Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXIII, págs. 2419-2438 (artículo completo)
Graciela Bertolino, Mariano Cantero, Mario Storti y Federico Teruel (Eds.)
San Carlos de Bariloche, 23-26 Setiembre 2014

SIMULACIONES PSEUDO-ESPECTRALES DE ALTO DESEMPEÑO DE CORRIENTES DE GRAVEDAD

Jorge S. Salinas[a,b,c], **Mariano I. Cantero**[a,b,c] **y Enzo Dari**[a,b,c]

[a]*Instituto Balseiro, San Carlos de Bariloche, Río Negro, Argentina.*

[b]*Consejo Nacional de Investigaciones Científicas y Técnicas, San Carlos de Bariloche, Río Negro, Argentina.*

[c]*Centro Atómico Bariloche, Comisión Nacional de Energía Atómica, San Carlos de Bariloche, Río Negro, Argentina.*

Palabras Clave: Corrientes de gravedad, Simulación directa de turbulencia, DNS, MPI, PETSC

Resumen. Las corrientes de gravedad están presentes en la naturaleza en escalas geofísicas, y son uno de los principales mecanismos para la transferencia de energía, masa y momento en la atmósfera y el océano. El fuerte comportamiento no lineal de estas corrientes genera una amplia gama de escalas temporales y espaciales que deben ser capturadas en su totalidad para predecir de forma precisa la dinámica del flujo. Para la simulación de estos flujos empleamos un código pseudo-espectral que utiliza expansiones de Fourier en dos direcciones y de Chebyshev en la tercera. Este código se encontraba paralelizado con OpenMP para sistemas de memoria compartida. Aunque en las últimas décadas el avance tecnológico en la industria de la computación ha sido exponencial, las simulaciones directas de turbulencia (DNS) de flujos a números de Reynolds reales todavía son imposibles. No sólo estamos limitados por la capacidad de cálculo de las computadoras de hoy en día, sino que además la capacidad de almacenamiento de memoria es una variable fundamental que determina la viabilidad de una simulación directa de turbulencia. Con el fin de superar esta barrera tecnológica en nuestra investigación de los flujos turbulentos, en este trabajo se implementó la paralelización del código de pseudo-espectral bajo el paradigma de memoria distribuida. Para esto utilizamos el estándar "MPI" (Message Passing Interface) junto con el paquete de estructuras de datos y rutinas "PETSc". Se presenta una detallada descripción del código, las principales rutinas paralelizadas y un análisis de rendimiento del nuevo código.

1. INTRODUCCIÓN

Las corrientes de gravedad son flujos generados por gradientes de presión horizontales resultantes del efecto de la gravedad sobre fluidos de diferente densidad. Estos flujos se manifiestan como una corriente horizontal de un fluido liviano por encima de un fluido pesado a lo largo de un contorno superior, o como una corriente de fluido pesado debajo de un fluido liviano sobre una superficie inferior, y puede ser producida por una pequeña diferencia de densidad entre los dos fluidos (García, 1992).

En los últimos años, debido al avance en la tecnología informática, se han podido realizar estudios de corrientes de gravedad mediante simulaciones numéricas. Como en la naturaleza estos flujos son altamente turbulentos, el espectro de escalas temporales y espaciales que presentan en casos reales no pueden ser capturados por las simulaciones actuales. Sin embargo, se han llevado a cabo estudios en base a simulaciones numéricas bi- y tridimensionales de alta resolución a escalas de laboratorio con el objetivo de explorar la dinámica de las corrientes de gravedad (Droegemeier y Wilhelmson, 1986; Terez y Knio, 1998b,a; Härtel et al., 1997, 1999, 2000b,a; Necker et al., 2002). Se han realizado simulaciones tanto en configuraciones planas (Härtel et al., 2000b; Cantero et al., 2008, 2007, 2006, 2009a,b) como cilíndricas (Cantero et al., 2007, 2006).

Uno de los métodos numéricos más utilizados para el estudio de las corrientes de gravedad son las simulaciones directas de turbulencia utilizando métodos pseudo-espectrales. En nuestro código se emplean expansiones de Fourier en las dos direcciones horizontales y expansiones de Chebyshev en la dirección vertical. Este código estaba implementado con el paradigma de memoria compartida mediante la interfaz de programación de aplicaciones *"OpenMP"*, por lo que las simulaciones realizadas con el mismo deben ser en una sola computadora.

En un intento de superar esta barrera tecnológica en la investigación de las corrientes de gravedad, en este trabajo abordamos la implementación de un código pseudo-espectral con el paradigma de memoria distribuida empleando el estándar *"MPI"* ("Message Passing Interface") junto con el paquete de estructuras de datos y rutinas *"PETSc"*. Se presenta una descripción detallada del código, las principales rutinas paralelizadas y el análisis de rendimiento del mismo mediante cinco pruebas en un cluster del departamento de *Medicina Asistida por Computación Científica* del *Instituto de Ciencia y Tecnología de Brasil* (INCT-MACC).

2. FORMULACIÓN MATEMÁTICA

La configuración física de las corrientes de gravedad se muestra en la figura 1. En el instante inicial de la simulación, la región con fluido de densidad ρ_1 (la región gris sombreada en la figura 1) se encuentra separada del fluido ambiente de densidad ρ_0 ($\rho_1 > \rho_0$). El fluido más denso ocupa una región de ancho $2x_0$ a lo largo de la dirección del flujo (x) y se extiende sobre toda la altura H y el ancho del canal L_y. El sistema se encuentra rotando alrededor del eje vertical z, que pasa por el centro del dominio, a una velocidad Ω_z.

Consideramos flujos en los que la diferencia de densidad es lo suficientemente pequeña como para que la aproximación de Boussinesq sea válida. Bajo estas circunstancias, las ecuaciones de conservación adimensionales son

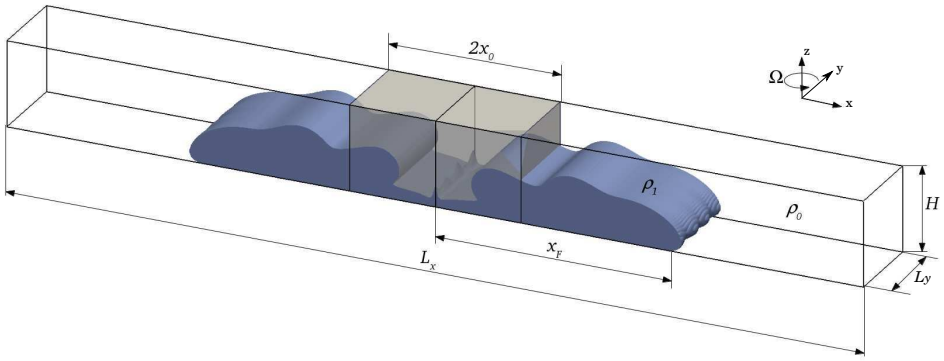

Figura 1: Esquema de la configuración física de la corriente de gravedad en estudio

$$\frac{D\tilde{\mathbf{u}}}{D\tilde{t}} = -\nabla\tilde{p} + \frac{1}{Re}\nabla^2\tilde{\mathbf{u}} + sgn(g)\,\tilde{\rho}\,\hat{\mathbf{z}} + 2\,\tilde{\mathcal{C}}\,(\tilde{v}\,\hat{\mathbf{x}} - \tilde{u}\,\hat{\mathbf{y}})\,, \tag{1}$$

$$\nabla\cdot\tilde{\mathbf{u}} = 0\,, \tag{2}$$

$$\frac{D\tilde{\rho}}{D\tilde{t}} = \frac{1}{Pe}\nabla^2\tilde{\rho}. \tag{3}$$

Aquí $\tilde{\mathbf{u}} = \{\tilde{u}, \tilde{v}, \tilde{w}\}$ es la velocidad adimensional, $\tilde{p}$ la presión adimensional y los parámetros adimensionales son el número de Reynolds, de Schmidt y el parámetro de Coriolis definidos como:

$$Re = \frac{U\,H}{\nu}\,, \qquad Sc = \frac{\nu}{\kappa} \quad \text{y} \quad \tilde{\mathcal{C}} = \frac{\Omega_z H}{U}, \tag{4}$$

donde ν es la viscosidad cinemática y κ la difusividad molecular.

En las ecuaciones (1)-(3) hemos utilizado las escalas de longitud H, de velocidad $U = \sqrt{\beta\,|\mathbf{g}|\,H}$ con $\beta = (\rho_1 - \rho_0)/\rho_0$, y la escala de tiempo $T = H/U$. La densidad adimensional está dada por

$$\tilde{\rho} = \frac{\rho - \rho_0}{\rho_1 - \rho_0}\,. \tag{5}$$

Las ecuaciones de conservación (1)-(3) se resuelven en un dominio adimensional rectangular de dimensiones $\tilde{L}_x \times \tilde{L}_y \times \tilde{H}$ con condiciones de contorno periódicas a lo largo de las direcciones horizontales $\tilde{x}$ e $\tilde{y}$ para todas las variables.

3. CÓDIGO DE CÁLCULO PSEUDO-ESPECTRAL

3.1. Discretización temporal

Para resolver las ecuaciones de conservación (1)-(3) utilizaremos un método de pasos fraccionados (Deville et al., 2002). El campo de velocidades del flujo es avanzado desde el instante de tiempo $t^{(n)}$ al $t^{(n+1)}$ en dos pasos fraccionados. Primero utilizamos una ecuación de advección-difusión para avanzar desde el instante $t^{(n)}$ al paso intermedio $t^{(n*)}$. Luego de este paso intermedio $(*)$ se utiliza un paso de corrección de presión para avanzar al instante de tiempo $t^{(n+1)}$ (Deville et al., 2002).

La discretización temporal del paso de advección-difusión y de la ecuación de transporte del escalar pasivo se logra utilizando un esquema mixto Runge-Kutta de tercer orden y Crank-Nicolson (Canuto et al., 2006). El esquema se lleva a cabo en tres etapas. El paso de tiempo de

$t^{(n)}$ a $t^{(n+1)}$ (Δt) es fraccionado en tres pasos más pequeños, con corrección de presión al final de cada paso de Runge-Kutta. La corrección de la presión se lleva a cabo en dos etapas. En la primera se resuelve una ecuación de Poisson y una vez que se determina la presión, se actualiza el campo de velocidades.

De aquí en adelante, para simplificar la notación, *omitiremos las tildes (∼) que identifican a las variables adimensionales* y definiremos los operadores de advección y difusión. El operador de difusión para la ecuación de conservación de momento es definido como

$$D() = \frac{1}{Re}\nabla^2() \tag{6}$$

Se utilizarán dos operadores de advección para el campo de velocidades

$$A_c() = -\mathbf{u}\cdot\nabla() \tag{7}$$

$$A_d() = -\nabla\cdot\big[\mathbf{u}()\big] \tag{8}$$

Esto corresponde a las formas convectivas y divergentes del operador de advección. Alternando entre estos dos operadores de advección para sucesivos pasos de tiempos se obtienen resultados similares a los que se ven con formas asimétricas de las ecuaciones de conservación, pero nuestro método tiene la ventaja que el número de derivadas que se necesitan evaluar para completar un paso de tiempo es reducido a la mitad (Zang, 1991). De aquí en adelante el operador de advección será referido como $A()$ sin mencionar su forma específica.

Además se definirán los operadores de advección y difusión para la ecuación de transporte del escalar. El operador de advección es

$$A_\rho() = -\nabla\cdot\big[\rho()\big], \tag{9}$$

y el de difusión

$$D_\rho = \frac{1}{Pe}\nabla^2\rho. \tag{10}$$

Definimos además el vector de fuerzas

$$\mathbf{F} = \mathbf{F}(\mathbf{u},\rho) = (2\,\mathcal{C}\,v, -2\,\mathcal{C}\,u, sgn(g)\,\rho), \tag{11}$$

donde $\mathbf{u} = (u, v, w)$ es el campo de velocidades y ρ el campo escalar.

Ahora mostraremos un paso de tiempo completo a través del esquema de tres etapas. Los super-índices 0, 1, 2 y 3 indican el nivel de la etapa de Runge-Kutta. Los campos en la etapa de nivel 0 son iguales a los del tiempo $t^{(n)}$, por ejemplo

$$\mathbf{u}^{(0)} = \mathbf{u}\left(t^{(n)}\right), \tag{12}$$

$$\rho^{(0)} = \rho\left(t^{(n)}\right). \tag{13}$$

A continuación se calculan el campo de velocidades de la etapa (1*) utilizando la ecuación de advección-difusión y el campo escalar de la etapa (1). Luego la presión correspondiente se evalúa resolviendo una ecuación de Poisson. Paso seguido, los gradientes de presión son

utilizados para corregir el campo de velocidades a un campo de divergencia nula:

$$\mathbf{H}^{(1)} = \Delta t\ A\left(\mathbf{u}^{(0)}\right) + \Delta t\ \mathbf{F}^{(0)}, \tag{14}$$

$$H_\rho^{(1)} = \Delta t\ A_\rho\left(\mathbf{u}^{(0)}\right), \tag{15}$$

$$\mathbf{u}^{(1*)} = \mathbf{u}^{(0)} + \frac{1}{3}\mathbf{H}^{(1)} + \frac{\Delta t}{6}\Big[D\left(\mathbf{u}^{(0)}\right) + D\left(\mathbf{u}^{(1*)}\right)\Big], \tag{16}$$

$$\rho^{(1)} = \rho^{(0)} + \frac{1}{3}H_\rho^{(1)} + \frac{\Delta t}{6}\Big[D_\rho^{(0)} + D_\rho^{(1)}\Big], \tag{17}$$

$$\nabla^2 p^{(1)} = \frac{3}{\Delta t}\nabla\cdot\mathbf{u}^{(1*)}, \tag{18}$$

$$\mathbf{u}^{(1)} = \mathbf{u}^{(1*)} - \frac{\Delta t}{3}\nabla p^{(1)}, \tag{19}$$

donde $\mathbf{F}^{(m)} = \mathbf{F}(\mathbf{u}^{(m)}, \rho^{(m)})$. Una vez que la primer etapa es completada, se realiza la segunda etapa del esquema:

$$\mathbf{H}^{(2)} = \Delta t\ A\left(\mathbf{u}^{(1)}\right) - \frac{5}{9}\mathbf{H}^{(1)} + \Delta t\ \mathbf{F}^{(1)}, \tag{20}$$

$$H_\rho^{(2)} = \Delta t\ A_\rho\left(\mathbf{u}^{(1)}\right) - \frac{5}{9}H_\rho^{(1)}, \tag{21}$$

$$\mathbf{u}^{(2*)} = \mathbf{u}^{(1)} + \frac{15}{16}\mathbf{H}^{(2)} + \frac{5\Delta t}{24}\Big[D\left(\mathbf{u}^{(1)}\right) + D\left(\mathbf{u}^{(2*)}\right)\Big], \tag{22}$$

$$\rho^{(2)} = \rho^{(1)} + \frac{15}{16}H_\rho^{(2)} + \frac{5\Delta t}{24}\Big[D_\rho^{(1)} + D_\rho^{(2)}\Big], \tag{23}$$

$$\nabla^2 p^{(2)} = \frac{12}{5\Delta t}\nabla\cdot\mathbf{u}^{(2*)}, \tag{24}$$

$$\mathbf{u}^{(2)} = \mathbf{u}^{(2*)} - \frac{5\Delta t}{12}\nabla p^{(2)}. \tag{25}$$

La tercera etapa del esquema se realiza a continuación:

$$\mathbf{H}^{(3)} = \Delta t\ A\left(\mathbf{u}^{(2)}\right) - \frac{153}{128}\mathbf{H}^{(2)} + \Delta t\ \mathbf{F}^{(2)}, \tag{26}$$

$$H_\rho^{(3)} = \Delta t\ A_\rho\left(\mathbf{u}^{(2)}\right) - \frac{153}{128}H_\rho^{(2)}, \tag{27}$$

$$\mathbf{u}^{(3*)} = \mathbf{u}^{(2)} + \frac{8}{15}\mathbf{H}^{(3)} + \frac{\Delta t}{8}\Big[D\left(\mathbf{u}^{(2)}\right) + D\left(\mathbf{u}^{(3*)}\right)\Big], \tag{28}$$

$$\rho^{(3)} = \rho^{(2)} + \frac{8}{15}H_\rho^{(3)} + \frac{\Delta t}{8}\Big[D_\rho^{(2)} + D_\rho^{(3)}\Big], \tag{29}$$

$$\nabla^2 p^{(3)} = \frac{4}{\Delta t}\nabla\cdot\mathbf{u}^{(3*)}, \tag{30}$$

$$\mathbf{u}^{(3)} = \mathbf{u}^{(3*)} - \frac{\Delta t}{4}\nabla p^{(3)}. \tag{31}$$

Una vez que la última etapa es completada, los campos del flujo son actualizados y el proceso puede comenzar nuevamente:

$$\mathbf{u}\left(t^{(n+1)}\right) = \mathbf{u}^{(3)}, \tag{32}$$

$$\rho\left(t^{(n+1)}\right) = \rho^{(3)}. \tag{33}$$

Estas ecuaciones pueden simplificarse definiendo un número de etapa *m*, un coeficiente para el primer término no lineal *cnl1*, un coeficiente para el segundo término no lineal *cnl2* y un coeficiente para el término difusivo *cd*. Estas cantidades son definidas como:

$$m = (1,2,3), \tag{34}$$

$$cnl1(m) = \left(0, -\frac{5}{9}, -\frac{153}{128}\right), \tag{35}$$

$$cnl2(m) = \left(\frac{1}{3}, \frac{15}{16}, \frac{8}{15}\right), \tag{36}$$

$$cd(m) = \left(\frac{\Delta t}{6}, \frac{5\Delta t}{24}, \frac{\Delta t}{8}\right). \tag{37}$$

Con esta nueva notación, se pueden resumir las tres etapas del esquema:

$$\mathbf{u}^{(0)} = \mathbf{u}\left(t^{(n)}\right), \tag{38}$$

$$\rho^{(0)} = \rho\left(t^{(n)}\right), \tag{39}$$

Para m=1,2,3:

$$\mathbf{H}^{(m)} = \Delta t\ A\left(\mathbf{u}^{(m-1)}\right) + cnl1(m)\ \mathbf{H}^{(m-1)} + \Delta t\ \mathbf{F}^{(m-1)}, \tag{40}$$

$$H_\rho^{(m)} = \Delta t\ A_\rho\left(\mathbf{u}^{(m-1)}\right) + cnl1(m)\ H_\rho^{(m-1)}, \tag{41}$$

$$\mathbf{u}^{(m*)} = \mathbf{u}^{(m-1)} + cnl2(m)\ \mathbf{H}^{(m)} + cd(m)\ \left[D\left(\mathbf{u}^{(m-1)}\right) + D\left(\mathbf{u}^{(m*)}\right)\right], \tag{42}$$

$$\rho^{(m)} = \rho^{(m-1)} + cnl2(m)\ H_\rho^{(m)} + cd(m)\ \left[D_\rho^{(m-1)} + D_\rho^{(m)}\right], \tag{43}$$

$$\nabla^2 p^{(m)} = \frac{1}{2\ cd(m)} \nabla \cdot \mathbf{u}^{(m*)}, \tag{44}$$

$$\mathbf{u}^{(m)} = \mathbf{u}^{(m*)} - 2\ cd(m)\ \nabla p^{(m)}, \tag{45}$$

$$\mathbf{u}\left(t^{(n+1)}\right) = \mathbf{u}^{(3)}, \tag{46}$$

$$\rho\left(t^{(n+1)}\right) = \rho^{(3)}. \tag{47}$$

3.2. Discretización espacial

Las condiciones de contorno periódicas en las direcciones horizontales x e y sugieren el uso de expansiones de Fourier. En las direcciones x e y los puntos de grilla son uniformemente espaciados. En cambio, la no periodicidad de la dirección normal z junto con la necesidad de resolver los altos gradientes cercanos a las paredes superior e inferior sugiere el uso de puntos de grilla no uniformemente espaciados en la dirección z. Se utilizan polinomios de Chevyshev

como funciones base para las expansiones en la dirección normal a las paredes. La expansión utilizada para discretizar una variable u es:

$$u\left(x_j, y_k, z_l, t\right) = \sum_{k_x=-\frac{N_x}{2}}^{\frac{N_x}{2}-1} \sum_{k_y=-\frac{N_y}{2}}^{\frac{N_y}{2}-1} \sum_{k_z=0}^{N_z-1} \hat{u}\left(k_x, k_y, k_z, t\right)\ e^{\left(i\frac{2\pi k_x}{L_x}x_j\right)}\ e^{\left(i\frac{2\pi k_y}{L_y}y_k\right)}\ T_{k_z}(z_l), \tag{48}$$

donde T_{k_z} se refiere al k_z-esimo polinomio de Chevyshev y k_x y k_y son los números de onda.

En la dirección normal a las paredes z utilizaremos puntos de cuadratura Gauss-Lobatto, los cuales proveen una mayor resolución cerca de las paredes así como puntos de grilla en las paredes superior e inferior, lo que permite un tratamiento directo de las condiciones de contorno de Dirichlet y Neumann. La posición de los puntos de grilla en las direcciones horizontales x e y y en la dirección normal z son

$$x_j = \frac{jL_x}{N_x}, \qquad 0 \leq j \leq N_x - 1, \tag{49}$$

$$y_k = \frac{kL_y}{N_y}, \qquad 0 \leq k \leq N_y - 1, \tag{50}$$

$$z_l = \frac{H}{2}\cos\left[\frac{(l-1)\pi}{N_z - 1}\right], \qquad 1 \leq l \leq N_z, \tag{51}$$

donde N_x, N_y y N_z son la cantidad de puntos de grilla en las direcciones x, y y z, respectivamente.

Derivadas espectralmente precisas se obtienen utilizando técnicas de transformadas rápidas de Fourier (*FFT*) en las direcciones horizontales y técnicas de multiplicación matricial en la dirección normal a las paredes.

Un paso de tiempo se completa utilizando las ecuaciones (38) a (47). Las ecuaciones (40) a (45) deben ser resueltas para cada una de las tres etapas del esquema. Con trabajo algebraico de las ecuaciones (42) a (44) obtenemos ecuaciones Helmholtz/Poisson para las tres componentes de la velocidad, el campo escalar pasivo y para la presión. El término difusivo en el que interviene $\mathbf{u}^{(m*)}$ puede moverse al lado izquierdo de la ecuación (42). La componente x de la ecuación de conservación de momento puede reescribirse como

$$u^{(m*)} - cd(m)\ D\left(u^{(m*)}\right) = u^{(m-1)} + cnl2(m)\ H_x^{(m)} + cd(m)\ D\left(u^{(m-1)}\right). \tag{52}$$

Reemplazando al operador $D()$ por su forma escalar obtenemos

$$\left(\frac{\partial^2 u^{(m*)}}{\partial x^2} + \frac{\partial^2 u^{(m*)}}{\partial y^2} + \frac{\partial^2 u^{(m*)}}{\partial z^2}\right) - \frac{Re}{cd(m)}u^{(m*)} =$$

$$= \frac{Re}{cd(m)}\left[u^{(m-1)} + cnl2(m)\ H_x^{(m)}\right] + \left(\frac{\partial^2 u^{(m-1)}}{\partial x^2} + \frac{\partial^2 u^{(m-1)}}{\partial y^2} + \frac{\partial^2 u^{(m-1)}}{\partial z^2}\right). \tag{53}$$

El lado derecho de la ecuación (53) puede ser calculado con información conocida. Para simplificar el análisis, el lado derecho de la ecuación se escribirá como $rhs_x^{(m*)}$.

Como siguiente paso, se realiza la transformada de Fourier (en las direcciones x e y) de la ecuación (53)

$$\frac{\partial^2 \hat{u}^{(m*)}}{\partial z^2} - \left(\frac{Re}{cd(m)} + k_x'^{\,2} + k_y'^{\,2}\right)\hat{u}^{(m*)} = \hat{rhs}_x^{(m*)}, \tag{54}$$

donde $k_x'^2 = \frac{2\pi k_x}{L_x}$, $k_y'^2 = \frac{2\pi k_y}{L_y}$ y el sombrero "ˆ" indica coeficiente de Fourier. De manera similar se puede escribir la ecuación (43) para el campo escalar

$$\frac{\partial^2 \hat{\rho}^{(m)}}{\partial z^2} - \left(\frac{Pe}{cd(m)} + k_x'^2 + k_y'^2 \right) \hat{\rho}^{(m)} = r\hat{h}s_\rho^{(m)}. \tag{55}$$

La discretización en la dirección z puede expresarse en términos de una operación de multiplicación matricial (Canuto et al., 1988). Combinando una transformación de Chevyshev discreta hacia adelante, diferenciación recursiva y una transformación inversa de Chevyshev, puede ser definida una única operación de multiplicación matricial para diferenciar una variable discretizada. La matriz para la derivada primera con respecto a z es escrita de manera simbólica como

$$\frac{\partial()}{\partial z} = \overline{\overline{D}}_1(). \tag{56}$$

La matriz $\overline{\overline{D}}_1()$ es asimétrica de N_z filas y N_z columnas (Canuto et al., 1988). Un operador para realizar la segunda derivada parcial con respecto a z puede ser formulado simplemente multiplicando dos operadores matriciales de derivada primera:

$$\frac{\partial^2()}{\partial z^2} = \frac{\partial\left(\frac{\partial()}{\partial z}\right)}{\partial z} = \overline{\overline{D}}_1\left(\overline{\overline{D}}_1()\right) = \overline{\overline{D}}_2(). \tag{57}$$

La notación de la ecuación (57) puede ser utilizada para reemplazar las derivadas parciales en la ecuación (54):

$$\overline{\overline{D}}_2\left(\hat{u}^{(m*)}\right) - \left(\frac{Re}{cd(m)} + k_x'^2 + k_y'^2 \right) \hat{u}^{(m*)} = r\hat{h}s_x^{(m*)}. \tag{58}$$

De manera similar, las componentes y y z de la ecuación de advección-difusión puede ser discretizada como

$$\overline{\overline{D}}_2\left(\hat{v}^{(m*)}\right) - \left(\frac{Re}{cd(m)} + k_x'^2 + k_y'^2 \right) \hat{v}^{(m*)} = r\hat{h}s_y^{(m*)} \tag{59}$$

$$\overline{\overline{D}}_2\left(\hat{w}^{(m*)}\right) - \left(\frac{Re}{cd(m)} + k_x'^2 + k_y'^2 \right) \hat{w}^{(m*)} = r\hat{h}s_z^{(m*)}. \tag{60}$$

La ecuación (55) para el escalar pasivo escrita con la notación de la ecuación (57) es

$$\overline{\overline{D}}_2\left(\hat{\rho}^{(m)}\right) - \left(\frac{Pe}{cd(m)} + k_x'^2 + k_y'^2 \right) \hat{\rho}^{(m)} = r\hat{h}s_\rho^{(m)}. \tag{61}$$

3.3. Procedimiento de resolución

Para resolver los campos del flujo en el paso de tiempo intermedio, se deben resolver las ecuaciones (58)-(61) para las tres componentes de velocidad y el campo escalar, para cada combinación de números de onda horizontales. Además se debe resolver la ecuación de Poisson para la presión. Esto quiere decir que para, por ejemplo, una grilla de $60 \times 60 \times 61$ puntos se deben resolver 54000 veces las ecuaciones de Helmholtz o Poisson para cada paso de tiempo

completo (tres componentes de velocidad, el campo escalar y la presión para tres pasos fraccionados en 60×60 números de onda horizontales). Es necesario una técnica para resolver estas ecuaciones de manera rápida. En su forma actual, las ecuaciones requieren una inversión matricial para cada combinación de número de onda. Diagonalizando la matriz es posible eliminar la dependencia de la inversión con respecto a los números de onda, y por ende, será posible invertir la matriz una sola vez. Definimos la matriz modal $\overline{\overline{T}}$ de la matriz $\overline{\overline{D}}_2$ (ver ecuaciones (58) a (61)), $\overline{\overline{T}}^{-1}$ la inversa de la matriz modal, $\boldsymbol{\lambda}$ el vector cuyos elementos son los autovalores de $\overline{\overline{D}}_2$ y $\overline{\overline{\Lambda}}$ la matriz diagonal de dimensiones $N_z \times N_z$ cuyos elementos diagonales son los autovalores de $\overline{\overline{D}}_2$. Como $\overline{\overline{D}}_2$ es diagonalizable, puede ser descompuesta como

$$\overline{\overline{D}}_2 = \overline{\overline{T}}\ \overline{\overline{\Lambda}}\ \overline{\overline{T}}^{-1}. \tag{62}$$

De esta forma, la ecuación (58) para la componente $\hat{u}$ de la velocidad puede ser reescrita como

$$\overline{\overline{T}}\ \overline{\overline{\Lambda}}\ \overline{\overline{T}}^{-1}\,\hat{u}^{(m*)} - \alpha\,\hat{u}^{(m*)} = r\hat{h}s_x^{(m*)}, \tag{63}$$

donde el coeficiente α es

$$\alpha \;=\; \frac{Re}{cd(m)} + k_x'^{\,2} + k_y'^{\,2} \quad \text{para } \hat{u}, \hat{v} \text{ y } \hat{w}. \tag{64}$$

Para el campo escalar $\hat{\rho}$

$$\alpha \;=\; \frac{Pe}{cd(m)} + k_x'^{\,2} + k_y'^{\,2}. \tag{65}$$

El coeficiente α puede ser combinado con $\overline{\overline{\Lambda}}$. Sea $\overline{\overline{I}}$ una matriz identidad de dimensiones $N_z \times N_z$, la ecuación (63) puede acomodarse para dar una expresión para $\hat{u}^{(m*)}$:

$$\hat{u}^{(m*)} = \overline{\overline{T}}\ \left(\overline{\overline{\Lambda}} - \alpha\overline{\overline{I}}\right)^{-1}\ \overline{\overline{T}}^{-1} r\hat{h}s_x^{(m*)}. \tag{66}$$

Los sistemas matriciales a resolver para las componentes $\hat{v}$, $\hat{w}$ y para el campo escalar $\hat{\rho}$ son

$$\hat{v}^{(m*)} \;=\; \overline{\overline{T}}\ \left(\overline{\overline{\Lambda}} - \alpha\overline{\overline{I}}\right)^{-1}\ \overline{\overline{T}}^{-1} r\hat{h}s_y^{(m*)}. \tag{67}$$

$$\hat{w}^{(m*)} \;=\; \overline{\overline{T}}\ \left(\overline{\overline{\Lambda}} - \alpha\overline{\overline{I}}\right)^{-1}\ \overline{\overline{T}}^{-1} r\hat{h}s_z^{(m*)}. \tag{68}$$

$$\hat{\rho}^{(m)} \;=\; \overline{\overline{T}}\ \left(\overline{\overline{\Lambda}} - \alpha\overline{\overline{I}}\right)^{-1}\ \overline{\overline{T}}^{-1} r\hat{h}s_\rho^{(m)}. \tag{69}$$

4. PARALELIZACIÓN DEL CÓDIGO DE CÁLCULO PSEUDO-ESPECTRAL MEDIANTE MPI Y PETSC

Aunque en las últimas décadas el avance tecnológico en la industria informática a sido exponencial, las simulaciones directas de turbulencia para flujos a números de Reynolds cercanos a los reales es todavía imposible de realizar. No sólo estamos limitados por la capacidad de cálculo de las computadoras actuales sino que, al resolver las simulaciones tipo DNS un amplio espectro de escalas espaciales y temporales, la capacidad de almacenamiento en memoria es una variable fundamental en la factibilidad de una simulación directa de turbulencia.

El código speudo-espectral presentado en la sección 3 se encontraba programado en el lenguaje *FORTRAN 77* y utilizando las bibliotecas de programas *BLAS* (subprogramas de álgebra lineal básica) y *LAPACK* (Paquetes de álgebra lineal numérica). El código fue implementado mediante la interfaz de programación de aplicaciones (API) para programación en paralelo bajo el paradigma de memoria compartida *"OpenMP"*, por lo que las simulaciones que se podían realizar con este código estaban limitadas a la capacidad de la memoria de una computadora ($\sim 64\,GB$). Con el objetivo de superar esta barrera tecnológica en nuestra investigación de los flujos turbulentos, implementamos la paralelización del código bajo el paradigma de memoria distribuida. Para esto utilizamos la implementación *MPICH* del estandar *MPI* ("Message passing interface") junto al paquete de herramientas extensibles y portables para la computación científica *PETSc*. Además realizamos la programación del código en el lenguaje *FORTRAN 90*, permitiendo el uso de reserva dinámica de memoria.

4.1. Rutinas paralelizadas

4.1.1. Rutina principal

La rutina principal comienza con la lectura de los datos de entrada por parte del proceso "maestro", y de la comunicación de estos datos hacia el resto de los procesos mediante la rutina de MPI *"MPI_Bcast"*. Luego de esto, se calcula cuantos planos $x-y$ y $x-z$ se le asignarán a cada proceso para que luego reserven memoria. La distribución de datos en planos $x-y$ se utilizará en la mayoría de las subrutinas como ser las transformaciones al espacio de Fourier, donde los bucles principales son con respecto a la variable z. La distribución en planos $x-z$ se utilizará principalmente en la rutina que realiza la derivación en la dirección z y en el resolvedor de Helmholtz, donde los bucles principales son con respecto a la variable y.

Para realizar la transposición de los datos se crearán vectores de PETSc. Por cada arreglo a transponer, se creará un vector con la distribución en planos $x-y$ y un vector con la distribución en planos $x-z$. Para lograr la transposición se utilizarán sets de índices enteros. A continuación se muestra como ejemplo la creación del contexto de dispersión (Scatter Context) de los arreglos que contienen el campo escalar $\hat{\rho}$:

```
!       Se define el tamaño global del arreglo para el campo
!       escalar (nx0=nx+1 y ny0=ny+1).
!
        glosize = nx0 * ny0 * nz

!       Se define el tamaño del arreglo que contiene los datos
!       locales del escalar distribuido en planos x-y.
!       La variable nzpp contiene la  cantidad de planos que
!       le corresponde a cada proceso.
!
        locsize = nx0 * ny0 * nzpp

!       Se reserva memoria para el arreglo local (tt)
!       y se crea el Vector de PETSc
        ALLOCATE(tt(nx0,ny0,nzpp))
        Call VecCreateMPIWithArray(PETSC_COMM_WORLD, 1,locsize,
     &                             glosize, tt, p_tt, ierr)
```

```
!       Se define el tamaño del arreglo que contiene los
!       datos locales del escalar distribuido en planos x-z.
!       Se reserva memoria para dicho arreglo (tt_trans) y
!       se crea  el Vector de PETSc. La variable nypp contiene
!       la cantidad de planos x-z que le corresponde a cada
!       proceso.
!
        locsize = nx0 * nypp * nz
        ALLOCATE(tt_trans(nz, nx0, nypp))
        Call VecCreateMPIWithArray (PETSC_COMM_WORLD, 1,locsize,
     &                         glosize, tt_trans, p_tt_trans, ierr)


!       Se crea el Index Set (set de índices enteros)
!       para el Vector distribuido en planos x-y.
!
        locsize = nx0 * ny0 * nzpp
        Call ISCreateStride (PETSC_COMM_WORLD, locsize,
     &                         stzp*nx0*ny0, 1, p_isv, ierr)


!       Se crea el Index Set (set de índices enteros)
!       para el Vector distribuido en planos x-z.
!
        ALLOCATE(iaux(locsize))
        Do iz = 1, nzpp
           Do iy = 1, ny0
              Do ix = 1, nx0
!    Acceso secuencial a iaux
                 iaux(ix-1 + nx0*(iy-1) + nx0*ny0*(iz-1) + 1) =
!     indice z: comenzando desde el primer plano z
     &                 (iz-1) + stzp +
!     indice x: multipliación por el tamaño global nz
     &                 (ix-1) * nz +
!     indice y: multipliación por los tamaños globales nx0 y nz
     &                 (iy-1) * nz*nx0
              End Do
           End Do
        End Do
        Call ISCreateGeneral (PETSC_COMM_WORLD, locsize,iaux,
     &                          PETSC_COPY_VALUES, p_isvt, ierr)

!       Se crea el contexto de dispersión (Scatter context)
!       para la transposición de los datos del campo escalar.
!
        Call VecScatterCreate (p_tt,p_isv,p_tt_trans,p_isvt
```

```
     &                                ,p_vscat_tt,ierr)
```

La distribución de los datos se esquematiza en la figura 2 en donde los datos del dominio global $nx0 \times ny0 \times nz$ se distribuyen entre 10 procesos (del proceso 0 "maestro" al proceso 9). Cada proceso reserva memoria para un arreglo de dimensiones $nx0 \times ny0 \times nzpp$ y otro de dimensiones $nz \times nx0 \times nypp$. El primero será utilizado en la mayoría de los casos, por ejemplo en las transformaciones al espacio de Fourier de las variables del flujo, mientras que el segundo arreglo (el transpuesto) se utilizará en las rutinas de derivación en la dirección z y en el resolvedor de Helmholtz.

En la figura 2 se destacan los datos que tiene el proceso 5 (regiones azules sombreadas). Para poder utilizar datos actualizados en las rutinas que los requieren distribuidos en planos $x - z$, debemos realizar la transposición hacia adelante con las rutinas de PETSc *"VecScatterBegin"* y *"VecScatterEnd"* utilizando el argumento *"SCATTER_FORWARD"*. Luego de esto debemos utilizar las mismas rutinas pero con el argumento *"SCATTER_REVERSE"*, para actualizar las variables en la distribución original de planos $x - y$. El uso de estos se ejemplificará en la sección 4.1.4.

Luego de la reserva de memoria y de la creación de los vectores de PETSc, se realiza la lectura de las condiciones iniciales, las cuales se encuentran en dos archivos binarios, uno para el campo escalar y otro para el campo de velocidades, y se transforman al espacio de Fourier dichos campos. El campo escalar $\hat{\rho}$ se encuentra contenido en el arreglo local tt, mientras que el campo de velocidades se encuentra en los arreglos locales u, v y w.

Paso siguiente se llama a la rutina *"advance"* en la cual se calculan los términos $r\hat{h}s$ de las ecuaciones (58) a (61) a resolver y se llama al resolvedor Helmholtz, obteniéndose $\hat{\mathbf{u}}^{(m*)}$ y $\hat{\rho}^{(m)}$. Luego de esto (también en la rutina *"advance"*) se realiza el paso de corrección de presión con el cual se actualiza el campo de velocidades al siguiente paso de Runge-Kutta. Cabe destacar que este proceso se realiza para las tres etapas m del esquema numérico (ver sección 3.1).

Finalmente se escribe en dos archivos binarios los datos del campo escalar y del campo de velocidades para el nuevo paso de tiempo.

Detalles sobre las rutinas de MPI y PETSc utilizadas se pueden encontrar en MPI (1996) y PET (2012).

4.1.2. Entrada y salida de datos

Las rutinas *"input"* y *"output"* se encargan de la lectura y la escritura de los datos en archivos binarios. Si se parte de una condición inicial nueva, la rutina *"input"* no es llamada y se crean las condiciones iniciales (campo de velocidades nulo y campo escalar definido por el usuario).

Para la lectura se utilizan las rutinas de MPI *"MPI_File_open"*, *"MPI_File_set_view"*, *"MPI_File_read_at_all"* y *"MPI_File_close"*. A modo de ejemplo mostramos la porción de código en la que se lee el archivo binario "conc.0" que contiene el campo escalar:

```
!     Se abre el archivo binario conc.0 en modo solo lectura.
!
      Call MPI_File_open (MPI_COMM_WORLD, 'conc.0'
     &        ,MPI_MODE_RDONLY, MPI_INFO_NULL, FH, ierr)

!     Se ajusta la vista de los datos en el archivo,
!     indicando el desplazamientos en bytes desde el
!     comienzo del archivo (idisp).
```

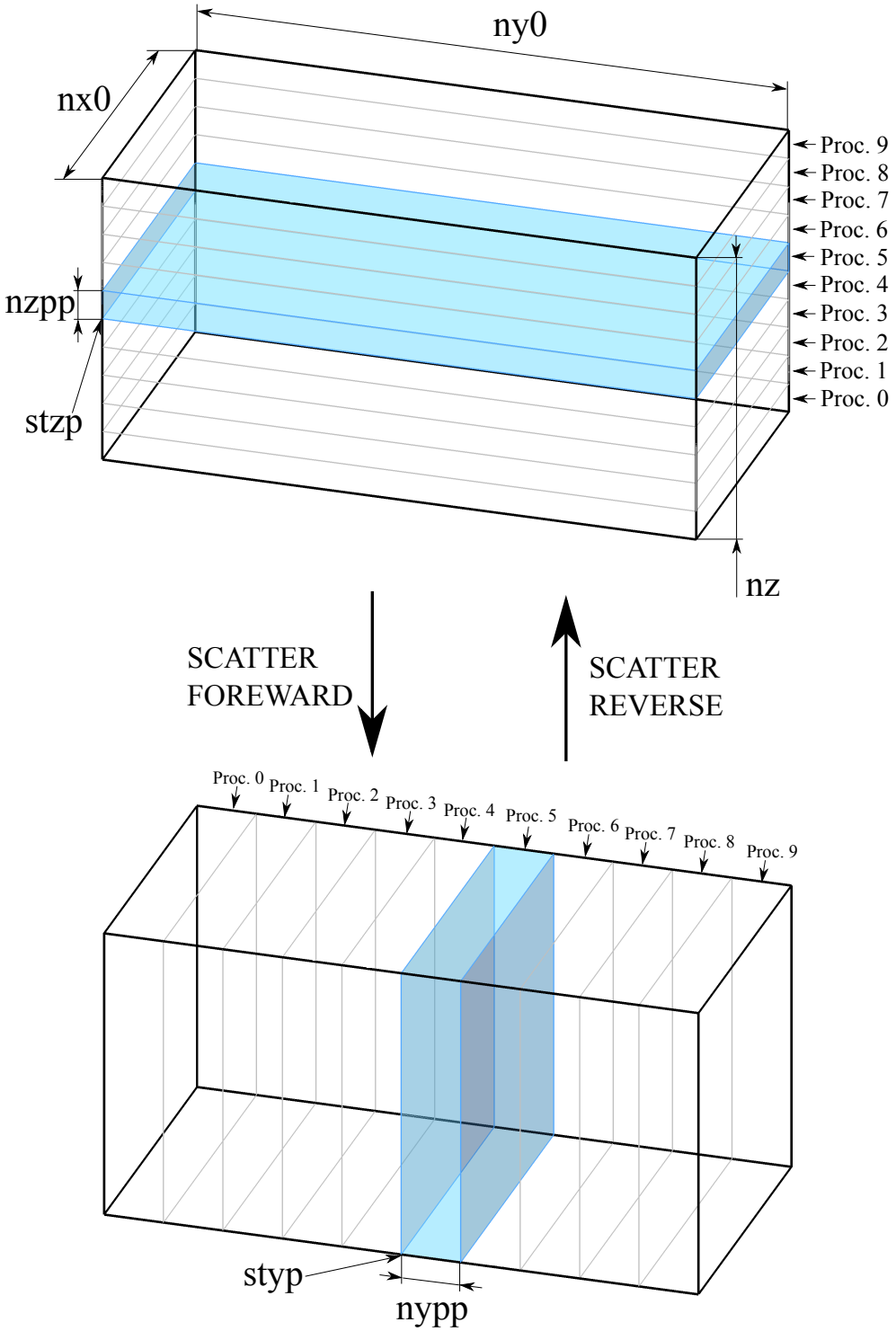

Figura 2: Esquema del proceso de transposición realizado mediante el contexto de dispersión de datos (Scatter Context) de PETSc. En la figura superior se muestran los datos distribuidos en planos $x - y$, donde cada proceso tiene $nx0 \times ny0 \times nzpp$ datos de la variable. En la figura inferior se muestran los datos distribuidos en planos $x - z$, donde cada proceso tiene $nz \times nx0 \times nypp$ datos de la variable.

```
!
      Call MPI_File_set_view (FH, idisp, MPI_DOUBLE_PRECISION,
     &        MPI_DOUBLE_PRECISION, "native", MPI_INFO_NULL, ierr)
```

```
!     Cada proceso lee el archivo a desplazamientos
!     especificados por la variable "ioff".

      ioff = stzp*nx0*ny0
      Call MPI_File_read_at_all(FH, ioff, tt, nzpp*nx0*ny0,
     &        MPI_DOUBLE_PRECISION, status, ierr)

!      Se cierra el archivo

       Call MPI_File_close (FH, ierr)
```

Como se puede ver en el código presentado arriba, cada proceso leerá $nzpp \times nx0 \times ny0$ valores de doble precisión, y cada proceso comenzará a leerlos con un desplazamiento $stzp \times nx0 \times ny0$, donde $stzp$ es el primer plano $x - y$ que le corresponde a cada proceso.

Para la escritura de los datos se utilizaron las siguientes rutinas de MPI:

```
!     Se abre el archivo binario conc.1 en modo creación
!     y escritura.
!
      Call MPI_File_open (MPI_COMM_WORLD, 'conc.1'
     & ,MPI_MODE_WRONLY+MPI_MODE_CREATE, MPI_INFO_NULL, FH, ierr)

!     Se ajusta la vista de los datos en el archivo, indicando
!     el desplazamiento en bytes desde el comienzo del
!     archivo (idisp).
!
      Call MPI_File_set_view (FH, idisp, MPI_DOUBLE_PRECISION,
     &        MPI_DOUBLE_PRECISION, "native", MPI_INFO_NULL, ierr)

!     Cada proceso escribe el archivo a desplazamientos
!     especificados por la variable "ioff".

      ioff = stzp*nx0*ny0
      Call MPI_File_write_at_all(FH, ioff, tt, nzpp*nx0*ny0,
     &        MPI_DOUBLE_PRECISION, status, ierr)

!      Se cierra el archivo

       Call MPI_File_close (FH, ierr)
```

4.1.3. Transformadas de Fourier

Para las transformaciones al espacio de Fourier se utilizan FFTWs (Transformada de Fourier más rápida del Oeste). En esta rutina del código, los datos de cada arreglo local que contiene planos $x - y$ son transformados al espacio de Fourier. Estos se realizan en un bucle sobre todos los planos $x - y$ que contiene cada proceso ($nzpp$). A modo de ejemplo mostramos una de las rutinas:

```
      subroutine fft_r2f_2d(ur)
      USE variables
      IMPLICIT none
      REAL*8 ur(nx0,ny0,nzpp)
      INTEGER i,j,k,jimag

      do k=1,nzpp
         call dscal(nx0y,1.d0/dble(nx)/dble(ny),ur(1,1,k),1)
! calcula fft en la dirección y
         call dfftw_execute_dft_r2c(plan_nx_r2c_y
     &        ,ur(1,1,k),tmpf(1,1,k))
! calcula fft en la dirección x
         call dfftw_execute_dft(plan_nyh_c2cf_x
     &        ,tmpf(1,1,k),tmpf(1,1,k))
         do j=1,nyh
            jimag=nyh+j
            do i=1,nx
               ur(i,j    ,k) = dreal(tmpf(j,i,k))
               ur(i,jimag,k) = dimag(tmpf(j,i,k))
            enddo
         enddo
      enddo
```

Detalles sobre las rutinas de BLAS y FFTW utilizadas se pueden encontrar en BLA (2010) y Frigo (2003).

4.1.4. Derivada en la dirección vertical "z"

Para realizar la derivación en la dirección vertical z, requerida en el cálculo de los términos no lineales, se llama a la rutina ddz la cual realiza el producto matricial de cada plano $x-z$ con la matriz de derivada primera de Chebyshev $\overline{\overline{D}}_1$ (ver ecuación (56)). Debido a esto cada proceso deberá tener una cantidad $nypp$ de planos $x-z$. Para esto utilizamos el contexto de dispersión creado en la rutina principal. Para transponer los datos en cada proceso, de planos $x-y$ a $x-z$, se utilizarán las rutinas de PETSc *"VecScatterBegin"* y *"VecScatterEnd"*:

```
      Call VecScatterBegin(p_vscat_tt, p_tt, p_tt_trans,
     &     INSERT_VALUES, SCATTER_FORWARD, ierr)
      Call VecScatterEnd(p_vscat_tt, p_tt, p_tt_trans,
     &     INSERT_VALUES, SCATTER_FORWARD, ierr)
```

siendo *"p_vscat_tt"* el contexto de dispersión (Scatter context) creado en la rutina principal, *"p_tt"* el vector de PETSc desde el cual realizamos la dispersión de los datos, y *"p_tt_trans"* el vector hacia el cual realizamos la dispersión de los datos.

Luego de realizada la dispersión de los datos, multiplicamos planos $x-z$ del arreglo que contiene los datos a derivar por la matriz de derivada primera de Chebyshev $\overline{\overline{D}}_1$.

Luego de hacer el producto matricial, invertimos el proceso de dispersión para tener en cada proceso los datos calculados distribuidos en planos $x-y$:

```
 Call VecScatterBegin(p_vscat_tt, p_tt_trans, p_tt,
&     INSERT_VALUES, SCATTER_REVERSE, ierr)
 Call VecScatterEnd(p_vscat_tt, p_tt_trans, p_tt,
&     INSERT_VALUES, SCATTER_REVERSE, ierr).
```

Estos contextos de dispersión también son utilizados cuando se requiere calcular las derivadas segundas en la dirección vertical z para obtener $\hat{rhs}_x^{(m*)}$, $\hat{rhs}_y^{(m*)}$, $\hat{rhs}_z^{(m*)}$ y $\hat{rhs}_\rho^{(m)}$ (ver ecuación (53)), en donde se realiza el producto matricial de cada plano $x - z$ con la matriz de derivada segunda de Chebyshev $\overline{\overline{D}}_2$ (ver ecuación (57)).

En el resolvedor de las ecuaciones de Helmholtz ((58) a (61)) también se utiliza una distribución de los datos en planos $x - z$, por lo que se utilizan estos contextos de dispersión.

4.2. Desempeño del código distribuido

Las pruebas del código distribuido se realizaron en el cluster del departamento de *Medicina Asistida por Computación Científica* del *Instituto de Ciencia y Tecnología de Brasil* (INCT-MACC), el cual cuenta con nodos interconectados mediante una red Infiniband. Los mismos son computadoras con procesadores *Intel® Xeon® CPU X5670 de 12 nucleos a 2.93GHz* y 36GB de memoria RAM.

En las pruebas realizadas se combinó el uso de *OpenMP*, *MPI* y se compararon todos los resultados para determinar la mejor estrategia de distribución del trabajo de cálculo. Se utilizaron como condiciones iniciales los campos de un flujo transitorio en una grilla de $80 \times 1536 \times 180$ puntos.

Como en nuestro caso el código ya se encontraba paralelizado con OpenMP, es adecuado que los *Speed Ups* sean determinados con respecto a una simulación distribuida en varios procesadores de un mismo nodo mediante *OpenMP*. En la tabla 1 se muestran las cinco pruebas que se realizaron, las cuales están identificadas de la siguiente forma:

$$X\,nodos \times Y\,procesos\,MPI \times Z\,hilos\,OpenMP, \tag{70}$$

donde la cantidad total de procesos de cálculo es $X \times Y \times Z$. Todos los tiempos se especifican en segundos. Para comparar las distintas pruebas obtuvimos los tiempos de lectura de datos (ver sección 4.1.2), de avance de un paso temporal Δt (sólo cálculo numérico y comunicación de datos) y de escritura de datos. Además se presentan los tiempos de las comunicaciones realizadas entre procesos en un paso temporal (*"VecScatter en* Δt*"* en la tabla 1) y el porcentaje del tiempo de cálculo que este representa

$$VecScatter\ en\ \Delta t\ (\%) = \frac{VecScatter\ en\ \Delta t}{Avance\ \Delta t} \times 100. \tag{71}$$

Como en una simulación real la lectura y escritura de los datos desde y hacia el disco no se realiza en cada paso temporal, es conveniente definir dos *Speed Ups*. En el primero (*Speed Up* (I)) se realiza el cociente entre el tiempo total de la simulación (incluyendo lectura y escritura de los datos) con respecto al tiempo total de la simulación $1 \times 1 \times 12$ (simulación distribuida en la mayor cantidad de hilos *OpenMP* en un único nodo de cálculo). El segundo (*Speed Up* (II)) es el cociente entre el tiempo de avance de un paso de tiempo de la simulación (sin considerar la lectura y escritura de los datos) y el tiempo de avance de un paso de tiempo de la simulación $1 \times 1 \times 12$.

Para realizar una comparación entre *OpenMP* y *MPI* se realizaron las simulaciones $1 \times 1 \times 12$ y $1 \times 12 \times 1$, en las cuales se ejecutan la misma cantidad de procesos de cálculo, pero

con la diferencia que en la segunda simulación los doce procesos son de *MPI* (*OpenMP* no realizará ninguna transferencia de datos). En la primera simulación la comunicación entre los procesos de cálculo las manejará *OpenMP*, mientras que *MPI* se encargará solamente de realizar la transposición de datos (transferencia de datos entre lugares de memoria en el mismo nodo). Como podemos ver, existe un aumento en la velocidad de ejecución en la segunda simulación ($1 \times 12 \times 1$), debido principalmente a la disminución del tiempo de lectura (de $28{,}751$ para la primer simulación a $7{,}183$ para la segunda) y a la disminución del tiempo requerido por las comunicaciones.

Comparando las otras simulaciones ($2 \times 12 \times 1$, $4 \times 12 \times 1$ y $8 \times 12 \times 1$) podemos ver que no es la simulación con más número de procesos la que presenta los valores más altos de *Speed Up (I)*, sino que es la simulación $4 \times 12 \times 1$. Esto es debido a que al requerirse los datos distribuidos en una mayor cantidad de nodos, los tiempos de lectura y escritura son mayores, aumentando el tiempo de ejecución total de la simulación. Sin embargo, analizando el *Speed Up (II)* en el cual sólo consideramos el tiempo de cálculo, podemos ver que la simulación $8 \times 12 \times 1$, con la mayor cantidad de procesos, presenta los valores de *Speed Up* más altos. Este es un resultado prometedor ya que en una simulación real la escritura de los datos no se realiza en todos los pasos de tiempo y la lectura de los datos se realiza sólo en los casos donde queremos retomar una simulación.

	$1 \times 1 \times 12$	$1 \times 12 \times 1$	$2 \times 12 \times 1$	$4 \times 12 \times 1$	$8 \times 12 \times 1$
$Lectura\ de\ datos$	$28,751$	$7,183$	$7,146$	$7,472$	$12,223$
$Avance\ \Delta t$	$49,199$	$38,705$	$19,949$	$11,334$	$8,039$
$VecScatter\ en\ \Delta t$	$12,285$	$4,436$	$2,507$	$2,207$	$2,713$
$VecScatter\ en\ \Delta t\ (\%)$	$24,97\,\%$	$11,461\,\%$	$12,568\,\%$	$19,474\,\%$	$33,751\,\%$
$Escritura\ de\ datos$	$8,826$	$9,184$	$10,164$	$11,519$	$13,25$
Tiempo total	$86,776$	$55,072$	$37,259$	$30,325$	$33,512$
Speed Up (I)	$1,00$	$1,58$	$2{,}33$	$2,86$	$2,59$
Speed Up (II)	$1,00$	$1,27$	$2{,}47$	$4,34$	$6,12$

Tabla 1: Pruebas que se realizaron para medir el desempeño del código distribuido. Los tiempos se encuentran en segundos. Las mismas se encuentran identificadas como X nodos $\times$ Y procesos $MPI \times Z$ hilos OpenMP, donde la cantidad total de procesos de cálculo es $X \times Y \times Z$. Además se presentan los tiempos de las comunicaciones realizadas entre procesos en un paso temporal (*"VecScatter en* Δt*"*) y el porcentaje del tiempo de cálculo que este representa (*"VecScatter en* Δt *(%)"*). En el *Speed Up* (I) se realiza el cociente entre el tiempo total de la simulación (incluyendo lectura y escritura de los datos) con respecto al tiempo total del la simulación $1 \times 1 \times 12$. En el segundo *Speed Up* (II) se realiza el cociente entre el tiempo de avance de un paso temporal de la simulación (sin considerar la lectura y escritura de los datos) y el tiempo de avance de un paso de tiempo de la simulación $1 \times 1 \times 12$.

Aunque una mayor distribución de los datos nos brinda una velocidad de cálculo más alta, analizando el porcentaje que representa el tiempo de comunicaciones de los datos con respecto al tiempo total de cálculo (*"VecScatter en* Δt *(%)"* en la tabla 1) notamos que las comunicaciones toman mayor importancia a medida que utilizamos más nodos de cálculo, representando en la simulación $8 \times 12 \times 1$ un tercio del tiempo de cálculo. Es lógico pensar que existirá un punto en el que la mayoría del tiempo de cálculo sea debido a las comunicaciones entre procesos, por lo que trabajos futuros se centrarán en la disminución de la cantidad de comunicaciones.

Además del aumento en la velocidad de cálculo que nos permite la paralelización de nuestro código mediante *MPI* y *PETSc*, este nuevo paradigma de cálculo (memoria distribuida) nos per-

mitirá realizar simulaciones de flujos con mayores números de Reynolds que requieren mayores resoluciones, anteriormente limitadas por la memoria de una computadora actual (aproximadamente 64GB).

4.3. Validación del código distribuido

Para la validación del código realizamos la comparación de los archivos binarios de salida (campo escalar y campo vectorial de velocidades) del código distribuido con los del código secuencial, determinando el máximo valor absoluto de la diferencia entre los dos, y el rms de los valores absolutos de las diferencias:

$$\varepsilon_{max_f} = \max \left| f_{sec}(x,y,z) - f_{dist}(x,y,z) \right|, \quad con\, x = 1, nx;\ y = 1, ny;\ z = 1, nz, \tag{72}$$

$$\varepsilon_{rms_f} = \sqrt{\frac{\sum_{x=1}^{nx}\sum_{y=1}^{ny}\sum_{z=1}^{nz} |f_{sec}(x,y,z) - f_{dist}(x,y,z)|^2}{nx \times ny \times nz}}, \tag{73}$$

donde f puede ser el campo escalar ρ o las componentes u, v o w del campo vectorial de velocidades.

Se logró obtener una diferencia máxima para el campo escalar de $\varepsilon_{max_\rho} = 4{,}03 \times 10^{-14}$ y el rms de $\varepsilon_{rms_\rho} = 2{,}8 \times 10^{-19}$. Para el campo de velocidades se logró obtener una diferencia máxima de $\varepsilon_{max_{\mathbf{u}}} = 0{,}3 \times 10^{-14}$ y el rms de $\varepsilon_{rms_{\mathbf{u}}} = 0{,}66 \times 10^{-19}$. Estos valores se encuentran en el orden de los errores de cálculo numérico computacional (error de máquina).

5. CONCLUSIONES

Debido a la limitación que significaba para nuestra investigación en flujos turbulentos realizar las simulaciones en una sola computadora, implementamos la paralelización del código pseudo-espectral bajo el paradigma de memoria distribuida. Para esto utilizamos la implementación *"MPICH"* del estandar *"MPI"* ("Message passing interface") junto al paquete de herramientas extensibles y portables para la computación científica *"PETSc"*. Además realizamos la programación del código en lenguaje *FORTRAN 90*, permitiendo el uso de reserva dinámica de memoria.

Se realizaron pruebas del código distribuido en el cluster del departamento de *Medicina Asistida por Computación Científica* del *Instituto de Ciencia y Tecnología de Brasil* (INCT-MACC), el cual cuenta con nodos interconectados mediante una red Infiniband. Los mismos son computadoras con procesadores *Intel® Xeon® CPU X5670 de 12 nucleos a 2.93GHz* y 36GB de memoria RAM.

En las pruebas realizadas se combinó el uso de *"OpenMP"*, *"MPI"* y se compararon todos los resultados para determinar la mejor estrategia de distribución del trabajo de cálculo.

Se pudo ver que existe un aumento en la velocidad de ejecución al utilizar sólo procesos *"MPI"*, debido principalmente a la disminución del tiempo de lectura (de $28{,}751$ seg. sólo con procesos *"MPI"* a $7{,}183$ seg. sólo con procesos *"OpenMP"*) y a la disminución del tiempo requerido por las comunicaciones.

Pudimos ver que no es la simulación con más número de procesos la que presenta los valores más altos de *Speed Up (I)*. Esto es debido a que al requerirse los datos distribuidos en una mayor cantidad de nodos, los tiempos de lectura y escritura son mayores, aumentando el tiempo

de ejecución total de la simulación. Sin embargo, analizando el *Speed Up (II)* en el cual sólo consideramos el tiempo de cálculo (sin considerar la lectura y escritura de los datos), podemos ver que la simulación con la mayor cantidad de procesos presenta los valores de *Speed Up* más altos. Este es un resultado prometedor ya que en una simulación real la escritura de los datos no se realiza en todos los pasos de tiempo y la lectura de los datos se realiza sólo en los casos donde queremos retomar una simulación.

Analizando el porcentaje que representa el tiempo de comunicaciones de los datos con respecto al tiempo total de cálculo pudimos concluir que aunque una mayor distribución de los datos nos brinda una velocidad de cálculo más alta, las comunicaciones toman mayor importancia a medida que utilizamos más nodos de cálculo, representando en la simulación con 8 nodos un tercio del tiempo de cálculo. Es lógico pensar que existirá un punto en el que la mayoría del tiempo de cálculo sea debido a las comunicaciones entre procesos, por lo que trabajos futuros se centrarán en la disminución de la cantidad de comunicaciones.

Además del aumento en la velocidad de cálculo que nos permite la paralelización de nuestro código mediante *"MPI"* y *"PETSc"*, este nuevo paradigma de cálculo (memoria distribuida) nos permitirá realizar simulaciones de flujos con mayores números de Reynolds que requieren mayores resoluciones, anteriormente limitadas por la memoria de una computadora actual (aproximadamente 64GB).

REFERENCIAS

MPI: The Complete Reference. Massachusetts Institute of Technology, , 1996.

Engineering and Scientific Subroutine Library for AIX Version 3 Release 3 - Guide and Reference. IBM, , 2010.

PETSc Users Manual - Revision 3.3. U.S. Department of Energy, , 2012.

Cantero M., Balachandar S., Cantelli A., Pirmez C., y Parker G. Turbidity current with a roof:direct numerical simulation of self-stratified turbulent channel flow driven by suspended sediment. *Journal of Geophysical Research - Oceans*, 114:C03008, 2009a.

Cantero M., Balachandar S., García M., y Bock D. Turbulent structures in planar gravity currents and their influence of the flow dynamics. *Journal of Geophysical Research - Oceans*, 113:C08018, 2008.

Cantero M., Balachandar S., García M., y Ferry J. Direct numerical simulations of planar and cyindrical density currents. *Journal of Applied Mechanics*, 73:923–930, 2006.

Cantero M., Balachandar S., y Parker G. Direct numerical simulation of stratification effects in a sediment-laden turbulent channel flow. *Journal of Turbulence*, 10(27):1–28, 2009b.

Cantero M., Lee J.R., Balachandar S., y García M. On the front velocity of gravity currents. *Journal of Fluid Mechanics*, 586:1–39, 2007.

Canuto C., Hussaini M., Quarteroni A., y Zang T. *Spectral Methods in Fluid Dynamics*. Springer-Verlag, New York, 1988. 557 pages.

Canuto C., Hussaini M., Quarteroni A., y Zang T. *Spectral Methods: Fundamentals in Single Domains*. Springer-Verlag, Berlin, 2006. 563 pages.

Deville M.O., Fischer P.F., y Mund E.H. *High-Order Methods for Incompressible Fluid Flow*. Cambridge University Press , , 2002.

Droegemeier K. y Wilhelmson R. Kelvin-Helmholtz instability in a numerically simulated thunderstorm outflow. *Bulletin of the American Meteorological Society*, 67(4):416–417, 1986.

Frigo M. *FFTW User's Manual for version 2.1.5*. Massachusetts Institute of Technology, , 2003.

García M. Turbidity currents. En L. Brekhovskikh, K. Turekian, K. Emery, y C. Tseng, editores,

Encyclopedia of Earth System Science, volumen 4, páginas 399–408. Academic Press, Inc., New York, 1992.

Härtel C., Carlsson F., y Thunblom M. Analysis and direct numerical simulation of the flow at a gravity-current head. Part 2. The lobe-and-cleft instability. *Journal of Fluid Mechanics*, 418:213–229, 2000a.

Härtel C., Kleiser L., Michaud M., y Stein C. A direct numerical simulation approach to the study of intrusion fronts. *Journal of Engineering Mathematics*, 32:103–120, 1997.

Härtel C., Meiburg E., y Necker F. Vorticity dynamics during the start-up phase of gravity currents. *Il Nuovo Cimento*, 22(6):823–833, 1999.

Härtel C., Meiburg E., y Necker F. Analysis and direct numerical simulation of the flow at a gravity-current head. Part 1. Flow topology and front speed for slip and no-slip boundaries. *Journal of Fluid Mechanics*, 418:189–212, 2000b.

Necker F., Härtel C., Kleiser L., y Meiburg E. High-resolution simulations of particle-driven gravity currents. *International Journal of Multiphase Flow*, 28:279–300, 2002.

Terez D. y Knio O. Numerical simulation of large-amplitude internal solitary waves. *Journal of Fluid Mechanics*, 362:53–82, 1998a.

Terez D. y Knio O. Numerical study of the collapse of an axisymmetric mixed region in a pycnoclyne. *Physics of Fluids*, 10(6):1438–1448, 1998b.

Zang T.A. On the rotation and skew-symmetric forms for incompressible flow simulations. *Applied Numerical Mathematics*, 7(1):27–40, 1991.