GPUs. *International Journal for Numerical Methods in Fluids*, 69(2):314–331, 2012.

[CCLW11] Andrew Corrigan, Fernando F Camelli, Rainald Löhner, and John Wallin. Running unstructured grid-based CFD solvers on modern graphics hardware. *International Journal for Numerical Methods in Fluids*, 66(2):221–229, 2011.

[CMOS97] S Chen, B Merriman, S Osher, and P Smereka. A simple level set method for solving Stefan problems. *Journal of Computational Physics*, 135(1):8–29, 1997.

[CSP+14] Santiago D Costarelli, Mario A Storti, Rodrigo R Paz, Lisandro D Dalcin, and Sergio R Idelsohn. GPGPU implementation of the BFECC algorithm for pure advection equations. *Cluster Computing*, 17(2):243–254, 2014.

[DL03] T.F. Dupont and Y. Liu. Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. *Journal of Computational Physics*, 190(1):311–324, 2003.

[DL07] T.F. Dupont and Y. Liu. Back and forth error compensation and correction methods for semi-Lagrangian schemes with application to level set interface computations. *Mathematics of Computation*, 76(258):647–668, 2007.

[EFFM02] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational physics*, 183(1):83–116, 2002.

[FVOMY00] EA Fadlun, R Verzicco, P_ Orlandi, and J Mohd-Yusof. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of Computational Physics*, 161(1):35–60, 2000.

[GPS10] Luciano Garelli, Rodrigo R Paz, and Mario A Storti. Fluid–structure interaction study of the start-up of a rocket engine nozzle. *Computers and Fluids*, 39(7):1208–1218, 2010.

[HGC+14] C Hesch, AJ Gil, A Arranz Carreño, J Bonet, and P Betsch. A mortar approach for fluid–structure interaction problems: Immersed strategies for deformable and rigid bodies. *Computer Methods in Applied Mechanics and Engineering*, 278:853–882, 2014.

[HLOZ97] Thomas Y Hou, Zhilin Li, Stanley Osher, and Hongkai Zhao. A hybrid method for moving interface problems with application to the Hele–Shaw flow. *Journal of Computational Physics*, 134(2):236–252, 1997.

[HT06] Jaroslav Hron and Stefan Turek. *A monolithic FEM/multigrid solver for an ALE formulation of fluid-structure interaction with applications in biomechanics*. Springer, 2006.

[HZB12] Tao He, Dai Zhou, and Yan Bao. Combined interface boundary condition method for fluid-rigid body interaction. *Computer Methods in Applied Mechanics and Engineering*, 223-224(0):81 – 102, 2012.

[IMLO08] Sergio R Idelsohn, Julio Marti, A Limache, and Eugenio Oñate. Unified Lagrangian formulation for elastic solids and incompressible fluids:

application to fluid–structure interaction problems via the PFEM. *Computer Methods in Applied Mechanics and Engineering*, 197(19):1762–1776, 2008.

[IOPC06] SR Idelsohn, E Oñate, F Del Pin, and Nestor Calvo. Fluid–structure interaction using the particle finite element method. *Computer Methods in Applied Mechanics and Engineering*, 195(17):2100–2123, 2006.

[LD02] KM Lam and GQ Dai. Formation of vortex street and vortex pair from a circular cylinder oscillating in water. *Experimental Thermal and Fluid Science*, 26(8):901–915, 2002.

[LDB14] Rainald Löhner and Joseph D. Baum. On maximum achievable speeds for field solvers. *International Journal of Numerical Methods for Heat & Fluid Flow*, 24(7):1537–1544, 2014.

[LHL10] KM Lam, JC Hu, and P Liu. Vortex formation processes from an oscillating circular cylinder at high Keulegan–Carpenter numbers. *Physics of Fluids (1994-present)*, 22(1):015105, 2010.

[LNST07] Ezequiel J López, Norberto M Nigro, Mario A Storti, and Jorge A Toth. A minimal element distortion strategy for computational mesh dynamics. *International Journal for Numerical Methods in Engineering*, 69(9):1898–1929, 2007.

[MY98] J Mohd-Yusof. Development of immersed boundary methods for complex geometries. *Center for Turbulence Research, Annual Research Briefs*, pages 325–326, 1998.

[NLD07] Douglas Neill, Dean Livelybrooks, and Russell J Donnelly. A pendulum experiment on added mass and the principle of equivalence. *American Journal of Physics*, 75(3):226–229, 2007.

[Nvi10] Nvidia. Compute Unified Device Architecture (CUDA), 2010.

[OF06] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer Science & Business Media, 2006.

[PF01] Serge Piperno and Charbel Farhat. Partitioned procedures for the transient solution of coupled aeroelastic problems–Part II: energy transfer analysis and

three-dimensional applications. *Computer methods in applied mechanics and engineering*, 190(24):3147–3170, 2001.

[PMO+99] Danping Peng, Barry Merriman, Stanley Osher, Hongkai Zhao, and Myungjoo Kang. A PDE-based fast local level set method. *Journal of Computational Physics*, 155(2):410–438, 1999.

[PS72] Suhas V Patankar and D Brian Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787–1806, 1972.

[RC83] CM Rhie and WL Chow. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA journal*, 21(11):1525–1532, 1983.

[RC13] Andre L. Rossa and Alvaro L.G.A. Coutinho. Parallel adaptive simulation of gravity currents on the lock-exchange problem. *Computers and Fluids*, 88(0):782 – 794, 2013.

[RIO+11] Ricardo Rossi, S Idelsohn, E Oñate, J Cotela, and F Del Pin. Mejora de la solución fuertemente acoplada de problemas FSI mediante una aproximación de la matriz tangente de presión. *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería*, 27(3):165–179, 2011.

[RMSF11] Avi Robinson-Mosher, Craig Schroeder, and Ronald Fedkiw. A symmetric positive definite formulation for monolithic fluid structure interaction. *Journal of Computational Physics*, 230(4):1547–1566, 2011.

[S+76] Turgut Sarpkaya et al. In-line and transverse forces, on cylinders in oscillatory flow at high reynolds numbers. In *Offshore Technology Conference*. Offshore Technology Conference, 1976.

[Sar86] Turgut Sarpkaya. Force on a circular cylinder in viscous oscillatory flow at low keulegan—carpenter numbers. *Journal of Fluid Mechanics*, 165:61–71, 1986.

[Set96] J.A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591, 1996.

[SGP12] Mario A Storti, Luciano Garelli, and Rodrigo R Paz. A finite element formulation satisfying the discrete geometric conservation law based on

averaged Jacobians. *International Journal for Numerical Methods in Fluids*, 69(12):1872–1890, 2012.

[SNPD09] Mario A Storti, Norberto M Nigro, Rodrigo R Paz, and Lisandro D Dalcín. Strong coupling strategy for fluid–structure interaction problems in supersonic regime via fixed point iteration. *Journal of Sound and Vibration*, 320(4):859–877, 2009.

[SPD+13] Mario A Storti, Rodrigo R Paz, Lisandro D Dalcin, Santiago D Costarelli, and Sergio R Idelsohn. A FFT preconditioning technique for the solution of incompressible flow on GPUs. *Computers and Fluids*, 74:44–57, 2013.

[TBT12] Kenji Takizawa, Yuri Bazilevs, and Tayfun E Tezduyar. Space–time and ALE-VMS techniques for patient-specific cardiovascular fluid–structure interaction modeling. *Archives of Computational Methods in Engineering*, 19(2):171–225, 2012.

[Tez91] Tayfun E Tezduyar. Stabilized finite element formulations for incompressible flow computations. *Advances in Applied Mechanics*, 28:1–44, 1991.

[TT11] Kenji Takizawa and Tayfun E Tezduyar. Multiscale space–time fluid–structure interaction techniques. *Computational Mechanics*, 48(3):247–267, 2011.

[TT12] Kenji Takizawa and Tayfun E Tezduyar. Computational methods for parachute fluid–structure interactions. *Archives of computational methods in engineering*, 19(1):125–169, 2012.

[VHRS14] Mariano Vázquez, Guillaume Houzeaux, Félix Rubio, and Christian Simarro. Alya multiphysics simulations on Intel's Xeon Phi accelerators. In *High Performance Computing*, pages 248–254. Springer, 2014.

[VM07] Henk Kaarle Versteeg and Weeratunge Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education, 2007.

[WL04] Xiaodong Wang and Wing Kam Liu. Extended immersed boundary method using FEM and RKPM. *Computer Methods in Applied Mechanics and Engineering*, 193(12-14):1305 – 1321, 2004. Meshfree Methods: Recent Advances and New Applications.

Capítulo 8

Glosario y listado de símbolos más importantes

Glosario

BFECC	Compensación y corrección de errores hacia atrás y adelante
CA	Autómata celular
CFD	Dinámica de fluidos computacional
CG	Gradientes conjugados
CPU	Unidad central de procesamiento
CUDA	Arquitectura unificada de dispositivo de cómputo
DOF	Grados de libertad
DP	Doble precisión
FEM	Método de elementos finitos
FFT	Transformada rápida de Fourier
FLOPS	Operaciones de punto flotante por segundo
FSI	Interacción fluido estructura
FVM	Método de volúmenes finitos

GPGPU	Unidad de procesamiento gráfico de propósito general
GPU	Unidad de procesamiento gráfico
HPC	Computación de alto rendimiento
IBM	Método de fronteras embebidas
IB	Fronteras embebidas
IFFT	Transformada rápida de Fourier inversa
LS	Curvas de nivel
ODE	Ecuación diferencial ordinaria
PCG	Gradientes conjugados precondicionados
PFEM	Método de elementos finitos con partículas
PIM	Método de la potencia iterada
SIMPLE	Método semi-implícito para ecuaciones enlazadas por la presión
SP	Simple precisión
TVD	Disminución total de variación
VG	Generadores de vórtices

Símbolos

α_u	Parámetro de relajación
A_{tank}	Amplitud del movimiento armónico del tanque
β	Coeficiente de expansión volumétrica (y número de Stokes)
$\Re_{\mathbf{u}}$	Residuo de las ecuaciones de cantidad de movimiento
ΔV	Volumen de celda de discretización
ΔT	Salto de temperatura
Δt	Paso de tiempo
δ_*	Coeficiente adimensional de la ecuación explícita de cantidad de movimiento

$\dot{m}$	Flujo másico
$\epsilon(\mathbf{u})$	Tensor velocidad de deformación
ϵ^*_{atol}	Tolerancia absoluta
ϵ^*_{rtol}	Tolerancia relativa
κ	Conductividad térmica
λ	Factor de decaimiento exponencial de T en la dirección principal
μ	Viscosidad dinámica
ν	Viscosidad cinemática
ϕ	Fase del movimiento armónico del tanque
$\Phi(r)$	Función delimitadora (TVD)
Pr	Número de Prandtl
ψ	Función de curvas de nivel
Ra	Número de Rayleigh
$\Re$	Número de Reynolds
$\Re^*_p$	Contribución a la ecuación de Poisson para la presión de p^*
$\Re^*_u$	Contribución a la ecuación de Poisson para la presión de u^*
$\Re^n_u$	Contribución a la ecuación de Poisson para la presión de u^n
ρ	Densidad (constante)
ρ_0	Densidad de referencia a T_0
ρ_s	Densidad de la boya de silicona
CFL	Número de Courant-Friedrichs-Levy
Fo	Número de Fourier
$\tilde{S}^u_*$	Término fuente con aportes de fuerzas de cuerpo y correcciones diferidas
$\tilde{A}^u_*$	Coeficiente modificado de la ecuación de cantidad de movimiento

$\tilde{p}$	Altura gravimétrica
$\tilde{\mathbf{u}}$	Campo de velocidad interpolado (IB)
$\mathbf{A}$	Matriz ensamblada de la ecuación de Poisson para la presión
$\mathbf{D}$	Matriz diagonal con los valores propios de $\mathbf{A}$
$\mathbf{f}$	Fuerza de cuerpo y fuerza de compensación IB
$\mathbf{O}$	Matriz ensamblada de la transformada de Fourier
$\mathbf{u}$	Campo de velocidad (u, v, w)
$\mathbf{x}$	Punto en $\mathbb{R}^3$ (x, y, z)
A_*^u	Coeficiente de la ecuación de cantidad de movimiento
a_*^C	Coeficiente de la ecuación de cantidad de movimiento, término de convección
a_*^D	Coeficiente de la ecuación de cantidad de movimiento, término de difusión
c_p	Coeficiente de calor específico a presión constante
C_a	Coeficiente de masa agregada de la boya
C_d	Coeficiente de fricción de la boya
D	Diámetro de la boya
e	Error del BFECC
F	Campo escalar
f	Fuerza de cuerpo (continuo) y frecuencia del movimiento armónico del tanque
f_n	Frecuencia de resonancia de la boya
g	Constante de gravedad $9.81\, m/s^2$
K	Número de Keulegan-Carpenter
L	Longitud desde el punto de anclaje al centroide de la boya y longitud principal de la caja de la delta
$L(\cdot, \cdot)$	Operador

L_c	Longitud de cuerda de la delta
m_a	Masa agregada de la boya
m_{fl}	Masa del fluido desplazado por la boya
m_s	Masa de la boya
p	Campo de presión
$p^{'}$	Corrección del campo de presión
p^{*}	Campo de presión iterado de SIMPLE
p^{n+1}	Campo de presión a tiempo $(n+1) * \Delta t$
Q	Término fuente (compensación) de IB
S_u^{dc}	Término de corrección diferida (TVD)
S_u^{force}	Término de fuerzas de cuerpo
S_u^{ibm}	Término de compensación IB
S_u	Término fuente
T	Campo de temperatura
T_0	Temperatura de referencia
$u^{'}$	Corrección del campo de velocidad en dirección x
u^{*}	Campo de velocidad en dirección x iterado de SIMPLE
u^{n+1}	Campo de velocidad en dirección x a tiempo $(n+1) * \Delta t$
u^{n}	Campo de velocidad en dirección x a tiempo $n * \Delta t$
u_a^b	Campo de velocidad en dirección x evaluado en el nodo a a tiempo b

Capítulo 9

Bibliografía

[BC92] David Bleecker and George Csordas. *Basic partial differential equations*. CRC Press, 1992.

[BDGO85] PW Bearman, MJ Downie, JMR Graham, and ED Obasaju. Forces on cylinders in viscous oscillatory flow at low Keulegan-Carpenter numbers. *Journal of Fluid Mechanics*, 154:337–356, 1985.

[BF01] Richard L Burden and J Douglas Faires. Numerical analysis (7th). *Prindle Weber and Schmidt, Boston*, 2001.

[BL11] Achi Brandt and Oren E Livne. *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics, Revised Edition*. SIAM, 2011.

[BM82] JR Bates and A McDonald. Multiply-upstream, semi-lagrangian advective schemes: analysis and application to a multi-level primitive equation model. *Monthly Weather Review*, 110(12):1831–1842, 1982.

[CCLW11] Andrew Corrigan, Fernando F Camelli, Rainald Löhner, and John Wallin. Running unstructured grid-based CFD solvers on modern graphics hardware. *International Journal for Numerical Methods in Fluids*, 66(2):221–229, 2011.

[CD98] Shiyi Chen and Gary D Doolen. Lattice boltzmann method for fluid flows. *Annual review of fluid mechanics*, 30(1):329–364, 1998.

[Cho99] Seok Ki Choi. Note on the use of momentum interpolation method for unsteady flows. *Numerical Heat Transfer: Part A: Applications*, 36(5):545–550, 1999.

[CSP+14] Santiago D Costarelli, Mario A Storti, Rodrigo R Paz, Lisandro D Dalcin, and Sergio R Idelsohn. Gpgpu implementation of the bfecc algorithm for pure advection equations. *Cluster Computing*, 17(2):243–254, 2014.

[EFFM02] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational physics*, 183(1):83–116, 2002.

[ELF05] Douglas Enright, Frank Losasso, and Ronald Fedkiw. A fast and accurate semi-lagrangian particle level set method. *Computers & structures*, 83(6):479–490, 2005.

[FP12] Joel H Ferziger and Milovan Peric. *Computational methods for fluid dynamics*. Springer Science & Business Media, 2012.

[FVOMY00] EA Fadlun, R Verzicco, P_ Orlandi, and J Mohd-Yusof. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of Computational Physics*, 161(1):35–60, 2000.

[GJ97] MC Gentry and AM Jacobi. Heat transfer enhancement by delta-wing vortex generators on a flat plate: vortex interactions with the boundary layer. *Experimental Thermal and Fluid Science*, 14(3):231–242, 1997.

[GRD+16] Juan M Gimenez, Damián E Ramajo, Santiago Márquez Damián, Norberto M Nigro, and Sergio R Idelsohn. An assessment of the potential of pfem-2 for solving long real-time industrial applications. *Computational Particle Mechanics*, pages 1–17, 2016.

[KLLR07] ByungMoon Kim, Yingjie Liu, Ignacio Llamas, and Jarek Rossignac. Advections with significantly reduced dissipation and diffusion. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):135–144, 2007.

[LCWW14] Rainald Löhner, Andrew Corrigan, Karl-Robert Wichmann, and Wolfgang Wall. Comparison of lattice-boltzmann and finite difference solvers. 2014.

[MCPN08] Jeroen Molemaker, Jonathan M Cohen, Sanjit Patel, and Jonyong Noh. Low viscosity flow simulations for animation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 9–18. Eurographics Association, 2008.

[Men12] Jure Mencinger. *An Alternative Finite Volume Discretization of Body Force Field on Collocated Grid*. INTECH Open Access Publisher, 2012.

[MI05] Rajat Mittal and Gianluca Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

[MKLU05] S Marella, SLHH Krishnan, H Liu, and HS Udaykumar. Sharp interface cartesian grid method i: an easily implemented technique for 3d moving boundary computations. *Journal of Computational Physics*, 210(1):1–31, 2005.

[MY97] J Mohd-Yusof. Combined immersed-boundary/b-spline methods for simulations of ow in complex geometries. *Annual Research Briefs. NASA Ames Research Center= Stanford University Center of Turbulence Research: Stanford*, pages 317–327, 1997.

[Nvi10] Nvidia. Compute Unified Device Architecture (CUDA), 2010.

[OF06] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer Science & Business Media, 2006.

[Pas11] Antonio Pascau. Cell face velocity alternatives in a structured colocated grid for the unsteady navier–stokes equations. *International Journal for Numerical Methods in Fluids*, 65(7):812–833, 2011.

[Pes73] Charles S Peskin. Flow patterns around heart valves: a digital computer method for solving the equations of motion. *IEEE Transactions on Biomedical Engineering*, (4):316–317, 1973.

[PWR11] Merle Potter, David Wiggert, and Bassem Ramadan. *Mechanics of Fluids SI Version*. Nelson Education, 2011.

[S+76] Turgut Sarpkaya et al. In-line and transverse forces, on cylinders in oscillatory flow at high reynolds numbers. In *Offshore Technology Conference*. Offshore Technology Conference, 1976.

[S+94] Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[Sar86] Turgut Sarpkaya. Force on a circular cylinder in viscous oscillatory flow at low keulegan—carpenter numbers. *Journal of Fluid Mechanics*, 165:61–71, 1986.

[SC91] Andrew Staniforth and Jean Côté. Semi-lagrangian integration schemes for atmospheric models-a review. *Monthly weather review*, 119(9):2206–2223, 1991.

[SFK+08] Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. An unconditionally stable maccormack method. *Journal of Scientific Computing*, 35(2-3):350–371, 2008.

[SPD+13] Mario A Storti, Rodrigo R Paz, Lisandro D Dalcin, Santiago D Costarelli, and Sergio R Idelsohn. A fft preconditioning technique for the solution of incompressible flow on gpus. *Computers & Fluids*, 74:44–57, 2013.

[TF03] Yu-Heng Tseng and Joel H Ferziger. A ghost-cell immersed boundary method for flow in complex geometry. *Journal of computational physics*, 192(2):593–623, 2003.

[VM07] Henk Kaarle Versteeg and Weeratunge Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education, 2007.

[YWH85] HC Yee, RF Warming, and A Harten. Implicit total variation diminishing (tvd) schemes for steady-state calculations. *Journal of Computational Physics*, 57(3):327–360, 1985.

Doctorado en Ingeniería
mención Mecánica Computacional

Título de la obra:

Algoritmos de alta performance en Unidades de Procesamiento Gráfico (GPU) aplicados a la Dinámica de Fluidos Computacional

Autor: Santiago Daniel Costarelli

Lugar: Santa Fe, Argentina

Palabras Claves:

CUDA, Navier-Stokes, CFD
IBM, FVM, OpenFOAM.