PEDECIBA Informática
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Reporte Técnico RT 11-16

Estudio de un modelo hidrodinámico sobre arquitecturas multi-core

Mariana Fernández Pablo Santero Mónica Fossati

Ernesto Dufrechou Pablo Ezzatti

2011

Estudio de un modelo hidrodinámico sobre arquitecturas multi-core
Fernández, Mariana; Santero, Pablo; Fossati, Mónica; Dufrechou, Ernesto; Ezzatti, Pablo.
ISSN 0797-6410
Reporte Técnico RT 11-16
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay, diciembre de 2011

Estudio de un modelo hidrodinámico sobre arquitecturas multi-core

Mariana Fernández[1], Pablo Santoro[1] , Mónica Fossati[1], Ernesto Dufrechou[2] y Pablo Ezzatti[2]

[1] Instituto de Mecánica de Fluidos e Ingeniería Ambiental, Facultad de Ingeniería, Universidad de la República, Uruguay

{mfernand,psantoro,mfossati}@fing.edu.uy

[2] Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay

{edufrechou,pezzatti}@fing.edu.uy

Resumen

El Río de la Plata posee una gran importancia para nuestro país tanto desde el punto de vista económico como desde el punto de vista de manejo ambiental del ecosistema. En este sentido, hace varios años en la Facultad de Ingeniería se viene trabajando en el desarrollo de herramientas que permiten la simulación y predicción del comportamiento del mismo.

Uno de los modelos numéricos base que se utiliza actualmente es el MOHID, que permite resolver las ecuaciones de movimiento mediante discretizaciones de volumenes finitos y el uso de técnicas de dominios encajados para aumentar la precisión en la simulación de zonas específicas. Uno de los problemas de este tipo de herramientas son los elevados costos computacionales.

Teniendo en cuenta lo expresado anteriormente este trabajo presenta un estudio de las principales características del modelo hidrodinámico MOHID. En particular se evalúa con especial atención el uso en el modelo de técnicas de paralelismo de memoria compartida capaces de explotar arquitecturas de hardware multi-core.

Palabras claves: modelo hidrodinámico, arquitecturas multi-core, MOHID

1. Introducción

En Uruguay, en el grupo de Dinámica de Estuarios del Instituto de Mecánica de los Fluidos e Ingeniería Ambiental (IMFIA), desde hace algunos años se está trabajando en la modelación numérica del Río de la Plata. El modelo hidrodinámico desarrollado originalmente, basado en las técnicas de diferencias finitas con una resolución espacial de 1 km, fue ampliamente utilizado para estudios de base y para aplicaciones ingenieriles. El modelo fue mejorado en 1998, llevándolo a una versión paralelizada de descomposición de dominios. Posteriormente, con el fin de obtener mejoras en diversas capacidades se paso a trabajar con estrategias de modelación basadas en distintos modelos bases, y es así que en la actualidad se trabaja principalmente sobre tres modelos. Estos son la familia RMA (RMA-2, RMA-10, RMA-11), el MOHID y el MARS.

En particular, en este grupo se utiliza el modelo tridimensional MOHID para estudiar a nivel global el campo hidrodinámico y la dinámica de sustancias en el Río de la Plata y el Frente Marítimo. El MOHID es un modelo de flujo a superficie libre desarrollado en el Instituto Técnico MARETEC de Portugal. Utiliza una discretización en volúmenes finitos que resuelve las ecuaciones tridimensionales de Navier Stokes con la aproximación hidrostática del campo de presiones en forma directa, esto es, sin descomponer los modos barotrópicos de los modos baroclínicos. Para estos trabajos se implementó el modelo en su versión tridimensional y fueron definidas como variables del sistema las tres componentes de la velocidad, la elevación de la superficie libre y la salinidad del agua. El MOHID se está utilizando para estudiar la hidrodinámica del Río de la Plata con gran resolución utilizando la metodología de modelos encajados. A través de esta metodología, es posible anidar grillas de resolución espacial creciente, forzando los modelos locales con resultados de aplicaciones de mayor escala. De esta forma, el modelo permite estudiar áreas cada vez más cercanas a la región de interés, a partir del traspaso de las condiciones de borde del modelo "padre". Por otra parte, se está desarrollando la modelación del Atlántico Sur acoplada con un modelo atmosférico, para generar y propagar las oscilaciones meteorológicas hacia el Río de la Plata. Como consecuencia de estos enfoques numéricos, se requiere de una gran capacidad de cálculo para resolver los sistemas de ecuaciones resultantes. Estos costos motivaron la inclusión de estrategias de computación de alto desempeño (HPC), en especial paralelismo bajo el paradigma de memoria compartida utilizando la herramienta OpenMP.

Basados en lo expresado en los párrafos anteriores este trabajo estudia las principales características del modelo MOHID, su aplicación para la modelación numérica hidrodinámica del Río de la Plata y el uso de OpenMP para su aceleración.

El documento se estructura de la siguiente manera. Primero se detallan las estrategias numéricas utilizadas por el MOHID. Luego, en la Sección 3 se describe la arquitectura de software del modelo. Posteriormente, se estudian las estrategias disponibles en el modelo para la resolución de sistemas lineales, etapa crucial en este tipo de modelos. Un estudio de la utilización de estrategias de paralelismo de memoria compartida en el modelo se ofrece en la Sección 5. Por último, se agregan dos anexos con los códigos de los métodos de resolución de sistemas lineales basados en el método de Thomas y una pequeña descripción de OpenMP.

2. El modelo MOHID

El desarrollo del modelo MOHID comenzó en 1985, pasando desde entonces a través de las actualizaciones y mejoras continuamente, debido a su uso en diferentes proyectos de investigación científica y proyectos de ingeniería. Inicialmente, el sistema de agua de modelado MOHID fue un modelo hidrodinámico bidimensional, llamado MOHID 2D (Neves, 1985). Este modelo fue utilizado para estudiar los estuarios y zonas costeras que utilizan un enfoque clásico de diferencias finitas.

En los años siguientes, un modelo de transporte bidimensional Euleriano así como uno Lagrangiano fueron incluidos en el modelo. La primera versión en tres dimensiones fue presentada por Santos (1995), y utilizó sistema doble de coordenadas verticales sigma. Esta versión se llamó MOHID 3D.

Las limitaciones de la doble coordenada sigma pusieron de manifiesto la necesidad de desarrollar un modelo que utilice un sistema de coordenadas verticales genérico, permitiendo al usuario elegir el tipo de coordenada vertical, dependiendo de la zona de estudio. Debido a esta necesidad se introdujo el concepto de volúmenes finitos con la versión 3D de malla por Martins (1999). En el modelo 3D de malla se incluye un modelo tridimensional de transporte Euleriano, un modelo tridimensional de transporte Lagrangiano (Leitão, 1996) y el modelo de calidad del agua de dimensión cero (Miranda, 1999). Desde la introducción del enfoque de volúmenes finitos, esta discretización fue la adoptada en el modelo MOHID.

En este capítulo se presentan las principales características del modelo MOHID, su modelo matemático y la resolución numérica del mismo. La información aquí presentada se basa fuertemente en la Tesis de doctorado de F. A. Martins (1999), como se mencionó anteriormente fue quien introdujo la técnica de volúmenes finitos al MOHID y realizó una documentación muy detallada y clara tanto sobre el modelo matemático subyacente así como su resolución numérica.

2.1. Características generales

El MOHID es un modelo de flujo a superficie libre tridimensional baroclínico basado en las ecuaciones de Navier-Stokes y con las aproximaciones de Boussinesq e hidrostática. La malla tridimensional está formulada con una aproximación de volúmenes finitos con coordenada vertical sigma o cartesiana u otras que permiten una buena simulación de los efectos topográficos. La discretización temporal implícita ADI (Alternating Direction Implicit) utiliza una grilla desfasada, permitiendo solucionar problemas de estabilidad que ocurren en métodos explícitos y obteniendo una resolución más simple debido al uso de matrices tridiagonales en el cálculo de la elevación de la superficie libre y velocidades horizontales. El término de la fuerza de Coriolis y el transporte horizontal se resuelven explícitamente, mientras que el modelo utiliza un algoritmo implícito para resolver los términos de presión y el transporte vertical.

El MOHID ha sido aplicado con éxito en varias zonas costeras y estuarios demostrando una gran capacidad para simular flujos con comportamientos complejos (Taboada et al., 1998; Montero, 1999; Martins et al., 1999; Villareal et al,

2002). En su versión actual el MOHID resuelve las ecuaciones aplicando el método de los volúmenes finitos y trabaja subdividido en más de 40 módulos. Cada módulo contiene determinada información e interactúa con los demás a través de la definición de flujos de información. Los principales módulos son el hidrodinámico, geometría, propiedades del agua, turbulencia, calidad de agua, sedimentos, entre otros.

2.2. Ecuaciones de movimiento

El modelo resuelve las ecuaciones de movimiento que representan el flujo en cuerpos de agua a superficie libre como estuarios, océanos, reservorios, etc. Estos sistemas pueden representarse a través del sistema de ecuaciones de aguas poco profundas. Este conjunto de ecuaciones parte de las ecuaciones primitivas tridimensionales para fluido incompresible y además se asume como válido el equilibrio hidrostático, y las aproximaciones de Boussinesq y de Reynolds. En este punto se desarrolla este sistema de ecuaciones que resuelve el MOHID.

Las ecuaciones de cantidad de movimiento para las velocidades horizontales del flujo medio en coordenadas cartesianas se presentan en las ecuaciones 1 y 2; las cuales expresan que la evolución temporal de las velocidades es igual al transporte advectivo, la fuerza de Coriolis, el gradiente de presiones y la difusión turbulenta. A partir de estas dos ecuaciones se determina el campo horizontal de velocidades.

$$\frac{\partial u}{\partial t} = -\frac{\partial(uu)}{\partial x} - \frac{\partial(uv)}{\partial y} - \frac{\partial(uw)}{\partial z} + fv - \frac{1}{\rho_0}\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left((\nu_H + \nu)\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left((\nu_H + \nu)\frac{\partial u}{\partial y}\right) + \frac{\partial}{\partial z}\left((\nu_t + \nu)\frac{\partial u}{\partial z}\right)$$

Ecuación 1

$$\frac{\partial v}{\partial t} = -\frac{\partial(vu)}{\partial x} - \frac{\partial(vv)}{\partial y} - \frac{\partial(vw)}{\partial z} - fu - \frac{1}{\rho_0}\frac{\partial p}{\partial y} + \frac{\partial}{\partial x}\left((\nu_H + \nu)\frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial y}\left((\nu_H + \nu)\frac{\partial v}{\partial y}\right) + \frac{\partial}{\partial z}\left((\nu_t + \nu)\frac{\partial v}{\partial z}\right)$$

Ecuación 2

Donde *u*, *v*, *w* son las componentes del vector velocidad en la dirección *x*, *y*, *z* respectivamente, *f* es el parámetro de Coriolis, ν_H y ν_t son las viscosidades turbulentas en las direcciones horizontal y vertical, ν es la viscosidad cinemática turbulenta y *p* es la presión.

La velocidad vertical *w* se calcula a partir de la integración en la dirección vertical de la ecuación de continuidad para fluido incompresible (Ecuación 3), entre el fondo y la profundidad *z* donde se calcula *w* (Ecuación 4):

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

Ecuación 3

$$w(z) = -\frac{\partial}{\partial x}\left(\int_{-h}^{z} u dz\right) - \frac{\partial}{\partial y}\left(\int_{-h}^{z} v dz\right)$$

Ecuación 4

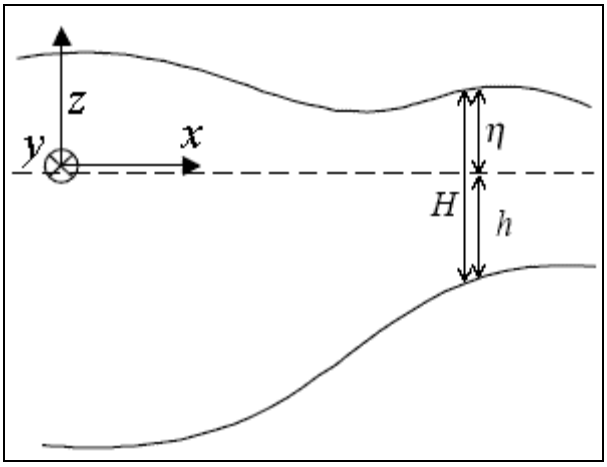

Figura 1: Sistema de coordenadas utilizado.

Para el cálculo de la elevación de la superficie libre η se utiliza la ecuación 5, que se obtiene integrando la ecuación de continuidad sobre toda la columna de agua e imponiendo las condiciones de borde cinemáticas en el fondo y en la superficie. El sistema de referencia utilizado se presenta en la Figura 1, donde H es la profundidad total desde el fondo a la superficie, h es la profundidad del fondo respecto a un nivel fijo de referencia y η es la altura de la superficie libre.

$$\int_{-h}^{\eta}\frac{\partial u}{\partial x}dz+\int_{-h}^{\eta}\frac{\partial v}{\partial y}dz+w(\eta)-w(-h)=0 \qquad w(-h)=0 \qquad w(\eta)=\frac{\partial \eta}{\partial t}+u\frac{\partial \eta}{\partial x}+v\frac{\partial \eta}{\partial y}$$

$$\frac{\partial \eta}{\partial t}=-\frac{\partial}{\partial x}\left(\int_{-h}^{\eta}udz\right)-\frac{\partial}{\partial y}\left(\int_{-h}^{\eta}vdz\right) \qquad \textbf{Ecuación 5}$$

El modelo utiliza la aproximación hidrostática ya que las escalas horizontales en los cuerpos de agua estuarinos u oceánicos son mucho mayores que las verticales. Se supone entonces que las aceleraciones verticales son pequeñas respecto a los otros términos que intervienen en la ecuación de cantidad de movimiento en la dirección vertical, y la misma se reduce al balance hidrostático que se presenta en la ecuación 6, donde g es la gravedad y ρ la densidad. Sustituyendo en esa ecuación la presión por la presión atmosférica p_{atm}, aplicando la descomposición de Boussinesq en donde la densidad ρ se divide en una densidad constante de referencia ρ_0 y una fluctuación de ese valor ρ', e integrando en la dirección vertical entre la superficie libre η y la profundidad z donde se calcula la presión, se obtiene la Ecuación 7. Esta ecuación relaciona la presión en cualquier profundidad con la presión atmosférica en la superficie libre, el nivel de agua y la presión integrada entre ese nivel y la superficie.

$$\frac{\partial p}{\partial z}+\rho g=0 \qquad \textbf{Ecuación 6}$$

$$p(z)=p_{atm}+g\rho_0(\eta-z)+g\int_{z}^{\eta}\rho' dz \qquad \textbf{Ecuación 7}$$

A partir de la expresión determinada para la presión (Ecuación 7), es posible determinar el gradiente de presiones horizontales en las direcciones x e y. El gradiente de presiones resulta igual a la suma del gradiente de presiones

atmosférico, el gradiente de la elevación de la superficie libre (gradiente de presiones barotrópico) y el gradiente de la distribución de densidades (gradiente de presiones baroclínico). Los gradientes de presiones que se presentan en las ecuaciones 8 y 9 se sustituyen en las ecuaciones 1 y 2 respectivamente, para resolver el campo horizontal de velocidades.

$$\frac{\partial p}{\partial x} = \frac{\partial p_{atm}}{\partial x} + g\rho_0 \frac{\partial \eta}{\partial x} + g\int_z^\eta \frac{\partial \rho'}{\partial x} dz \qquad \textbf{Ecuación 8}$$

$$\frac{\partial p}{\partial y} = \frac{\partial p_{atm}}{\partial y} + g\rho_0 \frac{\partial \eta}{\partial y} + g\int_z^\eta \frac{\partial \rho'}{\partial y} dz \qquad \textbf{Ecuación 9}$$

El modelo también resuelve las ecuaciones de transporte para la salinidad y la temperatura. El transporte de sal se expresa con la Ecuación 10, la cual expresa que la variación temporal de la salinidad es el balance entre al transporte advectivo por el flujo medio, la mezcla turbulenta y los aportes de posibles fuentes o descargas de sal expresados con F_s.

$$\frac{\partial S}{\partial t} = -\frac{\partial (uS)}{\partial x} - \frac{\partial (vS)}{\partial y} - \frac{\partial (wS)}{\partial z} + \frac{\partial}{\partial x}\left(\nu_H' \frac{\partial S}{\partial x}\right) + \frac{\partial}{\partial y}\left(\nu_H' \frac{\partial S}{\partial y}\right) + \frac{\partial}{\partial z}\left(\nu_t' \frac{\partial S}{\partial z}\right) + F_S$$

Ecuación 10

Donde S es la salinidad, ν'_H y ν'_t son los coeficientes de difusividad horizontal y vertical de sal, respectivamente.

La densidad ρ se calcula en función de la temperatura (T) y de la salinidad a través de una ecuación de estado simplificada:

$$\rho = \frac{5890 + 38T - 0.375T^2 + 3S}{(1779.5 + 11.25T - 0.0745T^2) - (3.8 + 0.01T)S + 0.698(5890 + 38T - 0.375T^2 + 3S)}$$

Ecuación 11

En definitiva el modelo resuelve el sistema formado por las dos ecuaciones de cantidad de movimiento horizontal (ecuaciones 1 y 2), la ecuación de continuidad (Ecuación 4), la ecuación de superficie libre (Ecuación 5), la ecuación de transporte de sal (Ecuación 10) y la ecuación de densidad (Ecuación 11), para determinar el campo de velocidades, la elevación de la superficie libre, la salinidad y la densidad en cada punto del dominio. En el caso que se considere también como variable a la temperatura, se agrega la ecuación de transporte de temperatura al sistema de ecuaciones a resolver.

2.3. Aplicación del método de los volúmenes finitos

Para determinar las variables de interés el modelo resuelve el sistema de ecuaciones que mencionamos anteriormente, ecuaciones que derivan de la conservación de la cantidad de movimiento y conservación de la masa. El modelo MOHID utiliza el método de los volúmenes finitos para discretizar las ecuaciones que gobiernan el flujo. La idea básica de este método es partir de la forma integral de las ecuaciones de conservación como punto de base. De forma general la ecuación de conservación para una escalar α con una fuente Q en un volumen de control V se escribe de la forma siguiente:

$$\frac{\partial}{\partial t}\int_V \alpha dV + \iint_S \vec{F}\vec{n}dS = \int_V QdV \qquad \textbf{Ecuación 12}$$

Donde F representa el flujo del escalar a través de la superficie frontera S del volumen considerado y n la normal exterior de la superficie. Las ecuaciones de Navier Stokes y las ecuaciones de transporte de salinidad y temperatura pueden escribirse de esa forma eligiendo convenientemente las variables α y Q. El volumen de control considerado se divide en un determinado número de pequeños volúmenes de control denominados celdas y el método resuelve los balances globales de las propiedades transportadas en dichos volúmenes aproximándolos por los flujos a través de las caras de dichas celdas. La ecuación se aplica en cada celda del dominio de cálculo y al sumar las ecuaciones en todas las celdas se obtiene la ecuación global de conservación, ya que las integrales de superficie en las caras de las celdas interiores se anulan. Es decir que una propiedad intrínseca del método es que se cumplen las leyes de conservación a nivel global y esta es su principal ventaja (Ferziger, 2002).

Las ecuaciones de conservación se discretizan por el método de los volúmenes finitos considerando por ejemplo una celda de geometría genérica y que las propiedades sean homogéneas en el interior de la celda. De forma discreta, aplicando la Ecuación 12 a un volumen de control V_{ijk} se obtiene la Ecuación 13, en donde los flujos pueden representarse de la forma:

$$\frac{\partial}{\partial t}\left(\alpha_{ijk}V_{ijk}\right) = -\sum_{m=1}^{N_{faces}} \vec{F}_m \cdot \vec{n}S_m + Q_{ijk}V_{ijk}$$

$$\vec{F}_m \cdot \vec{n}S_m = \alpha_m \vec{u}_m \cdot \vec{n}S_m$$

Ecuación 13

Siendo u_m el valor representativo del campo de velocidades en la cara m y α_m el valor representativo de la propiedad que atraviesa la cara. La forma de calcular estos dos valores define la discretización espacial utilizada. Si la cara fuera perpendicular a uno de los ejes de coordenadas entonces la componente de la velocidad a lo largo de dicho eje será la única componente responsable del flujo volumétrico a través de esa cara. En la discretización de las celdas que utiliza el modelo MOHID se cumple esto en la dirección horizontal, ya que la malla horizontal es de tipo cartesiana, siendo las caras laterales de las celdas verticales y ortogonales a dichos ejes, como se observa en la Figura 2. Esta geometría permite una reducción considerable del número de interpolaciones que se deben realizar, ya que solamente la componente de la velocidad calculada en esa cara contribuye al flujo volumétrico.

A continuación se introduce para cada una de las ecuaciones del sistema, la discretización utilizada para cada uno de los términos de las mismas. En el punto 2.4 se desarrolla en conjunto la discretización del sistema.

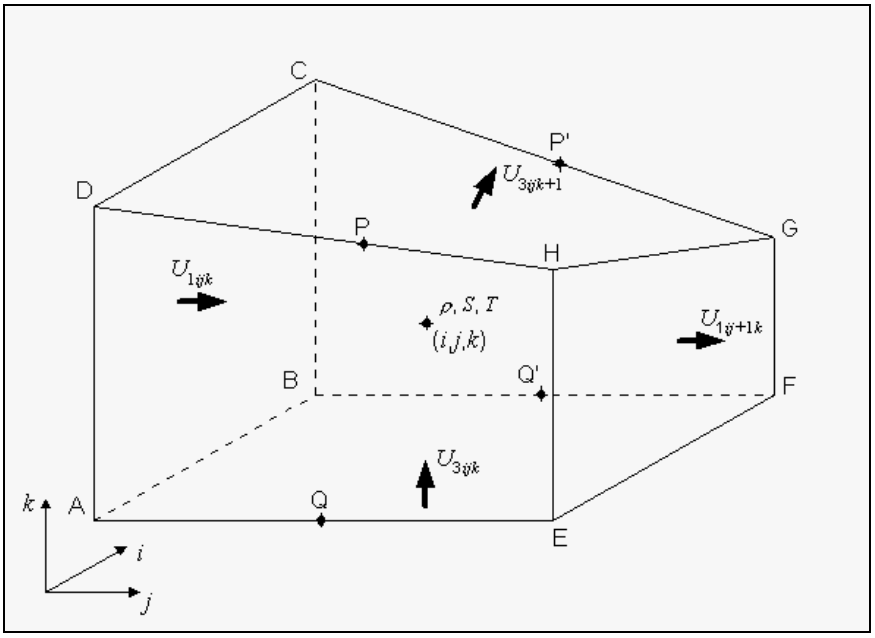

Figura 2: Celda de cálculo tridimensional del tipo C desfasada utilizada por el MOHID para el cálculo de escalares. En este caso U_1 representa la velocidad horizontal según x y U_3 la componente vertical de la velocidad.

2.3.1. Ecuación de continuidad

La ecuación de continuidad se expresa a través de la Ecuación 12 cuando la variable transportada α es igual a la densidad ρ y el término fuente Q es igual a cero. Los flujos $\vec{F} \cdot \vec{n} S$ de esta propiedad a través de las caras de cada volumen de control representan caudales másicos. En el volumen de cálculo utilizado en el MOHID, las propiedades escalares como la densidad se calculan en el centro del volumen de control. Será necesario entonces determinar el valor de la propiedad transportada en las caras de dicho volumen.

Para resolver el primer término de la ecuación de balance, es decir la variación temporal de la propiedad en la celda de control, el modelo utiliza la aproximación en diferencias progresiva en el tiempo:

$$\frac{\partial}{\partial t}\left(\alpha_{ijk} V_{ijk}\right) = \frac{\alpha_{ijk}^{n+1} V_{ijk}^{n+1} - \alpha_{ijk}^{n} V_{ijk}^{n}}{\Delta t} \qquad \textbf{Ecuación 14}$$

2.3.1.1. Flujos convectivos horizontales

Cuando la propiedad α transportada por el flujo no se calcula sobre la cara que se está considerando es necesario tomar hipótesis adicionales que van a determinar las características particulares del método numérico a utilizar. Por ejemplo para calcular la propiedad genérica en la cara ABCD, denominada $a_{ij-1/2k}$ para ser consistente con la notación, se puede utilizar una discretización tipo central, obteniéndose dicho valor por la media ponderada para el caso general de espaciamiento de celda variable. Si en cambio se utilizara una discretización del tipo "upwind" se tendría:

$$\alpha_{\text{ABCD}} = \frac{Fxz_{ijk} + \left|Fxz_{ijk}\right|}{2}\alpha_{ij-1k} + \frac{Fxz_{ijk} - \left|Fxz_{ijk}\right|}{2}\alpha_{ij-1k}$$ **Ecuación 15**

siendo $Fxz_{ijk} = U_{1_{ijk}} S_{\text{ABCD}}$ el caudal volumétrico por la cara ABCD. La notación *Fxz* expresa con la primera letra la dirección del flujo que se está calculando, y con la segunda letra la celda de cálculo a la que se refiere. La primer letra puede ser entonces igual a cada una de las tres direcciones, es decir *x*, *y* ó *z*, y la segunda letra puede ser *z*, *u* ó *v*. La letra *z* representa la celda de cálculo de escalares, mientras que la *u* y la *v* representan las celdas de cálculo de las respectivas velocidades horizontales, que son diferentes ya que el MOHID utiliza una celda de cálculo desfasada.

Otra opción para el cálculo del flujo es una formulación genérica que pondera entre las dos discretizaciones anteriores a través de un parámetro γ:

$$\left(\vec{F}\cdot\vec{n}S\right)_{\text{ABCD}} = -\gamma\left(\frac{Fxz_{ijk} + \left|Fxz_{ijk}\right|}{2}\alpha_{ij-1k} + \frac{Fxz_{ijk} - \left|Fxz_{ijk}\right|}{2}\alpha_{ijk}\right) - \left(1-\gamma\right)\alpha_{ij-1/2k}Fxz_{ijk}$$

Ecuación 16

Una formulación similar a la anterior se utiliza para resolver el flujo por las caras EHGF, ADHE y BFGC, utilizando en estos dos últimos casos los caudales volumétricos *Fyz* en lugar de los *Fxz* y los subíndices correspondientes.

2.3.1.2. Flujos convectivos verticales

Como se mencionó anteriormente la ecuación de continuidad es utilizada para determinar la velocidad vertical. La ecuación de continuidad se resuelve entonces considerando el flujo vertical como la variable dependiente:

$$\left(\vec{F}\cdot\vec{n}S\right)_{\text{DHGC}} = -\sum_{m=1}^{N-1_{faces}} \vec{F}_m\cdot\vec{n}S_m - \frac{\alpha_{ijk}^{n+1}V_{ijk}^{n+1} - \alpha_{ijk}^{n}V_{ijk}^{n}}{\Delta t}$$ **Ecuación 17**

Para el caso general en el cual la cara DHGC sea inclinada, el flujo total que atraviesa la cara y que se calcula a través de la ecuación anterior tendrá componentes en las tres direcciones:

$$\left(\vec{F}\cdot\vec{n}S\right)_{\text{DHGC}} = Fzz_{ijk+1}\cdot\alpha = \left(F_1n_1 + F_2n_2 + F_3n_3\right)S_{\text{DHGC}}\cdot\alpha$$ **Ecuación 18**

2.3.2. Ecuación de transporte de salinidad

La ecuación de transporte de salinidad se expresa de la forma de la Ecuación 12 con la propiedad α igual a la salinidad *S* (o la temperatura en el caso que se simule el transporte de temperatura) y el término fuente *Q* igual a cero. Este término fuente puede sin embargo utilizarse en las celdas de superficie para imponer evaporación o precipitación; siendo en este caso simulados dichos

procesos como una variación de salinidad y no como un ingreso o salida de agua. Las consideraciones anteriores respecto a la discretización de los flujos convectivos horizontales se utilizan también para el transporte de sal. En este caso además se deben calcular los términos difusivos.

2.3.2.1. Flujos difusivos horizontales

Para cada cara *m* de la celda de cálculo el flujo difusivo (*Fdh*) está dado por el vector presentado en la Ecuación 19. Para las cuatro caras verticales que tiene la celda que utiliza el modelo, la única componente del flujo que contribuye al transporte es la perpendicular a cada cara. Por ejemplo en la cara ABCD se tiene que $\vec{n}_{ABCD} = -1\vec{e}_1$ y considerando una discretización tipo central para la derivada, a partir de los valores de la propiedad en los centros de las celdas de cada lado de la cara, se obtiene la discretización del flujo difusivo presentada en la Ecuación 20.

$$\vec{F}dh_m = -\frac{1}{S_m}\int_{S_m} K\vec{\nabla}\alpha\, dS = -\frac{1}{S_m}\int_{S_m} K_H\frac{\partial\alpha}{\partial x}\vec{e}_1 + K_H\frac{\partial\alpha}{\partial y}\vec{e}_2 + K_V\frac{\partial\alpha}{\partial z}\vec{e}_3\, dS$$

Ecuación 19

$$\vec{F}dh_{\mathrm{ABCD}}\cdot\vec{n}S_{\mathrm{ABCD}} = -\int_{S_{\mathrm{ABCD}}} K\vec{\nabla}\alpha\cdot\vec{n}\, dS = \int_{S_{\mathrm{ABCD}}} K_H\frac{\partial\alpha}{\partial x}dS = K_{H_{\mathrm{ABCD}}}\frac{\alpha_{ijk}-\alpha_{ij-1k}}{\Delta x}S_{\mathrm{ABCD}}$$

Ecuación 20

2.3.2.2. Flujos convectivos verticales

Como el modelo utiliza una coordenada vertical genérica, en las celdas que utiliza las caras superior e inferior son oblicuas en relación a los ejes de coordenadas. Los flujos convectivos y difusivos a través de esas caras tienen entonces componentes en las tres direcciones espaciales. En este caso el flujo convectivo vertical puede discretizarse idéntico a los flujos convectivos horizontales:

$$\left(\vec{F}c\cdot\vec{n}S\right)_{\mathrm{DHGC}} = \gamma\left(\frac{Fzz_{ijk+1}+\left|Fzz_{ijk+1}\right|}{2}\alpha_{ijk} + \frac{Fzz_{ijk+1}-\left|Fzz_{ijk+1}\right|}{2}\alpha_{ijk+1}\right) + \left(1-\gamma\right)\alpha_{ijk+1/2}Fzz_{ijk+1}$$

Ecuación 21

Siendo *Fzz* el caudal volumétrico en la dirección "vertical" obtenido a través de la ecuación de continuidad discretizada presentada anteriormente (Ecuación 17), y que engloba las componentes en las tres direcciones. Esta discretización equivale a considerar que independiente de la dirección del flujo por la cara DHGC, la propiedad transportada siempre será una función de α_{ijk} y α_{ijk+1}.

2.3.2.3. Flujos difusivos verticales:

Para resolver el flujo difusivo por la cara DHGC es necesario calcular las tres componentes del flujo por separado. En la Figura 3 se muestra un corte por el plano *x z* de la celda presentada en la Figura 2.

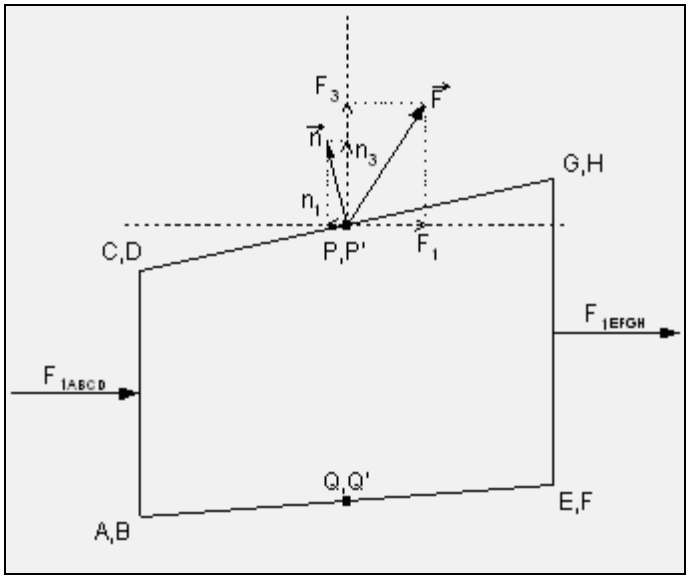

Figura 3: Flujo vertical por la cara superior de la celda.

El flujo difusivo en la dirección vertical $F_3 n_3 S_{\text{DHGC}}$ se discretiza de forma semejante a los flujos difusivos horizontales:

$$F_3 n_3 S_{\text{DHGC}} = -\int_{S_{\text{DHGC}}} K_V \frac{\partial \alpha}{\partial z} dS = K_{V_{\text{DHGC}}} \frac{\alpha_{ijk+1} - \alpha_{ijk}}{\Delta z} S_{\text{DHGC}} \qquad \textbf{Ecuación 22}$$

Los flujos difusivos horizontales que derivan de este término de flujo difusivo en la dirección vertical $F_1 n_1 S_{\text{DHGC}}$ y $F_2 n_2 S_{\text{DHGC}}$ en los modelos de coordenada sigma en general se desprecian porque introducen un error excesivo. Para otros tipos de coordenadas verticales con el método de los volúmenes finitos es posible calcular estos términos, calculando el flujo sobre un área vertical modificada de la celda que tenga en cuenta la inclinación de la misma. Sin embargo la contribución que se obtiene por estos flujos es muy pequeña y justifica entonces la no consideración de los mismos.

2.3.3. Ecuación de transporte de cantidad de movimiento

Las tres componentes de la ecuación de conservación de cantidad de movimiento se representan de forma global (Ecuación 12) son:

Para la componente *x*: $\alpha = \rho.u$; $Q = \rho.f.v$

Para la componente *y*: $\alpha = \rho.v$; $Q = -\rho.f.u$

Para la componente *z* $\alpha = 0$; $Q = -\rho.g$

En este caso los flujos $\vec{F}$ de la ecuación de balance representan flujos de cantidad de movimiento por unidad de volumen, por lo que se interpretan como fuerzas. De esta forma engloban todas las fuerzas ejercidas sobre el fluido, es decir las fuerzas viscosas, las fuerzas de presión y las fuerzas convectivas (flujo convectivo de cantidad de movimiento). La discretización de los flujos convectivos y difusivos es idéntica a la que se presentó anteriormente. Sin embargo como se

utiliza una malla descentrada la ubicación de las propiedades en relación a la celda es diferente a los casos anteriores, según se muestra en la Figura 4. Esto obliga a realizar interpolaciones del campo de velocidades para el cálculo del caudal volumétrico en las caras.

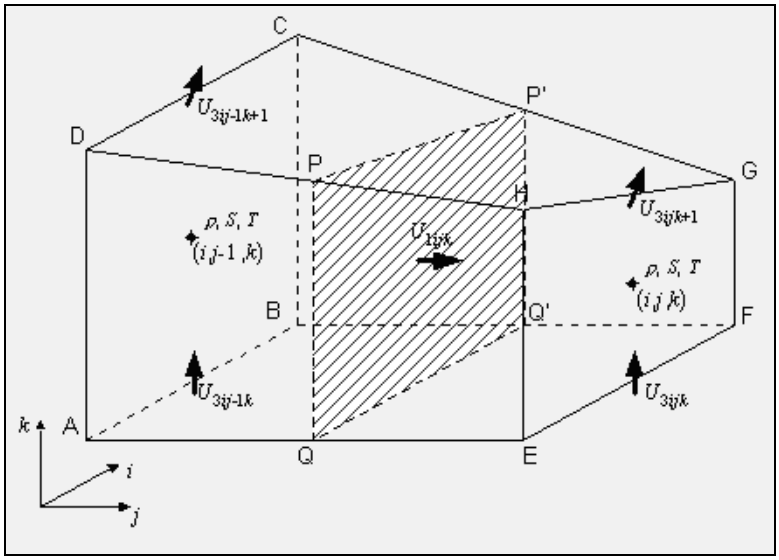

Figura 4: Celda tridimensional del tipo C para el cálculo de la velocidad *u*.

Para el caso particular de la malla tipo C los flujos por las caras de las celdas se calculan directamente a partir de los valores de las velocidades, estando garantido que su divergencia es nula. Para las celdas de velocidades los flujos en las caras se calculan a través de la media del campo de velocidades en la cara correspondiente. La no divergencia del campo de velocidades que se obtienen con estas operaciones de media es importante pues garantiza la conservación de los segundos momentos (u^2), y por eso de la energía cinética. Considerando la celda representativa de cálculo de u_{ij} que se muestra en la Figura 5, el campo convectivo que se utilizará para el transporte de las propiedades será el de las caras de la celda y se denomina u^*. Sus valores se obtienen a partir del campo u por promediación ponderada que a su vez deben garantizar la no divergencia del campo u^*.

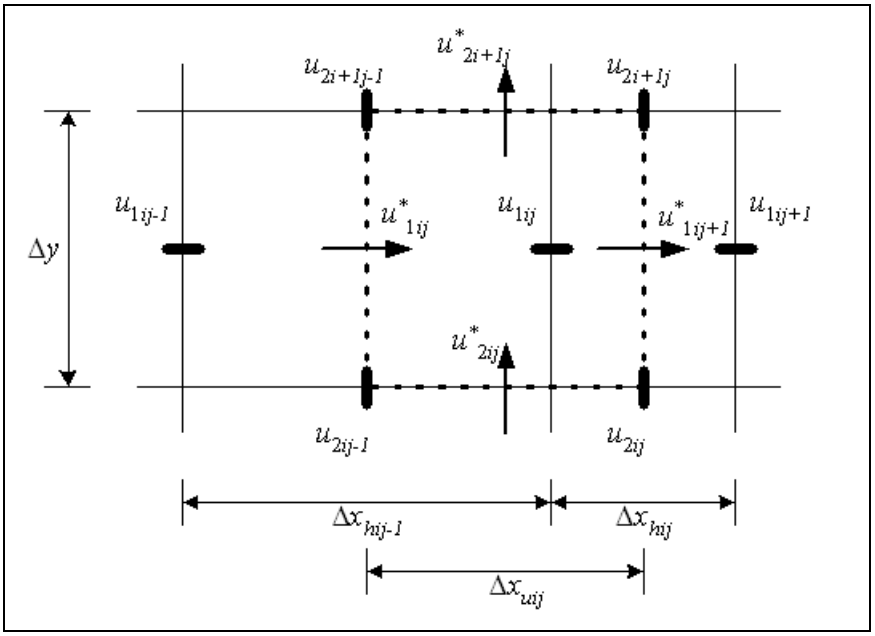

Figura 5: Flujos en la celda de velocidades u_1.

Los valores medios de las velocidades en las caras de la celda se pueden escribir en forma general, siendo α los parámetros de ponderación según la Ecuación 23. En dicha ecuación (y en la Figura 5) las variables u_1, u_2 y u_3 representan las componentes de la velocidad *u*, *v* y *w*, respectivamente.

$$u^*_{1_{ijk}} = \alpha_{u_1} u_{1_{ijk}} + (1-\alpha_{u_1}) u_{1_{ij-1k}}$$
$$u^*_{2_{ijk}} = \alpha_{u_2} u_{2_{ijk}} + (1-\alpha_{u_2}) u_{2_{ij-1k}}$$
$$u^*_{3_{ijk}} = \alpha_{u_3} u_{3_{ijk}} + (1-\alpha_{u_3}) u_{3_{ij-1k}}$$

Ecuación 23

A partir de la condición de divergencia nula, $\nabla \cdot \vec{u}^* = 0$, se obtienen las relaciones entre los parámetros de ponderación y se determinan las velocidades u^* en las caras. En el caso del método de volúmenes finitos, se utilizan como variables dependientes los flujos y no las velocidades. Para que los flujos a través de las caras de la celda de velocidades sean conservativos es necesario que las promediaciones realizadas a los flujos sean idénticas a las efectuadas a las velocidades.

A continuación se presenta la discretización por volúmenes finitos utilizada en el MOHID para representar las fuerzas representadas por los diferentes términos en las ecuaciones de cantidad de movimiento.

2.3.3.1. Fuerzas de presión

La fuerza de presión ejercida sobre una cara *m* de la celda se expresa según la Ecuación 24, y representa un vector de dirección normal a la cara y sentido contrario a la normal saliente a la misma. En la ecuación de transporte de cada componente de cantidad de movimiento, el flujo asociado a la fuerza de presión solo considera la componente de la fuerza en esa dirección.

$$\vec{\pi}_m = -\left(\frac{1}{S_m}\int_{S_m} p\,dS\right)\vec{n} = -P_m\,\vec{n}$$

Ecuación 24

El flujo de la fuerza de presión (*Fp*) correspondiente para la ecuación de la componente *x* de cantidad de movimiento es entonces:

$$\vec{F}p_m = -\pi_{1_m}\vec{e}_1 = P_m\,n_{1_m}\vec{e}_1$$

Ecuación 25

La sumatoria a lo largo de todas las caras de la celda, teniendo en cuenta que las normales de las caras verticales coinciden con los ejes de coordenadas, resulta:

$$\sum_{m=1}^{N_{faces}} \vec{F}p_m \cdot \vec{n}S_m = \left(Pn_1S\right)_{\mathrm{DHGC}} + \left(Pn_1S\right)_{\mathrm{AEFB}} - \left(PS\right)_{\mathrm{ABCD}} + \left(PS\right)_{\mathrm{EFGH}}$$

Ecuación 26

La presión media *P* en las caras de las celdas está dada por la integración de la presión en toda la superficie de cada cara. La presión para cada punto se expresa según la Ecuación 7 presentada anteriormente, y se obtiene la Ecuación 27[1].

$$P_m = \frac{1}{S_m}\int_{S_m}\left(p_{btrop} + p_{bclin}\right)dS = \frac{1}{S_m}\int_{S_m}\left[P_{atm} + \rho_0 g(\eta - x_3) + g\int_{x_3}^{\eta}\rho' dx_3\right]dS$$

Ecuación 27

Presión barotrópica:

En el método de los volúmenes finitos la elevación de la superficie libre se considera constante dentro de cada celda de la malla horizontal. Con el uso de la malla descentrada eso significa que todos los puntos de la cara ABCD y todos los puntos de las caras DPP'C y AQQ'B tienen la misma elevación η_{ij-1}; y los puntos de las caras EFGH, PHGP' y QEFQ' tienen η_{ij}. La sumatoria del término de presión barotrópica en todas las caras de la celda es por eso equivalente a la sumatoria de los dos lados de la superficie PP'Q'Q de la Figura 4:

$$\sum_{m=1}^{N_{faces}} \vec{F}bt_m \cdot \vec{n}S_m = \left\{\left[P_{atm_{ij}} + \rho_0 g\left(\eta_{ij} - \hat{x}_{3_{\mathrm{PP'Q'Q}}}\right)\right] - \left[P_{atm_{ij-1}} + \rho_0 g\left(\eta_{ij-1} - \hat{x}_{3_{\mathrm{PP'Q'Q}}}\right)\right]\right\}S_{\mathrm{PP'Q'Q}}$$

$$\sum_{m=1}^{N_{faces}} \vec{F}bt_m \cdot \vec{n}S_m = \left[\left(P_{atm_{ij}} - P_{atm_{ij-1}}\right) + \rho_0 g\left(\eta_{ij} - \eta_{ij-1}\right)\right]S_{\mathrm{PP'Q'Q}}$$

Ecuación 28

Siendo $\hat{x}_{3_{\mathrm{PP'Q'Q}}}$ la coordenada vertical del centro de gravedad de la superficie PP'Q'Q.

Presión baroclínica:

La componente baroclínica de la fuerza de presión está dada por la Ecuación 29. Tomando como ejemplo la celda $U_{ijkmax-1}$ de la Figura 6 e integrando para la cara

[1] En esta ecuación y en algunas posteriores se representa con x_3 la coordenada *z*.

ABCD se obtiene la Ecuación 30 (siendo $\hat{x}_{3_{ABCD}}$ la coordenada del centro de gravedad de la cara ABCD).

$$\sum_{m=1}^{N_{faces}} \vec{F}bc_m \cdot \vec{n}S_m = \sum_{m=1}^{N_{faces}} \left(\frac{g}{S_m} \int_{S_m} \int_{x_3}^{\eta} \rho' dx_3 \, dS \right) \vec{e}_1 \cdot \vec{n}S_m$$

Ecuación 29

$$\vec{F}bc_{ABCD} \cdot \vec{n}S = -g \int_{ABCD} \int_{x_3}^{\eta} \rho' dx_3 \, dS = -g \int_{ABCD} \left[\rho'_{ij-1kmax} \left(\eta_{ij-1} - x_{3_{CD}} \right) + \int_{x_3}^{x_{3_{CD}}} \rho'_{ij-1kmax-1} dx_3 \right] dS =$$

$$-gS_{ABCD} \left[\rho'_{ij-1kmax} \left(\eta_{ij-1} - x_{3_{CD}} \right) + \rho'_{ij-1kmax-1} \left(x_{3_{CD}} - \hat{x}_{3_{ABCD}} \right) \right]$$

Ecuación 30

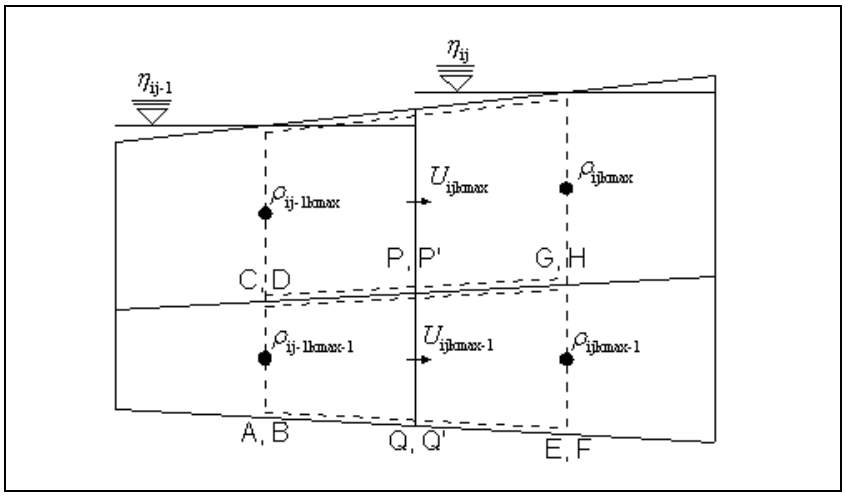

Figura 6: Corte Vertical de la celda de las velocidades.

Para las caras de la celda que no son verticales (las caras superior e inferior), la fuerza de presión baroclínica depende de la estructura de la estratificación existente sobre ellas (densidad y geometría de las celdas entre esa cara y la superficie). Para la cara AQQ'B por ejemplo, el flujo asociado a la componente de fuerza de presión baroclínica en la dirección *x* es:

$$\vec{F}bc_{AQQ'B} \cdot \vec{n}S = gn_1 \int_{AQQ'B} \left(\int_{x_3}^{\eta} \rho' dx_3 \right) dS = gn_1 \int_{AQQ'B} \left(\int_{x_{3DPP'C}}^{\eta_{ij-1}} \rho'_{ij-1kmax} dx_3 + \int_{x_3}^{x_{3DPP'C}} \rho'_{ij-1kmax-1} dx_3 \right) dS =$$

$$gn_1 \left(\rho'_{ij-1kmax} V_{kmax} + \rho'_{ij-1kmax-1} V_{kmax-1} \right)$$

Ecuación 31

donde V_{kmax} es el volumen de la mitad de la celda U_{ijkmax} que pertenece a la celda de $\rho_{ij\text{-}1kmax}$.

Esta forma de calcular el término baroclínico se vuelve excesivamente costosa desde el punto de vista computacional pues obliga al cálculo acumulado del peso de todas las celdas por encima de cada cara. Una forma simplificada de calcular el aporte de estas caras consiste en considerar que la estructura de la estratificación es idéntica a la que existe en los puntos de cálculo de la densidad. Con esa simplificación la presión baroclínica en un punto de coordenada vertical x_3 en la cara AQQ'B es:

$$p\left(x_{3_{AQQ'B}}\right) = p\left(x_{3_{ABCD}}\right) + g\int_{x_{3_{AQQ'B}}}^{x_{3_{ABCD}}} \rho'_{ij-1kmax-1}\,dx_3$$

Ecuación 32

donde x_{3ABCD} es un punto arbitrario de la cara ABCD. Tomando un punto de la arista CD y realizando la integración de la presión baroclínica a toda la cara AQQ'B se obtiene una expresión semejante a la Ecuación 30:

$$\vec{F}bc_{\mathrm{AQQ'B}} \cdot \vec{n}S = n_1 g \int_{\mathrm{AQQ'B}} \left[\rho'_{ij-1kmax}\left(\eta_{ij-1} - x_{3_{\mathrm{CD}}}\right) + \int_{x_{3_{\mathrm{AQQ'B}}}}^{x_{3_{\mathrm{CD}}}} \rho'_{ij-1kmax-1}\,dx_3 \right] dS =$$

$$n_1 g S_{\mathrm{AQQ'B}} \left[\rho'_{ij-1kmax}\left(\eta_{ij-1} - x_{3_{\mathrm{CD}}}\right) + \rho'_{ij-1kmax-1}\left(x_{3_{\mathrm{CD}}} - \hat{x}_{3_{\mathrm{AQQ'B}}}\right) \right]$$

Ecuación 33

De la misma forma, efectuando las integrales para las otras caras de la celda que pertenecen a la celda de $\rho_{\mathrm{ij}\text{-}1\mathrm{kmax}\text{-}1}$ (y por eso tienen elevación η_{ij-1}) se obtiene la Ecuación 34, donde se utilizó el hecho de que la coordenada vertical del centro de gravedad del conjunto de esas caras coincide con la coordenada $\hat{x}_{3_{PP'Q'Q}}$ del centro de gravedad de la superficie PP'Q'Q, y de que la superficie proyectada del conjunto de caras de la dirección *x* coincide también con el área de esa superficie.

$$\sum_{\substack{\mathrm{DPP'C}\\ \mathrm{ABCD}\\ \mathrm{AQQ'B}}} \vec{F}bc \cdot \vec{n}S = -gS_{\mathrm{PP'Q'Q}} \left[\rho'_{ij-1kmax}\left(\eta_{ij-1} - x_{3_{\mathrm{CD}}}\right) + \rho'_{ij-1kmax-1}\left(x_{3_{\mathrm{CD}}} - \hat{x}_{3_{\mathrm{PP'Q'Q}}}\right) \right]$$

Ecuación 34

Por otro lado, la contribución de las caras que pertenecen a la celda $\rho_{ijkmax-1}$ está dada por la Ecuación 35 y sumando, el flujo total de una celda *k* debido a la fuerza baroclínica se expresa por la Ecuación 36.

$$\sum_{\substack{\mathrm{PHGP'}\\ \mathrm{EFGH}\\ \mathrm{QEFQ'}}} \vec{F}bc \cdot \vec{n}S = gS_{\mathrm{PP'Q'Q}} \left[\rho'_{ijkmax}\left(\eta_{ij} - x_{3_{\mathrm{GH}}}\right) + \rho'_{ijkmax-1}\left(x_{3_{\mathrm{GH}}} - \hat{x}_{3_{\mathrm{PP'Q'Q}}}\right) \right]$$

Ecuación 35

$$\sum_{m=1}^{N_{faces}} \vec{F}bc_m \cdot \vec{n}S_m = gS_{\mathrm{PP'Q'Q}} \left\{ \left[\sum_{n=k+1}^{kmax} \left(\rho'_{ijn}\Delta x_{3_{ijn}}\right) + \rho'_{ijk}\left(x_{3_{\mathrm{GH}}} - \hat{x}_{3_{\mathrm{PP'Q'Q}}}\right) \right] - \right.$$

$$\left. - \left[\sum_{n=k+1}^{kmax} \left(\rho'_{ij-1n}\Delta x_{3_{ij-1n}}\right) + \rho'_{ij-1k}\left(x_{3_{\mathrm{CD}}} - \hat{x}_{3_{\mathrm{PP'Q'Q}}}\right) \right] \right\}$$

Ecuación 36

2.4. Discretizacion del sistema de ecuaciones

A partir de las diferentes discretizaciones presentadas anteriormente, en este punto se desarrolla la discretización del sistema de ecuaciones que realiza el MOHID utilizando los volúmenes finitos. Para esto se utiliza una malla descentrada tipo C y una coordenada vertical genérica con la posibilidad de dividirse en subdominios.

2.4.1. Cálculo de la geometría

Al aplicar el método de los volúmenes finitos es necesario conocer el volumen de las celdas y las áreas de las caras de las celdas en cada instante de tiempo simulado. En la celda adoptada por el modelo los puntos que definen las caras verticales son coplanares, mientras que los puntos que definen las caras superior e inferior no se encuentran en un mismo plano pues son definidos por la coordenada genérica. Sin embargo, debido a las hipótesis consideradas anteriormente mencionadas, solamente es necesario calcular la componente vertical de los flujos por esas caras. Estos flujos dependen solo de la proyección de las caras en el plano horizontal, por lo que no es necesario calcular el área. Por todo esto se calculan en cada iteración las áreas de las seis caras de cada celda a partir de la ecuación que expresa el área de un trapezoide plano.

El volumen de una celda hexaédrica genérica puede determinarse por su descomposición en formas geométricas regulares como pirámides o prismas. Debido a las particularidades de la celda del modelo se optó por una descomposición en cuatro prismas triangulares definidos por las caras verticales y compartiendo una arista común PQ que se obtiene por la media de los vértices de las caras superior e inferior. En la Figura 7 se representa esa descomposición y la nomenclatura utilizada. Esta celda se utiliza para el cálculo de las propiedades densidad ρ, salinidad S, temperatura T, y para la definición de la viscosidad horizontal. La elevación y las magnitudes turbulentas se calculan en el centro de la celda. Las celdas para el cálculo de las velocidades u, v y w se encuentran descentradas de forma que las áreas Au, Av y el punto Q pasan por sus centros, respectivamente. Los índices *i*, *j*, y *k* de ubicación espacial de las celdas coinciden con las direcciones *y*, *x* y *z*, respectivamente. Esto significa que las velocidades horizontales se calculan en los centros de las caras oeste (velocidad *u*) y sur (velocidad *v*). En la dirección vertical también se usa una grilla desfasada con la velocidad vertical *w* y las magnitudes turbulentas verticales ubicadas en las caras del techo y del fondo de la celda, y las velocidades horizontales y la elevación en el centro del elemento en la dirección vertical.

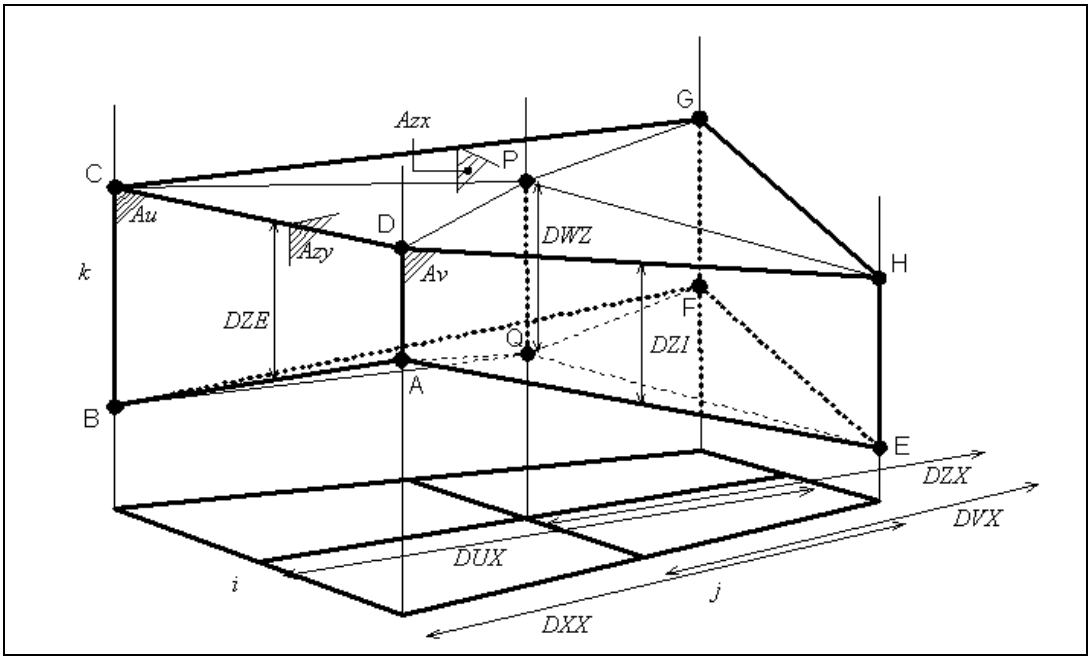

Figura 7: Geometría y nomenclatura de la celda usada por el modelo.

Los pasos espaciales horizontales son definidos de forma que para celda *DUX* puede ser diferente de *DXX*, y para la dirección *y*, *DVY* puede ser diferente de *DYY*. Las celdas no quedan así limitadas a ser rectangulares en su proyección horizontal, pudiendo ser cuadrangulares con cualquier forma. Del punto de vista geométrico las variables son suficientes para implementar una coordenada curvilínea genérica en el plano horizontal. Sin embargo, las ecuaciones de discretización consideran que los flujos por las caras verticales son perpendiculares a estas. La libertad en la geometría de la malla horizontal está por eso limitada a los casos de coordenada cartesiana y coordenadas geográficas, siendo en este último caso calculadas las componentes de la velocidad en las direcciones de esas coordenadas.

2.4.2. Discretización temporal

Para el caso de las ecuaciones tridimensionales los términos que más limitan la estabilidad son los términos de presión barotrópica y el del transporte vertical. El término de presión barotrópica es limitativo debido a la elevada celeridad de las ondas de superficie. El transporte vertical es limitativo debido al pequeño paso espacial en la dirección vertical. Debido a esto, en las ecuaciones de cantidad de movimiento el término barotrópico y el transporte vertical se calculan implícitos. El sistema de ecuaciones resultante es tridiagonal y puede resolverse de forma eficiente y rápida.

La discretización temporal de las ecuaciones discretizadas se realiza por el algoritmo semi-implícito ADI "Alternative Direction Implicit" (Neves, 1985). En esta formulación la ecuación para una de las componentes horizontales de cantidad de movimiento se calcula con el término barotrópico implícito, siendo la otra dirección calculada explícitamente. En la iteración siguiente las direcciones calculadas implícitamente y explícitamente se invierten. Esta metodología es originaria de los modelos bidimensionales integrados en profundidad. Debido a la aproximación hidrostática, el término barotrópico en el caso tridimensional es

idéntico al del caso bidimensional. Por ese motivo la discretización temporal más indicada para resolver las limitaciones de estabilidad asociadas a ese término es idéntica al caso bidimensional. Los términos de transporte vertical son también calculados de forma implícita. El sistema de ecuaciones resultante es de tipo tridiagional, y se resuelve de forma eficiente por el algoritmo de Thomas.

En el código del modelo están incluidas dos discretizaciones diferentes, el esquema S21 (Abbott et al., 1973) de cuatro ecuaciones con dos niveles temporales por iteración (Ecuación 37), y el esquema de Leendertse (Leendertse, 1967) de 6 ecuaciones, tres para cada medio paso de tiempo, donde las velocidades se actualizan cada medio paso (Ecuación 38).

$$\eta^{t+1/2}(u^{t+1},u^{t},v^{t+1/2},v^{t-1/2}) \rightarrow u^{t+1} \rightarrow w^{*t+1/2} \xrightarrow{geometria} w^{t+1/2} \rightarrow S^{t+1/2},T^{t+1/2}$$
$$\rightarrow \eta^{t+1}(u^{t},u^{t+1},v^{t+3/2},v^{t+1/2}) \rightarrow v^{t+3/2} \rightarrow w^{*t+1} \xrightarrow{geometria} w^{t+1} \rightarrow S^{t+1},T^{t+1}$$

Ecuación 37

$$v^{t+1} \rightarrow \eta^{t+1/2}(u^{t+1/2},v^{t}) \rightarrow u^{t+1} \rightarrow w^{*t+1/2} \xrightarrow{geometria} w^{t+1/2} \rightarrow S^{t+1/2},T^{t+1/2}$$
$$\rightarrow u^{t+1} \rightarrow \eta^{t+1}(u^{t+1/2},v^{t+1}) \rightarrow v^{t+1} \rightarrow w^{*t+1} \xrightarrow{geometria} w^{t+1} \rightarrow S^{t+1},T^{t+1}$$

Ecuación 38

En el sistema S21 cada iteración se divide en dos pasos de medio tiempo, en la primera iteración se calcula la elevación de la superficie libre y una velocidad horizontal (u) de forma implícita a partir de la ecuación de cantidad de movimiento correspondiente, mientras que para la otra componente de la velocidad (v) se utiliza el valor anterior. En el segundo medio paso de tiempo se calcula nuevamente la elevación de la superficie libre y la otra componente horizontal de la velocidad de forma implícita. En todos los medios pasos de tiempo se calcula una velocidad vertical w^* a partir de la ecuación de continuidad, luego se actualiza la geometría y se corrige la velocidad vertical. La salinidad y la temperatura se calculan cada medio paso.

En el esquema de Leendertse la componente de la velocidad en la dirección explícita se calcula también cada medio paso de tiempo antes del cálculo de la elevación de la superficie libre. En este caso como se calculan las dos componentes dentro de cada medio paso de tiempo, se actualizan con mayor frecuencia las condiciones de borde y por eso es más apropiado para zonas poco profundas. Ambos esquemas son de segundo orden de aproximación ya que realizan los cálculos centrados en cada medio paso de tiempo.

2.4.3. Ecuación de cantidad de movimiento

La ecuación de transporte de cantidad de movimiento que se expresó en su forma diferencial por la Ecuación 1 se discretiza por el método de los volúmenes finitos usando la Ecuación 13 con $\alpha = \rho U_1$ y $Q = \rho f U_2$ [2] para el caso de la componente según x. La discretización temporal se realiza por diferencia progresiva, resultando para la discretización S21:

$$\frac{\rho_0 Vu_{ijk}^n \left(U_{1_{ijk}}^{n+1} - U_{1_{ijk}}^n\right)}{\Delta t} + \sum_{m=1}^{N_{faces}} \vec{F}_m \cdot \vec{n} S_m = \rho_0 fu_{ijk} Vu_{ijk}^n \overline{U}_{2_{ijk}}^n \qquad \textbf{Ecuación 39}$$

Donde Vu_{ijk}^n es el volumen de la celda de cálculo de U_{1ijk} y fu_{ijk} es el valor del parámetro de Coriolis para esa celda. La velocidad $\overline{U}_{2_{ijk}}^n$ es el valor medio de la componente según y del campo de velocidades en la celda U_{1ijk} y se calcula por ponderación entre los 4 valores de U_2 en dicha celda:

$$\overline{U}_{2_{ijk}}^n = \frac{\dfrac{DXX_{i+1j} U_{2_{i+1\,j-1k}}^n + DXX_{i+1\,j-1} U_{2_{i+1\,jk}}^n}{DXX_{i+1j} + DXX_{i+1\,j-1}} + \dfrac{DXX_{ij} U_{2_{ij-1k}}^n + DXX_{ij-1} U_{2_{ijk}}^n}{DXX_{ij} + DXX_{ij-1}}}{2} \qquad \textbf{Ecuación 40}$$

Como se mencionó anteriormente, el término de Coriolis se discretiza de forma explícita, ya que la restricción que impone no es crítica en flujos costeros ($f\,\Delta t/2 \le 1$), incluso para latitudes elevadas. Los términos Fm representan los flujos convectivos de cantidad de movimiento y las fuerzas de superficie aplicadas en la celda. En las formulaciones que se desarrollan a continuación se considera el esquema S21 de 4 ecuaciones.

2.4.3.1. Flujo convectivo

Los flujos convectivos horizontales se discretizan de forma explícita ya que la restricción del número de Courant impuesta por los mismos no es crítica y la limitación siempre está dada por los términos de presión barotrópica y no por los términos convectivos. Sin embargo, los flujos convectivos verticales pueden ser restrictivos cuando se modelan pequeñas profundidades con coordenada sigma, de forma que hayan reducciones importantes del espesor de las capas. En estos casos el modelo desprecia el término convectivo vertical en esas regiones, ya que el escurrimiento en esas regiones es esencialmente en la dirección horizontal y por eso la convección vertical tiene muy poca importancia.

Para que el transporte de cantidad de movimiento se realice de forma conservativa es importante que los caudales utilizados tengan divergencia nula. Se utilizan por esto en todos los transportes convectivos lo flujos que se utilizaron en el último cálculo de la elevación y de la velocidad vertical. Por esto para calcular U_1^{n+1} se utilizan los caudales del cálculo de η^n que son:

[2] En adelante U_1, U_2 y U_3 representan las tres componentes de la velocidad *u*, *v* y *w*, respectivamente.

$$Fxz^{n-1/2} = \frac{U_1^n Au^{n-1} + U_1^{n-1} Au^{n-2}}{2}$$ **Ecuación 41**

$$Fyz^n = \frac{U_2^{n+1/2} Av^{n-1/2} + U_2^{n-1/2} Av^{n-3/2}}{2}$$ **Ecuación 42**

Para la celda de cálculo de U_1 se utilizarán los siguientes flujos a partir de los anteriores:

$$Fxu_{ijk}^{n-1/2} = \frac{Fxz_{ijk}^{n-1/2} + Fxz_{ij-1k}^{n-1/2}}{2}$$ **Ecuación 43**

$$Fyu_{ijk}^{n-1/2} = \frac{DXX_{ij-1k} Fyz_{ijk}^n + DXX_{ijk} Fyz_{ij-1k}^n}{DXX_{ij-1k} + DXX_{ijk}}$$ **Ecuación 44**

Los flujos convectivos deben calcularse en todas las caras de la celda, como lo indica la Ecuación 45, donde los índices espaciales de los flujos convectivos son relativos a la celda U_{1ijk}.

$$\sum_{m=1}^{N_{faces}} \vec{Fc}_m \cdot \vec{n}S_m = \left(\vec{Fc} \cdot \vec{n}S\right)_{ij-1/2k}^n + \left(\vec{Fc} \cdot \vec{n}S\right)_{ij+1/2k}^n + \left(\vec{Fc} \cdot \vec{n}S\right)_{i-1/2\,jk}^n + \left(\vec{Fc} \cdot \vec{n}S\right)_{i+1/2\,jk}^n + \\ + \left(\vec{Fc} \cdot \vec{n}S\right)_{ijk-1/2}^n + \left(\vec{Fc} \cdot \vec{n}S\right)_{ijk+1/2}^n$$

Ecuación 45

Para los flujos convectivos horizontales se utiliza la ponderación de la propiedad α según se presentó en la ecuación la Ecuación 16. Para la dirección *x* se obtiene la Ecuación 46 y para la dirección *y* la Ecuación 47; donde γ representa un factor de ponderación entre el esquema "*upwind*" y diferencia central.

$$\left(\vec{Fc} \cdot \vec{n}S\right)_{ij-½k}^n = \rho_0 \left[\gamma \left(\frac{Fxu_{ijk}^{n-1/2} + \left|Fxu_{ijk}^{n-1/2}\right|}{2} U_{1_{ij-1k}}^n + \frac{Fxu_{ijk}^{n-1/2} - \left|Fxu_{ijk}^{n-1/2}\right|}{2} U_{1_{ijk}}^n \right) + \right. \\ \left. + (1-\gamma) \frac{U_{1_{ijk}}^n + U_{1_{ij-1k}}^n}{2} \right] Fxu_{ijk}^{n-1/2}$$

Ecuación 46

$$\left(\vec{Fc} \cdot \vec{n}S\right)_{i-½\,jk}^n = \rho_0 \left[\gamma \left(\frac{Fyu_{ijk}^n + \left|Fyu_{ijk}^n\right|}{2} U_{1_{i-1\,jk}}^n + \frac{Fyu_{ijk}^n - \left|Fyu_{ijk}^n\right|}{2} U_{1_{ijk}}^n \right) + \right. \\ \left. + (1-\gamma) \frac{DYY_{ij} U_{1_{i-1\,jk}}^n + DYY_{i-1\,j} U_{1_{ijk}}^n}{DYY_{ij} + DYY_{i-1\,j}} \right] Fyu_{ijk}^n$$

Ecuación 47

Para la dirección vertical los flujos convectivos se discretizan utilizando la Ecuación 21, presentada anteriormente, pero considerando los flujos *Fzu* que atraviesan las caras superior e inferior.

$$\left(\vec{F}c\cdot\vec{n}S\right)^{n}_{ijk-½}=\rho_0\left[\gamma\left(\frac{Fzu^{n}_{ijk}+\left|Fzu^{n}_{ijk}\right|}{2}U^{n}_{1_{ijk-1}}+\frac{Fzu^{n}_{ijk}-\left|Fzu^{n}_{ijk}\right|}{2}U^{n}_{1_{ijk}}\right)+\right.$$
$$\left.+(1-\gamma)\frac{DZE_{ijk}U^{n}_{1_{ijk-1}}+DZE_{ijk-1}U^{n}_{1_{ijk}}}{DZE_{ijk}+DZE_{ijk-1}}\right]Fxu^{n}_{ijk}$$

Ecuación 48

2.4.3.2. Presión barotrópica

El flujo de cantidad de movimiento *Fbt* asociado a la acción de la fuerza de presión barotrópica se presentó anteriormente en la Ecuación 28. Ya se mencionó que este término debe ser discretizado de forma implícita pues la restricción en el paso temporal impuesta por el número de Courant basado en la velocidad de propagación de las ondas de superficie es muy condicionante ($\left(u_1+\sqrt{2gH}\right)\Delta t/\Delta x\le 1$). Para la celda U_{1ijk} y utilizando que el área $S_{PP'Q'Q}$ coincide con el área Au el término resulta:

$$-\sum_{m=1}^{N_{faces}}\vec{F}bt_m\cdot\vec{n}S_m=\left[\left(P^{n+1}_{atm_{ij-1}}-P^{n+1}_{atm_{ij}}\right)+\rho_0 g\left(\eta^{n+½}_{ij-1}-\eta^{n+½}_{ij}\right)\right]Au^{n}_{ijk}$$

Ecuación 49

Cuando se calculan las velocidades se utilizan los valores de la elevación calculados antes como lo indican los esquemas de resolución S21 y de Leendertse. Esto término puede entonces implementarse como explícito en el cálculo de las velocidades aunque se trate de un término implícito.

2.4.3.3. Presión baroclínica

La fuerza de presión baroclínica produce un flujo de cantidad de movimiento a través de las caras de la celda U_{1ijk} dada por la Ecuación 36 presentada anteriormente:

$$-\sum_{m=1}^{N_{faces}}\vec{F}bc_m\cdot\vec{n}S_m=g\left\{\left[\sum_{l=k+1}^{kmax}\left(\rho'^{n}_{ij-1l}DWZ_{ij-1l}\right)+\rho'^{n}_{ij-1k}\Delta\hat{x}^{n}_{3_{ij-1k}}\right]-\right.$$
$$\left.-\left[\sum_{l=k+1}^{kmax}\left(\rho'^{n}_{ijl}DWZ_{ijl}\right)+\rho'^{n}_{ijk}\Delta\hat{x}^{n}_{3_{ijk}}\right]\right\}Au^{n}_{ijk}$$

Ecuación 50

Siendo $\Delta\hat{x}_{3_{ijk}}$ la distancia vertical medida desde la arista superior de la celda al punto de cálculo de la velocidad.

2.4.3.4. Flujos difusivos horizontales

Los flujos difusivos horizontales se calculan de forma explícita en todas las caras verticales de la celda, considerando que la dirección normal a cada cara coincide con la dirección del flujo:

$$-\sum_{m=1}^{N_{faces_h}} \vec{F}dh_m \cdot \vec{n}S_m = \left(Fdh^n_{ij-½k} Azx^n_{ij-1k} - Fdh^n_{ij+½k} Azx^n_{ijk}\right) + \left(Fdh^n_{i-½jk} \frac{Av^n_{ijk} + Av^n_{ij-1k}}{2} - Fdh^n_{i+½jk} \frac{Av^n_{i+1jk} + Av^n_{i+1j-1k}}{2}\right)$$

Ecuación 51

Cada flujo difusivo horizontal *Fdh* se calcula con la discretización central definida anteriormente por la Ecuación 20. Para la dirección *x* se obtiene la Ecuación 52 y para la dirección *y* la Ecuación 53.

$$Fdh^n_{ij-½k} = -\rho_0 A^n_{h_{ij\text{-}1k}} \frac{U^n_{1_{ijk}} - U^n_{1_{ij-1k}}}{DUX_{ij-1}}$$

Ecuación 52

$$Fdh^n_{i-½jk} = -\rho_0 A^n_{h_{i-1/2\,j\text{-}1/2k}} \frac{U^n_{1_{ijk}} - U^n_{1_{i-1\,jk}}}{\left(DYY_{ij} + DYY_{i-1\,j}\right)/2}$$

Ecuación 53

Con:

$$A^n_{h_{i-1/2\,j\text{-}1/2k}} = \frac{DYY_{i-1\,j}\dfrac{DUX_{ij}A^n_{h_{ij\text{-}1k}} + DUX_{ij-1}A^n_{h_{ijk}}}{DUX_{ij} + DUX_{ij-1}} + DYY_{ij}\dfrac{DUX_{i-1\,j}A^n_{h_{i-1\,j\text{-}1k}} + DUX_{i-1\,j-1}A^n_{h_{i-1\,jk}}}{DUX_{i-1\,j} + DUX_{i-1\,j-1}}}{DYY_{i-1\,j} + DYY_{ij}}$$

la viscosidad turbulenta en la cara *i-1/2jk* de la celda U_{1ijk}.

2.4.3.5. Flujo difusivo vertical

Este término debe discretizarse de forma implícita ya que la restricción que impone en el paso temporal ($A_i\,\Delta t/\Delta x_i^2 \le ½ \qquad i = 1,2,3$) es del mismo orden que la restricción impuesta por el término barotrópico. Los flujos difusivos verticales se calculan en las caras superior e inferior de la celda considerando las aproximaciones vistas anteriormente, utilizando la Ecuación 22 para los términos *Fdv*:

$$-\sum_{m=1}^{N_{faces}} \vec{F}dv_m \cdot \vec{n}S_m = \left(Fdv^{n+1}_{ijk-½} - Fdv^{n+1}_{ijk+½}\right)Ah^n_{ij-½}$$

Ecuación 54

Con: $Ah^n_{ij-½} = \left(Ah^n_{ij-1} + Ah^n_{ij}\right)/2$

$$Fdv^{n+1}_{ijk-½} = -\rho_0 A^n_{v_{ij-1/2k-1}} \frac{U^{n+1}_{1_{ijk}} - U^{n+1}_{1_{ijk-1}}}{DUZ^n_{ijk-1}}$$

Ecuación 55

y $A^n_{v_{ij-1/2k-1}} = \left(DUX_{ij}A^n_{v_{ij-1k-1}} + DUX_{ij-1}A^n_{v_{ijk-1}}\right)/\left(DUX_{ij} + DUX_{ij-1}\right)$

es la viscosidad vertical en la cara superior de la celda y $DUZ^n_{ijk} = \left(DZE^n_{ijk+1} + DZE^n_{ijk}\right)/2$.

2.4.3.6. Ecuación resultante

La forma discretizada de la ecuación de transporte de cantidad de movimiento se obtiene sustituyendo las expresiones determinadas anteriormente para los diferentes flujos en la Ecuación 39. Para la componente según *x* se obtiene la Ecuación 56, agrupando los términos explícitos en un término común X_{ijk}. Si luego se despejan las velocidades que no se conocen, correspondientes al instante *n+1*, se obtiene la Ecuación 57.

$$\frac{U_{1_{ijk}}^{n+1}-U_{1_{ijk}}^{n}}{\Delta t}=\frac{1}{Vu_{ijk}^{n}}\left[\left(\frac{p_{atm_{ij-1}}^{t+1}-p_{atm_{ij}}^{t+1}}{\rho_0}+g\left(\eta_{ij-1}^{n+1/2}-\eta_{ij}^{n+1/2}\right)\right)\cdot Au_{ijk}^{n}+\right.$$
$$\left.+\left(A_{v_{ij-1/2k}}^{n}\frac{U_{1_{ijk+1}}^{n+1}-U_{1_{ijk}}^{n+1}}{DUZ_{ijk}^{n}}-A_{v_{ij-1/2k-1}}^{n}\frac{U_{1_{ijk}}^{n+1}-U_{1_{ijk-1}}^{n+1}}{DUZ_{ijk-1}^{n}}\right)\cdot Ah_{ij-1/2}+X_{ijk}^{n}\right]$$

Ecuación 56

$$\left(-\frac{\Delta t\,Ah_{ij-1/2}}{Vu_{ijk}^{n}}\frac{A_{v_{ij-1/2k-1}}^{n}}{DUZ_{ijk-1}^{n}}\right)U_{1_{ijk-1}}^{n+1}+\left[1+\frac{\Delta t\,Ah_{ij-1/2}}{Vu_{ijk}^{n}}\left(\frac{A_{v_{ij-1/2k-1}}^{n}}{DUZ_{ijk-1}^{n}}+\frac{A_{v_{ij-1/2k}}^{n}}{DUZ_{ijk}^{n}}\right)\right]U_{1_{ijk}}^{n+1}$$
$$+\left(-\frac{\Delta t\,Ah_{ij-1/2}}{Vu_{ijk}^{n}}\frac{A_{v_{ij-1/2k}}^{n}}{DUZ_{ijk}^{n}}\right)U_{1_{ijk+1}}^{n+1}=$$
$$U_{1_{ijk}}^{n}+\left[\frac{\Delta t}{Vu_{ijk}^{n}}\left(\left[\frac{p_{atm_{ij-1}}^{t+1}-p_{atm_{ij}}^{t+1}}{\rho_0}+g\left(\eta_{ij-1}^{n+1/2}-\eta_{ij}^{n+1/2}\right)\right]Au_{ijk}^{n}+X_{ijk}^{n}\right)\right]$$

Ecuación 57

La Ecuación 57 es de la forma $Du_{1_{ijk}}\cdot U_{1_{ijk-1}}^{n+1}+Eu_{1_{ijk}}\cdot U_{1_{ijk}}^{n+1}+Fu_{1_{ijk}}\cdot U_{1_{ijk+1}}^{n+1}=TIu_{1_{ijk}}$, con Du, Eu, Fu y TIu dados por dicha ecuación. La aplicación de la misma a todos los puntos del dominio de cálculo genera un sistema tridiagonal de ecuaciones que se resuelve eficientemente por el método de Thomas.

Para la componente según *y* de la ecuación de cantidad de movimiento el cálculo es semejante y se obtiene para el método S21 el sistema siguiente:

$$Du_{2_{ijk}}\cdot U_{2_{ijk-1}}^{n+3/2}+Eu_{2_{ijk}}\cdot U_{2_{ijk}}^{n+3/2}+Fu_{2_{ijk}}\cdot U_{2_{ijk+1}}^{n+3/2}=TIu_{2_{ijk}}$$

Ecuación 58

Siendo:

$$Du_{2_{ijk}}=-\frac{\Delta t\,Ah_{i-1/2\,j}}{Vv_{ijk}^{n+1/2}}\frac{A_{v_{i-1/2\,jk-1}}^{n+1/2}}{DVZ_{ijk-1}^{n+1/2}}$$

$$Eu_{2_{ijk}}=\left[1+\frac{\Delta t\,Ah_{i-1/2\,j}}{Vv_{ijk}^{n+1/2}}\left(\frac{A_{v_{i-1/2\,jk-1}}^{n+1/2}}{DVZ_{ijk-1}^{n+1/2}}+\frac{A_{v_{i-1/2\,jk}}^{n+1/2}}{DVZ_{ijk}^{n+1/2}}\right)\right]$$

$$Fu_{2_{ijk}}=-\frac{\Delta t\,Ah_{i-1/2\,j}}{Vv_{ijk}^{n+1/2}}\frac{A_{v_{i-1/2\,jk}}^{n+1/2}}{DVZ_{ijk}^{n+1/2}}$$

$$TIu_{2_{ijk}}=U_{2_{ijk}}^{n+1/2}+\left[\frac{\Delta t}{Vv_{ijk}^{n+1/2}}\left(\left[\frac{p_{atm_{i-1\,j}}^{n+3/2}-p_{atm_{ij}}^{n+3/2}}{\rho_0}+g\left(\eta_{i-1\,j}^{n+1}-\eta_{ij}^{n+1}\right)\right]Av_{ijk}^{n+1/2}+Y_{ijk}^{n+1/2}\right)\right]$$

2.4.3.7. Tensión de corte en la superficie y el fondo

En la celda de superficie el flujo difusivo por la cara superior debe entenderse como una tensión de corte debida a la acción del viento que se impone directamente en el sistema de ecuaciones. Esto equivale a cambiar en la Ecuación 56 el siguiente término para la capa *k=kmax*:

$$A^n_{v_{ij-1/2k}} \frac{U^{n+1}_{1_{ijk+1}} - U^{n+1}_{1_{ijk}}}{DUZ^n_{ijk}} = \tau u^{vento}_{ij}$$

Ecuación 59

Y los coeficientes se transforman en:

$$Du_{1_{ijkmax}} = -\frac{\Delta t\, Ah_{ij-1/2}}{Vu^n_{ijkmax}} \frac{A^n_{v_{ij-1/2kmax-1}}}{DUZ^n_{ijkmax-1}} \qquad Eu_{1_{ijkmax}} = \left[1 + \frac{\Delta t\, Ah_{ij-1/2}}{Vu^n_{ijkmax}} \frac{A^n_{v_{ij-1/2kmax-1}}}{DUZ^n_{ijkmax-1}}\right] \qquad Fu_{2_{ijk}} = 0$$

$$TIu_{1_{ijk}} = U^n_{1_{ijk}} + \left[\frac{\Delta t}{Vu^n_{ijk}}\left(\left[\frac{p^{n+1}_{atm_{i-1\,j}} - p^{n+1}_{atm_{ij}}}{\rho_0} + g\left(\eta^{n+1/2}_{i-1\,j} - \eta^{n+1/2}_{ij}\right)\right] Au^n_{ijk} + \frac{\tau u^{vento}_{ij}}{\rho_0} Ah_{ij-1/2} + X^n_{ijk}\right)\right]$$

El flujo difusivo por la cara inferior de la celda de fondo debe también entenderse como una tensión de corte de fondo. Esta tensión se calcula con una ley cuadrática en función de la velocidad de la celda de fondo y se incluye en el sistema de ecuaciones. Rescribiendo la Ecuación 56 para la capa *k=kfondo*:

$$U^{n+1}_{1_{ijkfundo}} = U^n_{1_{ijkfundo}} + \frac{\Delta t\, Ah_{ij-1/2}}{\rho_0 Vu^n_{ijkfundo}}\left(\tau u_{ijkfundo+1} - \tau u_{ijkfundo}\right) + X'^n_{ijkfundo}$$

Ecuación 60

Donde *X'* representa los restantes términos de la ecuación. Implementando la tensión de corte de fondo de forma explícita no se obtiene una formulación estable por lo que dicha tensión se calcula implícitamente:

$$\tau u_{ijkfundo} = \rho_0 C_D \left|\vec{u}_{ijkfundo}\right| U^{n+1}_{1_{ijkfundo}}$$

Ecuación 61

En este caso la Ecuación 60 se transforma en la Ecuación 62, y reordenando los términos se obtiene finalmente la Ecuación 63.

$$U^{n+1}_{1_{ijkfundo}}\left(1+R\right) = U^n_{1_{ijkfundo}} + \frac{\Delta t\, Ah_{ij-1/2}}{Vu^n_{ijkfundo}}\left(\tau u_{ijkfundo+1}\right) + X'^n_{ijkfundo}$$

Ecuación 62

$$U^{n+1}_{1_{ijkfundo}} = Fx_{ij}\left[U^n_{1_{ijkfundo}} + \frac{\Delta t\, Ah_{ij-1/2}}{Vu^n_{ijkfundo}}\left(\tau u_{ijkfundo+1}\right) + X'^n_{ijkfundo}\right]$$

Ecuación 63

donde:

$$Fx_{ij} = \frac{1}{1 + \dfrac{\Delta t\, Ah_{ij-1/2} C_D \left|\vec{u}_{ijkfundo}\right|}{Vu^n_{ijkfundo}}}$$

2.4.4. Elevación de la superficie libre

La elevación de la superficie libre se calcula por integración en toda la columna de agua de la ecuación de continuidad (Ecuación 5). Al discretizar esta ecuación utilizando el método de los volúmenes finitos la integración se aproxima por una sumatoria de los flujos volumétricos de todas las celdas de la columna de agua. Esta discretización utilizando el método S21 se expresa con la Ecuación 63 para el primer medio paso de tiempo y con la Ecuación 64 para el segundo medio paso de tiempo, siendo $Ah_{ij} = DUX_{ij} \cdot DVY_{ij}$ el área proyectada en el plano horizontal. Los dos medios pasos de tiempo generan que el cálculo sea globalmente centrado en *n+1/2*.

$$\frac{\eta_{ij}^{n+1/2}-\eta_{ij}^{n}}{\Delta t/2}=\frac{1}{Ah_{ij}}\left[\left\{\frac{\sum_{k=kfundo}^{kmax}\left(U_{1_{ijk}}^{n+1}\cdot Au_{ijk}^{n}\right)+\sum_{k=kfundo}^{kmax}\left(U_{1_{ijk}}^{n}\cdot Au_{ijk}^{n-1}\right)}{2}-\frac{\sum_{k=kfundo}^{kmax}\left(U_{1_{ij+1k}}^{n+1}\cdot Au_{ij+1k}^{n}\right)+\sum_{k=kfundo}^{kmax}\left(U_{1_{ij+1k}}^{n}\cdot Au_{ij+1k}^{n-1}\right)}{2}\right\}\right.$$
$$\left.+\left\{\frac{\sum_{k=kfundo}^{kmax}\left(U_{2_{ijk}}^{n+1/2}\cdot Av_{ijk}^{n-1/2}\right)+\sum_{k=kfundo}^{kmax}\left(U_{2_{ijk}}^{n-1/2}\cdot Av_{ijk}^{n-3/2}\right)}{2}-\frac{\sum_{k=kfundo}^{kmax}\left(U_{2_{i+1jk}}^{n+1/2}\cdot Av_{i+1jk}^{n-1/2}\right)+\sum_{k=kfundo}^{kmax}\left(U_{2_{i+1jk}}^{n-1/2}\cdot Av_{i+1jk}^{n-3/2}\right)}{2}\right\}\right]$$

Ecuación 64

$$\frac{\eta_{ij}^{n+1}-\eta_{ij}^{n+1/2}}{\Delta t/2}=\frac{1}{Ah_{ij}}\left[\left\{\frac{\sum_{k=kfundo}^{kmax}\left(U_{1_{ijk}}^{n+1}\cdot Au_{ijk}^{n}\right)+\sum_{k=kfundo}^{kmax}\left(U_{1_{ijk}}^{n}\cdot Au_{ijk}^{n-1}\right)}{2}-\frac{\sum_{k=kfundo}^{kmax}\left(U_{1_{ij+1k}}^{n+1}\cdot Au_{ij+1k}^{n}\right)+\sum_{k=kfundo}^{kmax}\left(U_{1_{ij+1k}}^{n}\cdot Au_{ij+1k}^{n-1}\right)}{2}\right\}\right.$$
$$\left.+\left\{\frac{\sum_{k=kfundo}^{kmax}\left(U_{2_{ijk}}^{n+3/2}\cdot Av_{ijk}^{n+1/2}\right)+\sum_{k=kfundo}^{kmax}\left(U_{2_{ijk}}^{n+1/2}\cdot Av_{ijk}^{n-1/2}\right)}{2}-\frac{\sum_{k=kfundo}^{kmax}\left(U_{2_{i+1jk}}^{n+3/2}\cdot Av_{i+1jk}^{n+1/2}\right)+\sum_{k=kfundo}^{kmax}\left(U_{2_{i+1jk}}^{n+1/2}\cdot Av_{i+1jk}^{n-1/2}\right)}{2}\right\}\right]$$

Ecuación 65

Para la discretización de Leendertse la elevación de la superficie libre en el primer medio paso de tiempo está dada por la Ecuación 66 y en el segundo medio paso por la Ecuación 67; siendo también globalmente centrada en *n+1/2*.

$$\frac{\eta_{ij}^{n+1/2}-\eta_{ij}^{n}}{\Delta t/2}=\frac{1}{Ah_{ij}}\left[\left\{\sum_{k=kfundo}^{kmax}\left(U_{1_{ijk}}^{n+1/2}\cdot Au_{ijk}^{n}\right)-\sum_{k=kfundo}^{kmax}\left(U_{1_{ij+1k}}^{n+1/2}\cdot Au_{ij+1k}^{n}\right)\right\}+\right.$$
$$\left.+\left\{\sum_{k=kfundo}^{kmax}\left(U_{2_{ijk}}^{n}\cdot Av_{ijk}^{n-1/2}\right)-\sum_{k=kfundo}^{kmax}\left(U_{2_{i+1jk}}^{n}\cdot Av_{i+1jk}^{n-1/2}\right)\right\}\right]$$

Ecuación 66

$$\frac{\eta_{ij}^{n+1}-\eta_{ij}^{n+1/2}}{\Delta t/2}=\frac{1}{Ah_{ij}}\left[\left\{\sum_{k=kfundo}^{kmax}\left(U_{1_{ijk}}^{n+1/2}\cdot Au_{ijk}^{n}\right)-\sum_{k=kfundo}^{kmax}\left(U_{1_{ij+1k}}^{n+1/2}\cdot Au_{ij+1k}^{n}\right)\right\}+\right.$$
$$\left.+\left\{\sum_{k=kfundo}^{kmax}\left(U_{2_{ijk}}^{n+1}\cdot Av_{ijk}^{n+1/2}\right)-\sum_{k=kfundo}^{kmax}\left(U_{2_{i+1jk}}^{n+1}\cdot Av_{i+1jk}^{n+1/2}\right)\right\}\right]$$

Ecuación 67

Los flujos $U_1 \cdot Au$ y $U_2 \cdot Av$, que son implícitos en las ecuaciones anteriores, deben calcularse a partir de cada ecuación de cantidad de movimiento respectiva (Ecuación 57 y Ecuación 58, respectivamente). Por ejemplo en la Ecuación 64, los flujos de la forma $U_1^{n+1} \cdot Au^n$ son los únicos flujos implícitos, y se calculan a partir de la Ecuación 57. En ese caso al realizar la sumatoria en toda la columna de agua los términos de difusión vertical se cancelan entre capas adyacentes y la difusión en la capa superior de la celda de superficie y en la capa inferior de la celda de fondo se calculan como tensiones de corte. Se obtiene entonces la Ecuación 68, que al calcular la tensión de fondo de forma implícita se transforma en la Ecuación 69.

$$\sum_{k=kfundo}^{kmax}\left(U_{1_{ijk}}^{n+1} \cdot Au_{ijk}^{n}\right) = \sum_{k=kfundo}^{kmax}\left(U_{1_{ijk}}^{n} \cdot Au_{ijk}^{n}\right) +$$
$$+ \Delta t\left\{\left(\frac{p_{atm_{ij-1}}^{n+1} - p_{atm_{ij}}^{n+1}}{\rho_0}\right) + g\left(\eta_{ij-1}^{n+1/2} - \eta_{ij}^{n+1/2}\right)\right\} \cdot \sum_{k=kfundo}^{kmax} \frac{\left(Au_{ijk}^{n}\right)^2}{Vu_{ijk}^{n}} +$$
$$+ \frac{\Delta t}{\rho_0} \cdot \left\{\tau u_{ij}^{vento} \cdot \frac{Au_{ijkmax}^{n}}{Vu_{ijkmax}^{n}} - \tau u_{ij}^{fundo} \cdot \frac{Au_{ij1}^{n}}{Vu_{ij1}^{n}}\right\} Ah_{ij-1/2} + \frac{\Delta t}{\rho_0} \cdot \sum_{k=kfundo}^{kmax}\left(X_{ijk} \frac{Au_{ijk}^{n}}{Vu_{ijk}^{n}}\right)$$

Ecuación 68

$$\sum_{k=kfundo}^{kmax}\left(U_{1_{ijk}}^{n+1} \cdot Au_{ijk}^{n}\right) = U_{1_{ijkfundo}}^{n+1} \cdot Au_{ijkfundo}^{n} + \sum_{k=kfundo+1}^{kmax}\left(U_{1_{ijk}}^{n+1} \cdot Au_{ijk}^{n}\right) = \sum_{k=kfundo+1}^{kmax}\left(U_{1_{ijk}}^{n} \cdot Au_{ijk}^{n}\right) +$$
$$+ Fx_{ij}\left(U_{1_{ijkfundo}}^{n} + \frac{\Delta t \cdot X_{ijkfundo}}{\rho_0 \cdot Vu_{ijkfundo}^{n}}\right) Au_{ijkfundo}^{n} + \frac{\Delta t}{\rho_0} \sum_{k=kfundo+1}^{kmax}\left(X_{ijk} \frac{Au_{ijk}^{n}}{Vu_{ijk}^{n}}\right) +$$
$$+ \Delta t\left\{\frac{\left(p_{atm_{ij-1}}^{n+1} - p_{atm_{ij}}^{n+1}\right)}{\rho_0} + g\left(\eta_{ij-1}^{n+1/2} - \eta_{ij}^{n+1/2}\right)\right\} \cdot \left[Fx_{ij} \frac{\left(Au_{ijkfundo}^{n}\right)^2}{Vu_{ijkfundo}^{n}} + \sum_{k=kfundo+1}^{kmax} \frac{\left(Au_{ijk}^{n}\right)^2}{Vu_{ijk}^{n}}\right] +$$
$$+ \frac{\Delta t}{\rho_0} \cdot \left\{\tau u_{ij}^{vento} \cdot \frac{Au_{ijkmax}^{n}}{Vu_{ijkmax}^{n}} + \left(Fx_{ij} \frac{Au_{ijkfundo}^{n}}{Vu_{ijkfundo}^{n}} - \frac{Au_{ijkfundo+1}^{n}}{Vu_{ijkfundo+1}^{n}}\right) \cdot \left(\rho_0 A_{v_{ij-1/2kfundo}}^{n} \frac{U_{1_{ijkfundo+1}}^{n} - U_{1_{ijkfundo}}^{n}}{DUZ_{ijkfundo}^{n}}\right)\right\} Ah_{ij-1/2}$$

Ecuación 69

La expresión dada por la Ecuación 69 es la que se sustituye en la Ecuación 63, junto con una expresión equivalente para la cara *ij+1k*, y rescribiendo la ecuación en función de la elevación de la superficie libre se obtiene la ecuación definitiva, de la forma:

$$Dz_{ij} \cdot \eta_{ij-1}^{n+1/2} + Ez_{ij} \cdot \eta_{ij}^{n+1/2} + Fz_{ij} \cdot \eta_{ij+1}^{n+1/2} = TIz_{ij}$$

Ecuación 70

Cuyos coeficientes están dados por:

$$Dz_{ij} = -\frac{\Delta t^2 g}{4Ah_{ij}}\left[Fx_{ij} \frac{\left(Au_{ijkfundo}^{n}\right)^2}{Vu_{ijkfundo}^{n}} + \sum_{k=kfundo+1}^{kmax} \frac{\left(Au_{ijk}^{n}\right)^2}{Vu_{ijk}^{n}}\right]$$

$$Ez_{ij} = 1 + \frac{\Delta t^2 g}{4Ah_{ij}} \left[Fx_{ij} \left\{ \frac{\left(Au^n_{ijkfundo}\right)^2}{Vu^n_{ijkfundo}} + \sum_{k=kfundo+1}^{kmax} \frac{\left(Au^n_{ijk}\right)^2}{Vu^n_{ijk}} \right\} + \right.$$
$$\left. + Fx_{ij+1} \left\{ \frac{\left(Au^n_{ij+1kfundo}\right)^2}{Vu^n_{ij+1kfundo}} + \sum_{k=kfundo+1}^{kmax} \frac{\left(Au^n_{ij+1k}\right)^2}{Vu^n_{ij+1k}} \right\} \right]$$

$$Fz_{ij} = -\frac{\Delta t^2 g}{4Ah_{ij}} \left[Fx_{ij+1} \frac{\left(Au^n_{ij+1kfundo}\right)^2}{Vu^n_{ij+1kfundo}} + \sum_{k=kfundo+1}^{kmax} \frac{\left(Au^n_{ij+1k}\right)^2}{Vu^n_{ij+1k}} \right]$$

$$TIz_{ij} = \eta^n_{ij} + \frac{\Delta t}{4 \cdot Ah_{ij}} \left[\left\langle \left(1 + Fx_{ij}\right) U^n_{1_{ijkfundo}} Au^n_{ijkfundo} + 2 \sum_{k=kfundo+1}^{kmax} \left(U^n_{1_{ijk}} Au^n_{ijk}\right) + \frac{\Delta t}{\rho_0} \left\{ Fx_{ij} X_{ijkfundo} \frac{Au^n_{ijkfundo}}{Vu^n_{ijkfundo}} + \right.\right.\right.$$
$$\left.\left.\left. + \sum_{k=kfundo+1}^{kmax} \left(X_{ijk} \frac{Au^n_{ijk}}{Vu^n_{ijk}} \right) \right\} + \Delta t \frac{p^{t+1/2}_{atm_{ij-1}} - p^{t+1/2}_{atm_{ij}}}{\rho_0} \left[Fx_{ij} \frac{\left(Au^n_{ijkfundo}\right)^2}{Vu^n_{ijkfundo}} + \sum_{k=kfundo+1}^{kmax} \frac{\left(Au^n_{ijk}\right)^2}{Vu^n_{ijk}} \right] + \right.$$
$$\left. \frac{\Delta t}{\rho_0} \cdot \left\{ \tau u^{vento}_{ij} \cdot \frac{Au^n_{ijkmax}}{Vu^n_{ijkmax}} + \left(Fx_{ij} \frac{Au^n_{ijkfundo}}{Vu^n_{ijkfundo}} - \frac{Au^n_{ijkfundo+1}}{Vu^n_{ijkfundo+1}} \right) \cdot \left(\rho_0 A_{v_{ij-\frac{1}{2}kfundo}} \frac{U^n_{1_{ijkfundo+1}} - U^n_{1_{ijkfundo}}}{DUZ^n_{ijkfundo}} \right) \right\} Ah_{ij-\frac{1}{2}} \right\rangle -$$
$$\left[\left\langle \left(1 + Fx_{ij+1}\right) U^n_{1_{ij+1kfundo}} Au^n_{ij+1kfundo} + 2 \sum_{k=kfundo+1}^{kmax} \left(U^n_{1_{ij+1k}} Au^n_{ij+1k}\right) + \frac{\Delta t}{\rho_0} \left\{ Fx_{ij+1} X_{ij+1kfundo} \frac{Au^n_{ij+1kfundo}}{Vu^n_{ij+1kfundo}} + \right.\right.\right.$$
$$\left.\left.\left. + \sum_{k=kfundo+1}^{kmax} \left(X_{ij+1k} \frac{Au^n_{ij+1k}}{Vu^n_{ij+1k}} \right) \right\} + \Delta t \frac{p^{t+1/2}_{atm_{ij}} - p^{t+1/2}_{atm_{ij+1}}}{\rho_0} \left[Fx_{ij+1} \frac{\left(Au^n_{ij+1kfundo}\right)^2}{Vu^n_{ij+1kfundo}} + \sum_{k=kfundo+1}^{kmax} \frac{\left(Au^n_{ij+1k}\right)^2}{Vu^n_{ij+1k}} \right] + \right.$$
$$\left. \frac{\Delta t}{\rho_0} \cdot \left\{ \tau u^{vento}_{ij+1} \cdot \frac{Au^n_{ij+1kmax}}{Vu^n_{ij+1kmax}} + \left(Fx_{ij+1} \frac{Au^n_{ij+1kfundo}}{Vu^n_{ij+1kfundo}} - \frac{Au^n_{ij+1kfundo+1}}{Vu^n_{ij+1kfundo+1}} \right) \cdot \left(\rho_0 A_{v_{ij+\frac{1}{2}kfundo}} \frac{U^n_{1_{ij+1kfundo+1}} - U^n_{1_{ij+1kfundo}}}{DUZ^n_{ij+1kfundo}} \right) \right\} Ah_{ij+\frac{1}{2}} \right\rangle +$$
$$\left. \left\langle \sum_{k=kfundo}^{kmax} \left(U^{n+1/2}_{2_{ijk}} \cdot Av^n_{ijk} + U^{n-1/2}_{2_{ijk}} \cdot Av^{n-1}_{ijk} \right) \right\rangle - \left\langle \sum_{k=kfundo}^{kmax} \left(U^{n+1/2}_{2_{i+1\,jk}} \cdot Av^n_{i+1\,jk} + U^{n-1/2}_{2_{i+1\,jk}} \cdot Av^{n-1}_{i+1\,jk} \right) \right\rangle \right]$$

La aplicación de esta ecuación a todas las celdas del dominio de cálculo genera un sistema tridiagonal de ecuaciones, que se resuelve para calcular la elevación de la superficie libre para los medios pasos de tiempo *n+1/2*. En esta ecuación la dirección implícita es *x* y los términos de la dirección *y* se calculan de forma explícita. Como se utiliza el método de resolución temporal ADI, en el siguiente medio paso de tiempo la dirección implícita será la dirección *y*, y la dirección *x* se calculará de forma explícita.

2.4.5. Velocidad vertical y redefinición de la geometría

La velocidad vertical se calcula utilizando la ecuación de continuidad (Ecuación 4). La discretización de dicha ecuación plantea que el valor para el límite de integración en la coordenada vertical sea la coordenada de la cara superior de la celda. Cuando se utiliza una coordenada vertical genérica, el método de cálculo de la velocidad vertical debe permitir el movimiento vertical de la malla. Esto equivale a que en la ecuación de continuidad integrada el límite de integración no está fijo en el tiempo, sino que depende de la variación del volumen de las celdas que se

encuentran por debajo. De esta forma la velocidad que se calcula es la relativa a la malla, responsable del caudal volumétrico a través de esa cara. Este caudal *Fzz* se calcula a partir de las ecuaciones 17 y 18 presentadas anteriormente.

El cálculo del movimiento de la malla y de la velocidad vertical se realizan en conjunto aplicando una metodología de tres pasos. En primer lugar se realiza una estimación de la velocidad vertical U_3* considerando que la malla permanece fija, luego se calcula el movimiento explícito de la malla y se redefine la geometría y por último se realiza el cálculo del valor final de la velocidad vertical U_3. Se analizan estos pasos para el esquema S21 en relación al cálculo de $U_{3_{ijk}}^{n+1/2}$.

2.4.5.1. Estimación de la velocidad vertical

Al considerar que la malla permanece fija, el caudal volumétrico a través de la cara superior de la celda *ijk* se calcula con la Ecuación 17, con una variación del volumen de la celda igual a cero (Ecuación 71). Para que haya compatibilidad entre el cálculo de la elevación y de la velocidad vertical, los flujos que se utilizan en esta expresión deben ser los mismos que los que se utilizan para calcular la elevación.

$$U_{3_{ijk+1}}^{*n+1/2} = U_{3_{ijk}}^{*n+1/2} + \frac{\left(Fxz_{ijk}^{n+1/2} - Fxz_{ij+1k}^{n+1/2}\right) + \left(Fyz_{ijk}^{n} - Fyz_{i+1jk}^{n}\right)}{Ah_{ij}}$$

Ecuación 71

$$Fyz_{ijk}^{n} = \frac{U_{2_{ijk}}^{n+1/2} Av_{ijk}^{n-1/2} + U_{2_{ijk}}^{n-1/2} Av_{ijk}^{n-3/2}}{2}$$

Ecuación 72

$$Fxz_{ijk}^{n+1/2} = \frac{U_{1_{ijk}}^{n+1} Au_{ijk}^{n} + U_{1_{ijk}}^{n} Au_{ijk}^{n-1}}{2}$$

Ecuación 73

La Ecuación 71 se resuelve explícitamente en toda la columna de agua utilizando la condición de frontera y realizando el cálculo desde el fondo hasta la superficie. Esta velocidad U_3* calculada de esta forma solo representa una velocidad vertical para el caso en que la malla tenga capas horizontales. En el caso genérico es un caudal por unidad de área que atraviesa la capa.

2.4.5.2. Redefinición de la geometría

Luego de calcular U_3* se puede mover la malla y redefinir la geometría para todos los tipos de malla. Existen dos casos límite para el movimiento de la malla, la malla fija y la malla lagrangeana. Cuando la malla es fija el valor estimado de la velocidad vertical coincide con el valor final, no siendo necesarios los pasos 2 y 3. Con la malla lagrangeana se pretende que la geometría se adapte al escurrimiento de forma que la velocidad vertical sea nula. Por esto se usa el valor de U_3* para determinar la variación de volumen que garantiza divergencia nula (Ecuación 74). En este caso no es necesario el paso 3 ya que con esa ecuación se garantiza que la velocidad vertical sea nula.

$$\frac{Vz_{ijk}^{n+1/2} - Vz_{ijk}^{n}}{Ah_{ij}\,\Delta t/2} = \frac{\left(Fxz_{ijk}^{n+1/2} - Fxz_{ij+1k}^{n+1/2}\right) + \left(Fyz_{ijk}^{n} - Fyz_{i+1jk}^{n}\right)}{Ah_{ij}} = U_{3_{ijk+1}}^{*n+1/2} - U_{3_{ijk}}^{*n+1/2}$$

Ecuación 74

Para el caso de coordenada vertical sigma el movimiento de la malla depende solamente de η y por eso no es función de U_3*. Sin embargo el valor de U_3* se calcula para todas las mallas porque su diferencia representa la divergencia horizontal del campo de velocidades, y se utiliza en el cálculo de la velocidad vertical final para todos los tipos de malla.

Estos diferentes procesos de movimiento de las diferentes mallas resultan en una redefinición de los lugares geométricos de los vértices de las celdas en el nuevo instante de tiempo. El cálculo de la geometría, es decir de áreas y volúmenes de celdas, se realiza después en conjunto para todos los subdominios en un algoritmo único. Este algoritmo es independiente del tipo de malla pues sus medidas geométricas dependen solo de las coordenadas de los vértices.

2.4.5.3. Velocidad vertical final

La velocidad vertical final se calcula nuevamente utilizando la Ecuación 17 pero incluyendo en este caso la variación del volumen de la celda (Ecuación 75). También se puede utilizar la velocidad vertical intermedia para calcular la velocidad vertical; incluyendo la Ecuación 71 en la Ecuación 75 se obtiene la Ecuación 76.

$$U_{3_{ijk+1}}^{n+1/2} = U_{3_{ijk}}^{n+1/2} + \frac{\left(Fxz_{ijk}^{n+1/2} - Fxz_{ij+1k}^{n+1/2}\right) + \left(Fyz_{ijk}^{n} - Fyz_{i+1jk}^{n}\right)}{Ah_{ij}} - \frac{Vz_{ijk}^{n+1/2} - Vz_{ijk}^{n}}{Ah_{ij}\,\Delta t/2}$$

Ecuación 75

$$U_{3_{ijk+1}}^{n+1/2} = U_{3_{ijk}}^{n+1/2} + \left(U_{3_{ijk+1}}^{*n+1/2} - U_{3_{ijk}}^{*n+1/2}\right) - \frac{Vz_{ijk}^{n+1/2} - Vz_{ijk}^{n}}{Ah_{ij}\,\Delta t/2}$$

Ecuación 76

Para garantizar la propiedad de conservación, los volúmenes de las celdas calculados durante la redefinición de la geometría deben ser tales que sumados en toda la columna de agua produzcan una variación de volumen igual a la generada por la variación de la superficie libre. Con el método de volúmenes finitos esta propiedad siempre se satisface si se calculan los mismos flujos en los dos cálculos.

2.4.5.4. Velocidad vertical real

La velocidad vertical calculada anteriormente no es la verdadera componente del campo de velocidades en la dirección vertical, sino que es el flujo volumétrico por unidad de área que atraviesa la cara inferior de la celda. En la Figura 8 se representa una superficie de corriente y una superficie del volumen de control definida por esa cara de la celda.

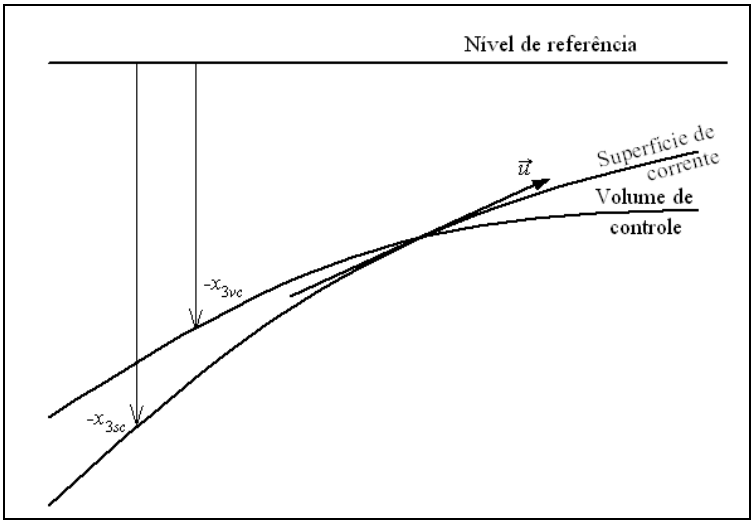

Figura 8: Velocidad real a través del volumen de control.

El campo de velocidades es paralelo a la superficie de corriente y la componente vertical de la velocidad real está dada por la Ecuación 77. La velocidad vertical con la que se mueve una partícula situada sobre la superficie del volumen de control se expresa por la Ecuación 78, y es la velocidad de transporte asociada al movimiento de la malla. La velocidad vertical calculada en la sección anterior es una velocidad relativa a la malla, y es igual a la diferencia entre estas dos últimas velocidades. La velocidad real se calcula entonces según la Ecuación 79.

$$u_3^{real} = \frac{\partial x_{3sc}}{\partial t} + u_1 \frac{\partial x_{3sc}}{\partial x_1} + u_2 \frac{\partial x_{3sc}}{\partial x_2}$$

Ecuación 77

$$u_3^{transporte} = \frac{\partial x_{3vc}}{\partial t} + u_1 \frac{\partial x_{3vc}}{\partial x_1} + u_2 \frac{\partial x_{3vc}}{\partial x_2}$$

Ecuación 78

$$u_3^{real} = u_3 + u_3^{transporte} = u_3 + \frac{\partial x_{3vc}}{\partial t} + u_1 \frac{\partial x_{3vc}}{\partial x_1} + u_2 \frac{\partial x_{3vc}}{\partial x_2}$$

Ecuación 79

La Ecuación 79 puede discretizarse utilizando una discretización progresiva en el tiempo y centrada en el espacio, obteniéndose la Ecuación 80 (a la coordenada vertical del punto P de la Figura 7 se le denomina $SZZ_{ijk}^{n} = -x_{3P_{ijk}}^{n}$). Dicha ecuación puede resolverse explícitamente al final de cada medio paso de tiempo ya que todas las medidas se conocen en dicho instante. De todas formas la velocidad real no es necesaria para calcular la hidrodinámica del sistema ni para el transporte de propiedades. Su cálculo se realiza solo para la representación gráfica del campo de velocidades y se realiza únicamente para los instantes de salida de resultados.

$$U_{3_{ijk}}^{real^{n+1/2}} = U_{3_{ijk}}^{n+1/2} - \frac{SZZ_{ijk-1}^{n+1/2} - SZZ_{ijk-1}^{n}}{\Delta t/2} - U_{1_{ij+1/2k-1/2}}^{n+1/2} \frac{SZZ_{ij+1k-1}^{n+1/2} - SZZ_{ij-1k-1}^{n}}{DZX_{ij-1} + DZX_{ij}} -$$

$$U_{2_{i+1/2\,jk-1/2}}^{n+1/2} \frac{SZZ_{i+1\,jk-1}^{n+1/2} - SZZ_{i-1\,jk-1}^{n}}{DZY_{i-1\,j} + DZY_{ij}}$$

Ecuación 80

2.4.6. Transporte de salinidad y temperatura

La salinidad, temperatura y cualquier otra propiedad conservativa son transportadas de acuerdo a la Ecuación 10 y se discretiza por el método de volúmenes finitos utilizando la Ecuación 13. Los flujos convectivos y difusivos se discretizan de forma semejante a la efectuada para el transporte de cantidad de movimiento, pero son aplicados a la celda de cálculo de propiedades. Los flujos convectivos se discretizan también utilizando la ponderación entre la aproximación "*upwind*" y central utilizada en el transporte de cantidad de movimiento. Los flujos convectivos y los flujos difusivos horizontales se discretizan de forma explícita mientras que los flujos difusivos verticales se discretizan de forma implícita. Esta discretización produce un sistema tridiagonal de ecuaciones que se resuelve eficientemente. Para el esquema S21 en el primer medio paso de tiempo la ecuación discretizada queda de la forma:

$$\frac{Vz_{ijk}^{n+1/2}\alpha_{ijk}^{n+1/2} - Vz_{ijk}^{n}\alpha_{ijk}^{n}}{\Delta t/2} = -\left(\vec{Fc}\cdot\vec{nS}\right)_{ij-1/2k}^{n+1/2} - \left(\vec{Fc}\cdot\vec{nS}\right)_{ij+1/2k}^{n+1/2} - \left(\vec{Fc}\cdot\vec{nS}\right)_{i-1/2\,jk}^{n+1/2} - \left(\vec{Fc}\cdot\vec{nS}\right)_{i+1/2\,jk}^{n+1/2} -$$

$$-\left(\vec{Fc}\cdot\vec{nS}\right)_{ijk-1/2}^{n+1/2} - \left(\vec{Fc}\cdot\vec{nS}\right)_{ijk+1/2}^{n+1/2} + \left(A_{h_{ij-1/2k}}^{n} \frac{\alpha_{ij-1k}^{n} - \alpha_{ijk}^{n}}{DZX_{ij-1k}} \cdot Au_{ijk}^{n} - A_{h_{ij+1/2k}}^{n} \frac{\alpha_{ijk}^{n} - \alpha_{ij+1k}^{n}}{DZX_{ijk}} \cdot Au_{ij+1k}^{n} \right) +$$

$$\left(A_{h_{i-1/2\,jk}}^{n} \frac{\alpha_{i-1\,jk}^{n} - \alpha_{ijk}^{n}}{DZY_{i-1\,jk}} \cdot Av_{ijk}^{n} - A_{h_{i+1/2\,jk}}^{n} \frac{\alpha_{i+1\,jk}^{n} - \alpha_{ijk}^{n}}{DZY_{ijk}} \cdot Av_{i+1\,jk}^{n} \right) + \left(A_{v_{ijk-1}}^{n} \frac{\alpha_{ijk-1}^{n+1/2} - \alpha_{ijk}^{n+1/2}}{DZZ_{ijk-1}^{n+1/2}} - A_{v_{ijk}}^{n} \frac{\alpha_{ijk}^{n+1/2} - \alpha_{ijk+1}^{n+1/2}}{DZZ_{ijk}^{n+1/2}} \right) Ah_{ij}$$

Ecuación 81

Los caudales volumétricos deben ser los mismos que se utilizan en el cálculo de $\eta^{n+1/2}$:

$$\left(\vec{Fc}\cdot\vec{nS}\right)_{ij-\frac{1}{2}k}^{n+1/2} = \left[\gamma \left(\frac{Fxz_{ijk}^{n+1/2} + \left|Fxz_{ijk}^{n+1/2}\right|}{2} \alpha_{ij-1k}^{n} + \frac{Fxz_{ijk}^{n+1/2} - \left|Fxz_{ijk}^{n+1/2}\right|}{2} \alpha_{ijk}^{n} \right) + \right.$$

$$\left. + (1-\gamma) \frac{DXU_{ij}\alpha_{ij-1k}^{n} + DXU_{ij-1}\alpha_{ijk}^{n}}{DXU_{ij} + DXU_{ij-1}} \right] Fxz_{ijk}^{n+1/2}$$

Ecuación 82

$$\left(\vec{Fc}\cdot\vec{nS}\right)_{i-\frac{1}{2}jk}^{n+1/2} = \left[\gamma \left(\frac{Fyz_{ijk}^{n} + \left|Fyz_{ijk}^{n}\right|}{2} \alpha_{i-1\,jk}^{n} + \frac{Fyz_{ijk}^{n} - \left|Fyz_{ijk}^{n}\right|}{2} \alpha_{ijk}^{n} \right) + \right.$$

$$\left. + (1-\gamma) \frac{DXV_{ij}\alpha_{i-1\,jk}^{n} + DXV_{i-1\,j}\alpha_{ijk}^{n}}{DXV_{ij} + DXV_{i-1\,j}} \right] Fyz_{ijk}^{n}$$

Ecuación 83

$$\left(\vec{Fc}\cdot\vec{n}S\right)_{ijk-1/2}^{n+1/2}=\left[\gamma\left(\frac{Fzz_{ijk}^{n+1/2}+\left|Fzz_{ijk}^{n+1/2}\right|}{2}\alpha_{ijk-1}^{n}+\frac{Fzz_{ijk}^{n+1/2}-\left|Fzz_{ijk}^{n+1/2}\right|}{2}\alpha_{ijk}^{n}\right)+\right.$$
$$\left.+\left(1-\gamma\right)\frac{DWZ_{ijk}^{n+1/2}\alpha_{ijk-1}^{n}+DWZ_{ijk-1}^{n+1/2}\alpha_{ijk}^{n}}{DWZ_{ijk}^{n+1/2}+DWZ_{ijk-1}^{n+1/2}}\right]Fzz_{ijk}^{n+1/2}$$

Ecuación 84

2.5. Condiciones iniciales y de borde

Las ecuaciones para el cálculo de la elevación de la superficie libre, de la velocidad y del transporte de salinidad y temperatura forman un sistema de naturaleza parabólica, por lo que necesitan condiciones iniciales en todo el dominio y condiciones de borde durante toda la simulación para poder resolverse.

2.5.1. Condiciones iniciales

Las condiciones iniciales utilizadas en todas las ecuaciones son del tipo de Dirischlet, y se imponen especificando directamente los valores de las variables en todos los puntos del dominio durante el instante inicial. Para la elevación de la superficie libre en general se utiliza un valor de referencia constante para todo el dominio como condición inicial. Para las velocidades en general se utiliza la condición de reposo y solamente es necesario un tiempo de arranque corto para que las mismas se ajusten a la acción de los diferentes forzantes considerados. Para las propiedades del agua como salinidad o temperatura sucede algo similar y se imponen en general valores uniformes en todo el dominio como condición inicial.

2.5.2. Superficie libre

En la superficie libre los flujos convectivos de cantidad de movimiento, salinidad y temperatura son nulos. Esta condición se impone directamente en las ecuaciones considerando que el flujo vertical en las celdas superficiales es cero (Ecuación 85). Por otro lado, el flujo difusivo de cantidad de movimiento se impone explícitamente en la capa superficial a través de la tensión de corte debida al viento. Esta se calcula con la Ecuación 87; siendo C_D el coeficiente de drag función de la velocidad del viento, ρ_a es la densidad del agua y W es la intensidad del viento a 10m sobre la superficie libre.

$$Fzz_{ijkmax+1}=0$$

Ecuación 85

$$\nu\frac{\partial\vec{v}_H}{\partial z}=\vec{\tau}_W$$

Ecuación 86

$$\vec{\tau}_w=C_D\rho_a\vec{W}\left|\vec{W}\right|$$

Ecuación 87

Los flujos difusivos de temperatura y salinidad también se imponen nulos en la superficie libre.

2.5.3. Fondo

También en el fondo la condición de borde impuesta para los flujos advectivos es que sean nulos (Ecuación 88). El flujo difusivo de cantidad de movimiento por la cara inferior de las celdas de fondo se impone como una tensión de corte que se calcula por una ley cuadrática en función de la velocidad de la celda de fondo (Ecuación 89); siendo C_f el coeficiente drag de fondo, κ la constante de Von Karman y z^b_0 el tamaño de la rugosidad absoluta de fondo. Este término se calcula de forma implícita. Los flujos difusivos de temperatura y salinidad en el fondo son nulos.

$$Fzz_{ijkfundo} = 0$$

Ecuación 88

$$\nu \frac{\partial \vec{v}_H}{\partial z} = \tau_{fondo} = C_f \vec{v}_H \left| \vec{v}_H \right|$$

Ecuación 89

$$C_f = \left(\frac{\kappa}{\log\left(\frac{z + z_0^b}{z_0^b} \right)} \right)^2$$

Ecuación 90

2.5.4. Bordes laterales cerrados

Los bordes laterales cerrados son los límites del dominio de cálculo con la tierra. Desde el punto de vista convectivo son fronteras impermeables, lo cual se representa en el modelo imponiendo que los flujos laterales sean nulos en las caras de las celdas que forman parte de estas fronteras (Ecuación 91). En estas fronteras los flujos difusivos de cantidad de movimiento, temperatura y salinidad son cero.

$$Fxz_{ijk} = 0 \qquad ; \qquad Fyz_{ijk} = 0$$

Ecuación 91

2.5.5. Bordes laterales abiertos

Las fronteras laterales abiertas se introducen como forma de limitar el dominio de cálculo a la región de interés. En estas fronteras deben imponerse valores de las variables de cálculo de forma de garantizar que la información correspondiente a lo que sucede fuera del dominio ingrese a la zona de cálculo y deben permitir que las ondas del interior del dominio se propaguen a través de las mismas y que el fluido pase libremente por éstas.

En las ecuaciones hidrodinámicas se pueden imponer valores de caudal o de elevación de la superficie libre, dependiendo de la razón por la cual se definió dicha frontera. La condición de caudal se impone directamente en los flujos horizontales (Fxz y Fyz) en las caras de contacto con esa frontera. La elevación en la

frontera se calcula por continuidad, de forma semejante a la que se realiza en los puntos interiores, salvo que los caudales en la frontera se imponen de forma explícita. Considerando una frontera con un río en *j+1* se tiene para el esquema S21 en el primer medio paso de tiempo la Ecuación 92; siendo Fxz el caudal impuesto. La ecuación resultante es implícita y se resuelve entonces conjuntamente con las ecuaciones de los puntos interiores.

$$\frac{\eta_{ij}^{n+1/2}-\eta_{ij}^{n}}{\Delta t/2}=\frac{1}{Ah_{ij}}\left[\left\{\frac{\sum_{k=kfundo}^{kmax}\left(U_{1_{ijk}}^{n+1}\cdot Au_{ijk}^{n}\right)+\sum_{k=kfundo}^{kmax}\left(U_{1_{ijk}}^{n}\cdot Au_{ijk}^{n-1}\right)}{2}-\sum_{k=kfundo}^{kmax}\left(Fxz_{ij+1k}^{n+1/2}\right)\right\}\right.$$

$$\left.+\left\{\frac{\sum_{k=kfundo}^{kmax}\left(U_{2_{ijk}}^{n+1/2}\cdot Av_{ijk}^{n-1/2}\right)+\sum_{k=kfundo}^{kmax}\left(U_{2_{ijk}}^{n-1/2}\cdot Av_{ijk}^{n-3/2}\right)}{2}-\frac{\sum_{k=kfundo}^{kmax}\left(U_{2_{i+1\,jk}}^{n+1/2}\cdot Av_{i+1\,jk}^{n-1/2}\right)+\sum_{k=kfundo}^{kmax}\left(U_{2_{i+1\,jk}}^{n-1/2}\cdot Av_{i+1\,jk}^{n-3/2}\right)}{2}\right\}\right]$$

Ecuación 92

La condición de nivel se utiliza en las fronteras influenciadas por la marea. En las caras exteriores de esas celdas se calculan las velocidades y los caudales por continuidad a partir de la elevación. También deben especificarse los valores de salinidad y temperatura del agua que ingresa por estas fronteras.

3. Arquitectura del modelo MOHID

El modelo MOHID se encuentra programado en lenguaje Fortran95 siguiendo el paradigma de programación orientada a objetos. El diseño global de MOHID se divide en módulos de Fortran95 que hacen a su vez de clases del sistema. Esto permite a MOHID utilizar algunas de las principales características de la orientación a objetos como polimorfismo, encapsulación, herencia y sobrecarga de funciones. Además, permite que cada módulo sea responsable de manejar ciertos datos asociados a cada uno, así como el manejo de la memoria asociada a estos datos, la cual se reserva dinámicamente. Todos los módulos (objetos) tienen cuatro familias de funciones estándar como lo son el constructor, los métodos selectores, los métodos modificadores y el destructor. Los módulos de MOHID están estructurados según una jerarquía, la cual se presenta en la Figura 9.

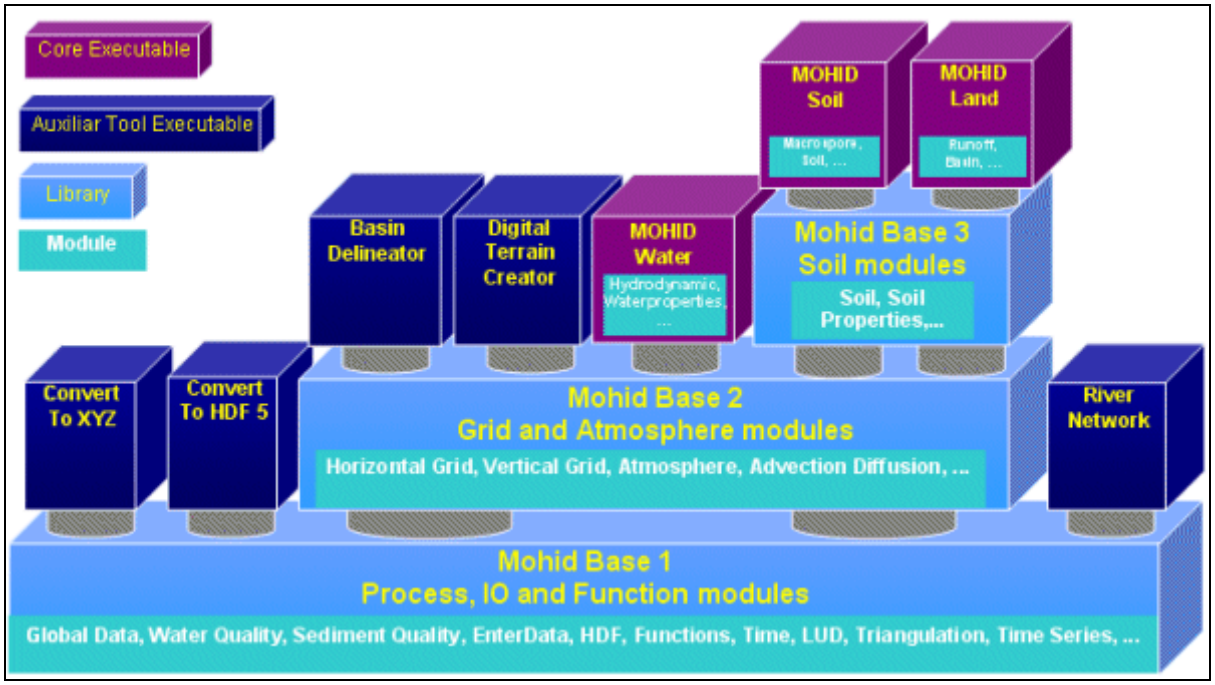

Figura 9- Arquitectura del framework de MOHID [extraido de Mohid, 2003].

El intercambio de la información entre módulos se realiza bajo el paradigma cliente-servidor. Esto significa que cuando una clase necesita algún valor que no pertenece a él, se lo solicita a la clase encargada de manejar ese atributo. Esta estrategia permite asegurar que la información sea encapsulada por cada módulo.

En la base de la arquitectura se encuentran dos bibliotecas fundamentales, denominadas MOHID Base 1 y MOHID Base 2. Estas bibliotecas resuelven problemas específicos, ya sea, numéricos, procesamiento de datos de entrada y salida, etc.

Sobre estas bibliotecas básicas se encuentran los tres núcleos ejecutables principales y otros cinco auxiliares. En lo que sigue se describirá someramente el

núcleo MOHID Water por ser el encargado de la resolución de la hidrodinámica del modelo y el único utilizado en el IMFIA hasta el momento.

3.1. MOHID Water

El programa se compone de una serie de módulos que están construidos sobre las bibliotecas MOHID Base 1 y MOHID Base 2. Los mismos se encargan de calcular los procesos físicos y biogeoquímicos, además del manejo de lectura y escritura de los datos requeridos por el modelo.

Principales módulos de MOHID Water

La estructura de módulos de MOHID se organiza en seis grandes grupos funcionales, cada uno de ellos compuesto por diversos módulos:

Módulos de parámetros globales: se encargan de mantener parámetros globales como el tiempo de simulación, funciones o variables generales:

- Module GlobalData posee principalmente información estática como los identificadores de dominios (generalmente numéricos), identificadores de tipos de error, constantes, parámetros y algunos tipos de datos derivados, usados frecuentemente en otras clases.
- Module Time posee funciones para el manejo del tiempo en MOHID.

Módulos de funciones independientes: manejan operaciones específicas que no se pueden representar en otros módulos o que dada la complejidad de lo que resuelve deben estar en un módulo aparte:

- Module Functions calcula diferentes funciones matemáticas o científicas. Este módulo puede ser visto como una biblioteca matemática disponible a todas las clases.
- Module Triangulation construye una triangulación de Delaunay con el objetivo de proveer la interpolación de niveles ("Gauge") en la frontera abierta del dominio.

Módulos estructurales: son los encargados de la discretización de la geometría del dominio, transformaciones geométricas y referencias para moverse en la jerarquía de los dominios. Se nutre de variables como distancias, áreas, volúmenes, etc:

- Module HorizontalGrid maneja la discretización horizontal.
- Module HorizontalMap realiza un mapeo horizontal de la grilla en 2D.
- Module Map realiza el mapeo 3D de las celdas de la grilla.
- Module Geometry maneja diferentes formas de discretización vertical.

Módulos de manejo de datos: realizan operaciones de entrada/salida:

- Module EnterData lee y escribe archivos en formato ASCII, realiza la lectura de los archivos de entrada y genera los archivos de salida de la simulación.
- Module HDF5 lee y escribe los datos de diferentes matrices en formato HDF5.
- Module TimeSerie lee y escribe archivos con información de la simulación a medida que avanza la ejecución.

Módulos de funciones específicas: implementan operaciones específicas que pueden ser utilizadas solamente por MOHID:

- Module BoxDif se encarga principalmente de inicializar las propiedades definiendo cajas. Una caja o box file define un área en 2D o 3D. Estas áreas pueden ser usadas para inicializar diferentes propiedades o integrar valores entre las diferentes cajas, esto se aplica o se puede aplicar para todas las variables modeladas. Otra característica del módulo BoxDif es que integra los valores de las variables que modela en espacio y tiempo dentro de una caja y entre diferentes cajas. Las operaciones que realiza este módulo se pueden aplicar en un dominio de dos o tres dimensiones.
- Module Statistics computa operaciones estadísticas básicas (en el espacio y tiempo) y va almacenando resultados de interés durante la simulación.
- Module Interface transfiere información (condiciones de fuerza y variables de estado) entre las estructuras de grillas de una, dos y tres dimensiones hacia arreglos unidimensionales y para llamar a los procedimientos de los módulos bioquímicos de dimensión cero.
- Module WaterQuality es un módulo que calcula la dinámica de diversos organismos o microorganismos que viven en cuerpos de agua.

Módulos de procesos, son módulos que corresponden a los diferentes procesos que tienen lugar en los diferentes compartimentos del medio.

A continuación se enumerarán los principales módulos del sistema y una breve descripción de ellos.

3.1.1.1. Module Model

Es el módulo principal en la arquitectura del sistema. El módulo Model maneja toda la información de un solo modelo. Es responsable de construir, modificar y destruir cada modelo. También se encarga de controlar el flujo de información entre diferentes modelos y la evolución del tiempo de simulación. Coordina la ejecución del módulo Hydrodynamic y el módulo Transport; y depende de los módulos Hydrodynamic, WaterProperties y Lagrangian.

La coordinación de ejecución del modelo consiste en la actualización del tiempo global del modelo y la actualización de los módulos Transport e Hydrodynamic en un solo modelo.

Cuando se trabaja con modelos encajados las comunicaciones entre modelos se hacen en una sola vía y de forma recursiva, es decir, las condiciones iniciales y otros datos que se calculan durante la simulación se envían desde el padre al hijo.

3.1.1.2. Module Geometry

El módulo Geometry almacena y actualiza la información sobre los volúmenes finitos, calculando las áreas laterales y los volúmenes de la discretización según la demanda. Para realizar estos cálculos se basa en la elevación de la superficie libre calculada por el módulo hidrodinámico y en los datos batimétricos del dominio ingresados.

El módulo Geometry proporciona datos a los módulos lagrangian, turbulence y water properties.

3.1.1.3. Module Hydrodynamic

El módulo Hydrodynamic calcula el nivel, la velocidad y el flujo de agua en cada paso de tiempo. La discretización espacial es realizada por una aproximación de volúmenes finitos y la discretización temporal se realiza con un algoritmo semi-implícito ADI (Alternating Direction Implicit).

Este algoritmo computa alternativamente una componente de la velocidad horizontal implícitamente mientras la otra es calculada explícitamente. El sistema de ecuaciones resultantes es tridiagonal y por lo tanto puede ser resuelto de manera eficiente por el algoritmo de Thomas.

3.1.1.4. Module Lagrangian

Es un modelo de transporte lagrangiano (gestiona las mismas propiedades que el módulo WaterProperties). Puede ser utilizado, por ejemplo, para la simulación de la dispersión del petróleo. El módulo lagrangiano del MOHID utiliza el concepto de trazador, cuyas propiedades fundamentales son la posición espacial (x, y, z) de las partículas utilizadas como trazadores, su volumen y la concentración de determinadas propiedades de interés, como pueden ser cualquiera de las evaluadas en el módulo de calidad de aguas (por ejemplo algún contaminante).

3.1.1.5. Module Oil

El módulo Oil se encarga de la simulación del movimiento de derrames de petróleo. Este tipo de herramientas cobran vital importancia a la hora de desarrollar planes de contingencia, permitiendo mitigar alguna catástrofe con este producto en el mar. El módulo también permite la evaluación de algunas características del impacto ambiental asociado al movimiento de petróleo. Por ejemplo, MOHID fue utilizado para calcular el posible impacto y el movimiento de la mancha de petróleo en la catástrofe del buque Prestige en las costas de Galicia en Noviembre de 2002.

El módulo Oil depende de los módulos Surface (presión atmosférica, oleaje, viento), WaterProperties (salinidad, temperatura, sedimentos cohesivos), Lagrangian.

3.1.1.6. Module WaterProperties

El módulo WaterProperties coordina y maneja la evolución de las propiedades del agua utilizando un modelo Euleriano de transporte. Para llevar a cabo esta tarea, se apoya o usa otros módulos como el de AdvectionDifussion (Advección-Difusión), encargado de calcular el transporte (por difusión o advección) de las propiedades, o el módulo WaterQuality (calidad del agua), el cual es uno de los tres módulos encargados de calcular procesos biogeoquímicos.

A través de este módulo, MOHID es capaz de simular diferentes propiedades como la temperatura, salinidad, sedimentos cohesivos, fitoplancton, nutrientes, contaminantes, etc.

3.1.1.7. Module WaterQuality

El módulo WaterQuality simula la producción primaria y secundaria, y el ciclo de los nutrientes de ecosistemas. Los forzantes principales son la temperatura y la luz. El módulo fue desarrollado en términos de fuentes y sumideros de ciertas propiedades, o dicho de otra manera, definiendo flujos de entrada y salidas al sistema, lo cual permite un fácil acoplamiento al módulo de transporte en ambas formulaciones, Euleriana y Lagrangiana. Debido a la interdependencia de las propiedades, un sistema lineal de ecuaciones es calculado para cada volumen de control.

3.1.1.8. Module Surface

El módulo Surface plantea las condiciones de frontera en la superficie de la columna de agua y representa la influencia de forzantes externos atmosféricos como el viento y el sol en la superficie del agua.

Hay dos tipos de condiciones. Una dada por el usuario, usualmente datos meteorológicos (velocidad del viento, temperatura del aire, etc), y otra como condiciones de borde calculadas automáticamente por el modelo a partir de las condiciones o datos meteorológicos.

3.1.1.9. Module FreeVerticalMovement

El módulo FreeVerticalMovement calcula las propiedades del flujo vertical. Básicamente se usa para determinar el movimiento vertical del flujo (en la dirección vertical). Usualmente se utiliza este módulo para calcular la velocidad de caída al simular sedimentos cohesivos, o la velocidad de distintas partículas en las simulaciones de transporte.

3.1.1.10. Module HydrodynamicFile

El módulo HydrodynamicFile es un módulo auxiliar que le permite al usuario de MOHID integrar espacial y temporalmente la solución obtenida con el modelo en un archivo.

La integración espacial consiste en unir varias celdas para tratarlas como una sola. Por otro lado, la integración temporal consiste en juntar varios pasos discretos de tiempo de la solución hidrodinámica y puede ser directamente conectada con la integración espacial.

3.1.1.11. Module Turbulence

El módulo Turbulence calcula los coeficientes de viscosidad horizontal y vertical y las difusividades para diferentes métodos de cierre de turbulencia. Estas propiedades pueden ser calculadas de una manera simplificada utilizando coeficientes de difusión constantes. Por otro lado el usuario puede calcular la evolución de las propiedades de flujo turbulento de una manera más realista mediante el modelo GOTM (Global Ocean Turbulence Model) [19]. Una interfase de este módulo calcula los coeficientes utilizando un cierre de turbulencia de una o dos ecuaciones con la subrutina tomada del modelo general de turbulencia oceánica GOTM.

3.2. Ejecución del modelo

La simulación computacional de un paso de tiempo en MOHID se puede dividir en dos partes, una parte de actualización del tiempo y redefinición de la frontera; y otra parte de ejecución y cálculo de acuerdo a los datos iniciales. El procedimiento que se encarga de la actualización es UpdateTimeAndMapping y el procedimiento que lleva a cabo el cálculo es RunModel.

La primer parte de actualización del tiempo que se está simulando, se basa en el campo DT definido en el archivo de entrada que se indica en la Keyword "IN_MODEL". Además se controla si se llegó al final de la simulación o no tomando en cuenta los campos start y end del archivo anteriormente mencionado.

Por otro lado, para actualizar la frontera, se toma en cuenta el nivel del agua y allí se establece qué puntos son de agua y cuales son tierra en el dominio de la simulación.

La segunda parte se dedica a realizar cálculos del modelo en sí mismo. Se resuelven las ecuaciones discretizadas según el método ADI que se implementa y se calcula la evolución en el tiempo de las variables modeladas en cada celda de la grilla, por ejemplo: velocidades, salinidad, temperatura, parámetros de turbulencia, elevación de superficie libre, etc.

Cabe mencionar que cuando se trabaja con dominios encajados se efectúa la comunicación de datos entre el modelo padre y el modelo hijo, en el cual el modelo hijo obtiene los datos de las diferentes variables de cálculo como condición de borde. Esto se realiza entre la actualización del tiempo y los cálculos del modelo mediante el procedimiento llamado SubModelCommunication.

4. Metodologías empeladas por el modelo en la resolución de sistemas lineales

Dentro del código del MOHID existen implementaciones de varios métodos de resolución de sistemas lineales. Si bien como se mostró en el capítulo anterior el modelo resuelve principalmente sistemas tridiagonales, bajo algunas configuraciones del modelo como por ejemplo aquellas que no asumen la hipótesis de distribución hidrostática de presiones, el modelo deber resolver sistemas que no son tridiagonales y para eso tiene implementados otros métodos más generales.

Este capítulo se centra en el estudio de las técnicas utilizadas para la resolución de sistemas lineales presentes en el modelo hidrodinámico de volúmenes finitos MOHID. En este capítulo se presenta una breve reseña teórica acerca de los métodos implementados en el modelo presentado. Por un lado el único método directo implementado que es el que se utiliza en la resolución de sistemas tridiagonales y por otro lado los métodos iterativos más generales. Los contenidos aquí presentados se basan fuertemente en el libro de Ferziger y Períc (2002).

Por último se presentan algunos comentarios acerca de la implementación del algoritmo que se utiliza para la resolución de sistemas tridiagonales en el código del modelo por ser el método más utilizado. Se realizaron simulaciones de prueba a los efectos de evaluar dónde el modelo resuelve sistemas lineales, cómo lo hace y que fracción del tiempo de la simulación representa su resolución. Para estose instrumentó el código fuente de manera de poder identificar la ruta dentro del mismo que conduce a la ejecución de las subrutinas que resuelven sistemas lineales.

4.1. Métodos directos

4.1.1. El algoritmo de Thomas

El algoritmo de Thomas (o Tridiagonal Matrix Algorithm - TDMA) es una versión simplificada del método de Gauss que es usado para resolver sistemas tridiagonales de ecuaciones, por ejemplo:

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = y_i \qquad i = 1, \ldots, n$$

Ecuación 93

o en forma matricial:

$$\begin{pmatrix} b_1 & c_1 & 0 & . & . & . & . & 0 \\ a_2 & b_2 & c_2 & 0 & . & . & . & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & . & . & 0 \\ & & & . & & & & \\ & & & . & & & & \\ & & & . & & & & \\ 0 & 0 & 0 & . & . & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & . & . & . & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_n \end{pmatrix}$$

El TDMA consiste al igual que el método de Gauss en dos etapas, una de eliminación y una de sustitución (forward elmination & backward sustitution). Si se considera el sistema dado por la Ecuación 93 y se modifica la primera ecuación de la siguiente forma:

$$Ec_{i=1}.b_1 - Ec_{i=2}.a_2$$

resulta

$$(b_1b_2 - c_1a_2)x_2 + c_2b_1x_3 = b_1y_2 - a_2y_1$$

De esta forma se ha eliminado x_1 de la segunda ecuación. Procediendo en igual forma con la ecuación modificada y la tercera ecuación se puede eliminar x_2:

$$(b_1b_2 - c_1a_2)Ec_{i=3} - a_3(\text{modif}.Ec_{i=2})$$

obteniendo

$$\left(b_3(b_1b_2 - c_1a_2) - c_2b_1a_3\right)x_3 + c_3(b_1b_2 - c_1a_2)x_4 = (b_1b_2 - c_1a_2)y_3 - (b_1y_2 - a_2y_1)a_3$$

Si el procedimiento se repite hasta la n-ésima ecuación, la última ecuación obtenida involucrará únicamente la variable x_n. A partir de ésta comienza la etapa de sustitución resolviendo la ecuación modificada *n-1* para obtener x_{n-1}, y así hasta resolver todas las incógnitas. Si escribimos la relación de sustitución como:

$$x_{i-1} = \gamma_i x_i + \beta_i$$

Ecuación 94

Si se sustituye la expresión de la Ecuación 94 en la

Ecuación 93 y se resuelve respecto a x_i se obtiene la siguiente relación:

$$\gamma_i = \frac{-c_i}{a_i\gamma_i + b_i}, \quad \beta_{i+1} = \frac{y_i - a_i\beta_i}{a_i\gamma_i + b_i}$$

Ecuación 95

Las ecuaciones 95 dan las fórmulas de recursión para los coeficientes γ_i y β_i para $i = 2,\ldots,n-1$. Los valores γ_1 y β_1 se pueden obtener a partir de la primera ecuación (*i=1*):

$$x_1 = \frac{y_1}{b_1} - \frac{c_1}{b_1} x_2 \quad \Rightarrow \quad \gamma_2 = -\frac{c_1}{b_1}, \quad \beta_2 = \frac{1}{b_1} \quad \Rightarrow \quad \gamma_1 = \beta_1 = 0$$

Por último la ecuación para obtener x_n y comenzar la iteración será:

$$a_n(\gamma_n x_n + \beta_n) + b_n x_n = y_n$$

obteniendo

$x_n = \frac{y_n - a_n \beta_n}{a_n \gamma_n + b_n}$, el cual puede ser obtenido directamente de la $x_{i-1} = \gamma_i x_i + \beta_i$

Ecuación 94 imponiendo $x_{n+1} = 0$.

Por tanto el procedimiento se puede resumir en el siguiente algoritmo:

1 – Establecer $\gamma_1 = \beta_1 = 0$

2 – Evaluar para $i = 1, \ldots, n-1$

$$\gamma_i = \frac{-c_i}{a_i \gamma_i + b_i}, \quad \beta_{i+1} = \frac{y_i - a_i \beta_i}{a_i \gamma_i + b_i}$$

3 – Establecer $x_{n+1} = 0$

4 – Evaluar para $i = n+1, \ldots, 2$

Algoritmo 1: Thomas.

La importante ventaja de este algoritmo es que permite resolver el sistema en $\Theta(n)$ operaciones mientras el método de Gauss en $\Theta(n^3)$.

4.2. Métodos iterativos

Cualquier sistema de ecuaciones determinado puede ser resuelto por eliminación gaussiana o por ejemplo mediante una factorización LU. Sin embargo, la factorización cuando se trata de una matriz dispersa presenta el problema de que las matrices triangulares obtenidas pierden dicha propiedad elevando el costo del método. Además usualmente el error que se comete en la discretización de las ecuaciones es mayor que el error de redondeo, por tanto no es necesario resolver el sistema con tanta precisión sino que basta con hacerlo de manera más precisa que el error de descretización.

Todo esto da lugar a los métodos iterativos, en los cuales se propone una solución inicial y se utilizan las ecuaciones para mejorarla sucesivamente hasta acercarse a la solución exacta tanto como se quiera. Estos métodos permiten resolver sistemas no lineales, pero también sistemas lineales que presenten las características mencionadas en el párrafo anterior pueden ser resueltos en menos tiempo en la medida que cada iteración sea barata y el número de iteraciones no sea demasiado grande.

La idea básica de estos métodos es la siguiente, si se tiene el sistema de ecuaciones representado en forma matricial en la Ecuación 96.

$$A\phi = Q$$

Ecuación 96

Luego de *n* iteraciones se tendrá una aproximación para la solución ϕ^n que no verifica las ecuaciones exactamente, por lo que existe un residuo ρ^n que cumple:

$$A\phi^n = Q - \rho^n$$

Ecuación 97

Si se define el error de iteración como:

$$\varepsilon^n = \phi - \phi^n$$

Ecuación 98

Restando las ecuaciones 96 y 97 se obtiene una relación entre el residuo y el error de iteración:

$$A\varepsilon^n = \rho^n$$

Ecuación 99

El objetivo de la iteración es reducir el residuo a cero, en ese proceso, el error también ser reducirá a cero. Para ver esto, considérese el siguiente esquema de iteración:

$$M\phi^{n+1} = N\phi^n + B$$

Ecuación 100

Naturalmente se debe exigir al método iterativo que la solución a la que se converge satisfaga la Ecuación 96, de esta manera por definición en la convergencia se tendrá $\phi^{n+1} = \phi^n = \phi$, y por tanto se cumple:

$$A = M - N \text{ y } B = Q$$

Ecuación 101

o en forma más general,

$$PA = M \text{ y } B = PQ$$

Ecuación 102

siendo *Q* un pre-condicionador.

Una alternativa es sustraer $M\phi^n$ a cada lado de la Ecuación 100, obteniendo así:

$$M(\phi^{n+1} - \phi^n) = B - (M - N)\phi^n \quad \text{o} \quad M\delta^n = \rho^n$$

Ecuación 103

donde $\delta^n = \phi^{n+1} - \phi^n$ es la "corrección" o actualización.

Para que el método iterativo sea efectivo, resolver el sistema de la Ecuación 100 debe ser barato y el método deber converger rápido. Para que la iteración no sea costosa, el cálculo de $N\phi^n$ y la resolución del sistema debe ser sencilla de llevar a cabo. Lo primero no es difícil pues si A es dispersa también los será N, lo segundo requiere que la matriz de iteración M sea fácilmente invertible. Para una rápida convergencia M deber ser una buena aproximación de A, haciendo así $N\phi$ pequeño en algún sentido.

4.2.1. Descomposición LU incompleta de Stone (SIP - ILU solver after Stone)

La descomposición LU es un método muy bueno en general pero no aprovecha la ventaja de dispersión de las matrices. En un método iterativo, si M es una buena aproximación de A, se tendrá una rápida convergencia. Estas dos observaciones conducen a la idea de utilizar una factorización LU aproximada de A como matriz de iteración M, por ejemplo:

$$M = LU = A + N$$

Ecuación 104

donde L y U son ambas dispersas y N es chica.

Una versión de este método para matrices simétricas, conocida como factorización incompleta de Cholesky se usa usualmente en conjunto con métodos de gradiente conjugado que se describen en el siguiente punto. Dado que las matrices que se obtienen de discretizar problemas que involucran las ecuaciones de Navier-Stokes no son simétricas, este método no se les puede aplicar. Una versión asimétrica del método llamada factorización LU incompleta (ILU) es posible pero no tiene un uso muy frecuente. En el método ILU se procede igual que en la descomposición LU pero para cada elemento de la matriz original A que es cero, el elemento correspondiente en L o U se fija igual a cero. Esta factorización no es exacta, pero el producto de estos factores se puede utilizar como matriz de iteración M para el método iterativo. Sin embargo este método converge de manera bastante lenta.

Otra descomposición incompleta es la propuesta por Stone en 1968, conocida como SIP - *Strongly Implicit Procedure*, y está diseñada específicamente para sistemas de ecuaciones algebraicas que derivan de la discretización de ecuaciones diferenciales y no aplica para sistemas de ecuaciones genéricos.

Con el objetivo de presentar los conceptos básicos del método a continuación se describe el método SIP para una molécula de cálculo de cinco puntos como se realiza en Ferziger & M.Peric (2002). Los mismos principios se aplican para construirlo para una molécula de 7 y 9 puntos en 3D.

Tal como en la ILU, las matrices L y U tienen elementos no nulos solo en las diagonales en los cuales A tiene elementos no nulos. El producto de las matrices triangulares inferior y superior con estas estructuras tiene más diagonales con elementos no nulos que A. Para la molécula de cinco puntos hay dos diagonales

más, y para una molécula 3D de siete puntos aparecen seis diagonales más. Si el ordenamiento de nodos es el siguiente:

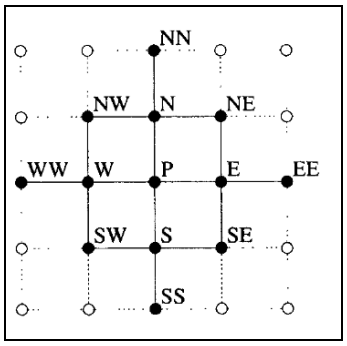

Figura 10

En ese caso las dos diagonales extras corresponden a los nodos NW y SE.

Para hacer estas matrices únicas, cada elemento en la diagonal de U es fijado igual a 1. De esta forma cinco sets de elementos deben ser determinados (tres en L y dos en U).

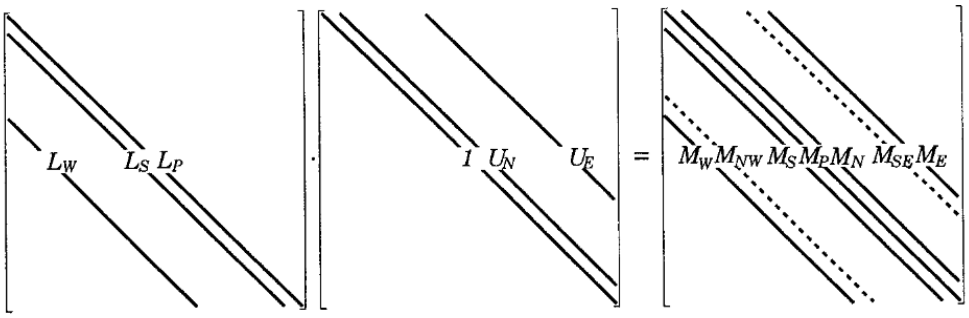

Figura 11: Esquema de las matrices L y U, y su producto M; las diagonales de M que no aparecen en A aparecen en punteado.

Si las matrices tiene la forma que se muestran en la Figura 11, realizando el producto entre las matrices L y U se obtiene que M es:

$$
\begin{aligned}
M_W^l &= L_W^l \\
M_{NW}^l &= L_W^l U_N^{l-Nj} \\
M_S^l &= L_S^l \\
M_P^l &= L_W^l U_E^{l-Nj} + L_S^l U_N^{l-1} + L_P^l \\
M_N^l &= U_N^l L_P^l \\
M_{SE}^l &= L_S^l U_E^{l-1} \\
M_E^l &= U_E^l L_P^l
\end{aligned}
$$

Ecuación 105

La idea es seleccionar L y U tal que M sea una buena aproximación de A. Como mínimo, N debe contener las dos diagonales de M que son cero en A. Esto es posible, de hecho, es el método ILU standard mencionado anteriormente, el cual como se dijo anteriormente converge de manera muy lenta.

Stone (1968) reconoció que la convergencia podría ser mejorada permitiendo que N tuviera elementos no nulos en las siete diagonales con

elementos no nulos de LU. El método se deriva más fácilmente considerando el vector $M\phi$:

$$(M\phi)_P = M_P\phi_P + M_S\phi_S + M_N\phi_N + M_E\phi_E + M_W\phi_W + M_{NW}\phi_{NW} + M_{SE}\phi_{SE}$$

Ecuación 106

Los últimos dos términos son los "extra", cada término en esta ecuación se corresponde a una diagonal de la matriz $M = LU$.

La matriz N debe contener las dos diagonales "extra" de M, y se quiere elegir los de las demás diagonales de manera tal que $N\phi \approx 0$ o en otras palabras:

$$N_P\phi_P + N_N\phi_N + N_S\phi_S + N_E\phi_E + N_W\phi_W + M_{NW}\phi_{NW} + M_{SE}\phi_{SE} \approx 0$$

Ecuación 107

Esto requiere que la contribución de los dos términos "extras" en la Ecuación 107 sean casi cancelados por la contribución de las otras diagonales. En otras palabras, la Ecuación 107 se debe reducir a:

$$M_{NW}\left(\phi_{NW} - \phi^*_{NW}\right) + M_{SE}\left(\phi_{SE} - \phi^*_{SE}\right) \approx 0$$

Ecuación 108

donde ϕ^*_{NW} y ϕ^*_{SE} son aproximaciones a ϕ_{NW} y ϕ_{SE}.

La idea central de Stone es la siguiente, como las ecuaciones aproximan a una ecuación diferencial elíptica, la solución se espera que sea "suave". Si esto es así, ϕ^*_{NW} y ϕ^*_{SE} pueden ser aproximados en términos de los valores de ϕ en los nodos correspondientes a las diagonales de A. Stone propuso la siguiente aproximación (luego aparecieron otras):

$$\phi^*_{NW} \approx \alpha\left(\phi_W + \phi_N - \phi_P\right),\ \phi^*_{SE} \approx \alpha\left(\phi_S + \phi_E - \phi_P\right)$$

Ecuación 109

Si $\alpha = 1$, se tiene una interpolación de segundo orden pero Stone encontró que para asegurar la estabilidad α debe ser menor que 1. Estas aproximaciones se basan en las propiedades de estas ecuaciones diferenciales y tienen poco sentido para ecuaciones algebraicas en general.

Si las aproximaciones de la Ecuación 109 se sustituyen en la Ecuación 108 se pueden obtener todos los elementos de N como combinación lineal de M_{NW} y M_{SE}. Los elementos de M, ver ecuación 105, pueden igualarse a la suma de elementos de A y N. Las ecuaciones resultantes no bastan para determinar todos los elementos de L y U, pero pueden resolverse en un orden secuencial comenzando desde la esquina SW de la grilla, esto es:

$$L_W^l = A_W^l / \left(1+\alpha U_N^{l-Nj}\right)$$
$$L_S^l = A_S^l / \left(1+\alpha U_E^{l-1}\right)$$
$$L_P^l = A_P^l + \alpha\left(L_W^l U_N^{l-Nj} + L_S^l U_E^{l-1}\right) - L_W^l U_E^{l-Nj} - L_S^l U_N^{l-1}$$
$$U_N^l = \left(A_N^l - \alpha L_W^l U_N^{l-Nj}\right) / L_P^l$$
$$U_E^l = \left(A_E^l - \alpha L_S^l U_E^{l-1}\right) / L_P^l$$

Ecuación 110

Los coeficientes deben ser calculados en ese orden. Para nodos adyacentes a la frontera, todo nodo con un índice correspondiente a la frontera es igualado a cero. Por tanto a lo largo de la frontera oeste (i=2), los elementos con índice $l - N_j$ son cero; a lo largo de la frontera sur (j=2), los elementos con índice $l-1$ son cero; a la largo de la frontera norte ($j = N_j - 1$), los elementos con índice $l+1$ son cero; y finalmente a lo largo de la frontera este ($i = N_i - 1$); los elementos con índice $l + N_j$ son cero.

Ahora se pasa a resolver el sistema con la ayuda de esta factorización aproximada. La ecuación que da la actualización del residuo es:

$$LU\delta^{n+1} = \rho^n$$

Ecuación 111

Las ecuaciones son resueltas tal como en la descomposición LU genérica. Si se multiplica la Ecuación 111 por L^{-1} se obtiene:

$$U\delta^{n+1} = L^{-1}\rho^n = R^n$$

Ecuación 112

R^n se puede calcular fácilmente a partir de:

$$R^l = \left(\rho^l - L_S^l R^{l-1} - L_W^l R^{l-Nj}\right) / L_P^l$$

Ecuación 113

Esta ecuación es resulta avanzando en un orden creciente de l. Cuando el cálculo de R esta completo, se debe resolver la Ecuación 112:

$$\delta^l = R^l - U_N^l \delta^{l+1} - U_E^l \delta^{l+Nj}$$

Ecuación 114

en orden decreciente del índice l.

En el método SIP los elementos de las matrices L y U solo necesitan ser calculados una vez, antes de la primera iteración. En las siguientes iteraciones, solo se debe calcular el residuo, luego R y finalmente δ, resolviendo dos sistemas triangulares.

El método de Stone en general converge en pocos pasos de iteración, y la tasa de convergencia se puede mejorar variando α de iteración en iteración y de

punto a punto. Sin embargo esto último implica realizar la factorización cada vez que α es modificado. Como hallar L y U suele ser tan costoso como una iteración con una descomposición dada, suele ser más eficiente mantener un α fijo.

A diferencia de otros, el método de Stone es un buen método iterativo en si mismo, pero además es una buena base para otros métodos, por ejemplo como precondicionador para los de gradiente conjugado y como suavizador para los métodos multigrilla.

4.2.2. Pre-Conditionned Conjugate Gradient Solver for symmetric matrices (CGS2D)

Dentro de los métodos de resolución de sistemas no lineales, existe la rama de los métodos tipo Newton y por otro lado los denominados métodos globales. Los primeros son muy rápidos pero dependen fuertemente de la cercanía de la solución inicial respecto a la verdadera y si dicha aproximación es mala pueden incluso no converger. Los métodos globales siempre encuentran la solución si esta existe, pero son más lentos. La mayoría son métodos descendentes, que convierten el sistema original de ecuaciones en un problema de minimización. Si la matriz *A* de la Ecuación 96 es definida positiva (simétrica y con valores propios positivos), entonces resolver dicha ecuación es equivalente a encontrar el mínimo de:

$$F = \frac{1}{2}\phi^T A\phi - \phi^T Q = \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} A_{ij}\phi_i\phi_j - \sum_{i=1}^{n}\phi_i Q_i$$

Ecuación 115

con respecto de todas las ϕ_i, esto se verifica tomando las derivadas de *F* respecto a cada variable e imponiendo que su valor sea nulo.

Uno de los métodos implementados en el MOHID es el Gradiente conjugado precondicionado para matrices simétricas. El método del gradiente conjugado se basa en lo siguiente: es posible minimizar una función con respecto a varias direcciones simultáneamente buscando en una dirección por vez. Esto es posible eligiendo cuidadosamente las direcciones en las cuales avanzar. A modo de ejemplo para el caso de dos direcciones, supóngase que se deben determinar los valores α_1 y α_2 tales que satisfacen:

$$\phi = \phi^0 + \alpha_1 p^1 + \alpha_2 p^2$$

Ecuación 116

y minimizan *F*, esto es, se trata de minimizar *F* en el plano $p^1 - p^2$. Este problema se puede reducir al problema respecto de p^1 y p^2 por separado siempre y cuando estas direcciones sean conjugadas en el siguiente sentido:

$$p^1.Ap^2 = 0$$

Ecuación 117

Esta propiedad es similar a la de ortogonalidad, los vectores p^1 y p^2 se dice que son conjugados respecto de la matriz *A*, lo cual le da el nombre al método.

Esto se puede extender a cualquier número de direcciones, debiendo ser cada nueva dirección conjugada de todas las anteriores.

Si bien el método garantiza que el error disminuye en cada iteración, cuánto disminuye depende de la dirección de búsqueda. Es usual en este método que el error se reduzca en pequeña proporción para un número de iteraciones y luego encontrar una dirección para la que error se reduce en un orden o más en solo una iteración.

Se puede mostrar que la tasa de convergencia del método depende del número de condición k de la matriz,

$$k = \frac{\lambda_{\max}}{\lambda_{\min}}$$

Ecuación 118

siendo $\lambda_{\max}$ y $\lambda_{\min}$ el mayor y menor valor propio de la matriz respectivamente. El método de gradiente conjugado por sí solo no es muy eficiente, pues para problemas de mecánica de los fluidos en general el número de condición es tal que la velocidad de convergencia no es lo suficientemente rápida como para justificar su uso. Sin embargo, el método puede ser mejorado si se reemplaza el problema que se desea resolver por uno con la misma solución pero con un número de condición menor, esto recibe el nombre de "pre-condicionar" el problema. Una manera de pre-condicionar el problema es multiplicar previamente la matriz por otra elegida cuidadosamente. Como esto podría destruir la simetría el precondicionamiento debe tomar la siguiente forma:

$$C^{-1}AC^{-1}C\phi = C^{-1}Q$$

Ecuación 119

El método del gradiente conjugado es aplicado así a la matriz $C^{-1}AC^{-1}$, y si se usa la forma residual del método resulta el siguiente algoritmo, donde ρ^k es el residuo en la *k*-ésima iteración, p^k es la *k*-ésima dirección de búsqueda, z^k es un vector auxiliar y α^k, β^k son los parámetros usados para construir la nueva solución, residuo y dirección de búsqueda. La derivación detalla de este algoritmo se encuentra en Golub & van Loan (1990).

1 – Inicialización estableciendo:

$k = 0,\ \phi^0 = \phi^{ini},\ \rho^0 = Q - A\phi_{ini},\ p^0 = 0,\ s_0 = 10^{30}$

2 – Avanzar el contador: $k = k + 1$

3 – Resolver el sistema: $Mz^k = \rho^{k-1}$

4 – Calcular:

$$s^k = \rho^{k-1}.z^k$$

$$\beta^k = s^k / s^{k-1}$$

$$p^k = z^k + \beta^k p^{k-1}$$

$$\alpha^k = s^k / (p^k.Ap^k)$$

$$\phi^k = \phi^{k-1} + \alpha^k p^k$$

$$\rho^k = \rho^{k-1} - \alpha^k Ap^k$$

5 – Repetir hasta la convergencia.

Algoritmo 2: Método del gradiente conjugado pre-condicionado.

Este algoritmo involucra la resolución de un sistema lineal como primer paso. La matriz que involucra es $M = C^{-1}$ donde C es la matriz pre-condicionadora. Para que el método sea eficiente, M debe ser fácilmente invertible. Una de las elecciones más comunes para M es la factorización de Cholesky para A, sin embargo se ha encontrado que si $M = LU$ donde L y U son los factores usados en el método de Stone SIP la convergencia es más rápida.

4.2.3. Bi-Conjugate Gradient Stabilized Algorithm

El método del gradiente conjugado visto en el punto anterior es aplicable solo a matrices simétricas, para aplicarlo a no necesariamente simétricos una opción es convertir el problema en uno simétrico. Hay muchas formas de hacer esto siendo la siguiente la más simple, considérese el siguiente sistema:

$$\begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \cdot \begin{pmatrix} \psi \\ \phi \end{pmatrix} = \begin{pmatrix} Q \\ 0 \end{pmatrix}$$

Ecuación 120

Este sistema puede ser descompuesto en dos subsistemas. El primero es el sistema original, el segundo involucra la traspuesta de la matriz y es irrelevante (si fuera necesario hacer es posible resolver el sistema que involucra la matriz traspuesta con un pequeño costo extra). Cuando el método del gradiente conjugado pre-condicionado es aplicado a este sistema, resulta el siguiente método, denominado gradiente biconjugado:

1 – Inicialización estableciendo:

$k = 0,\ \phi^0 = \phi^{ini},\ \rho^0 = Q - A\phi_{ini},\ \bar{\rho}^0 = Q - A^T\phi_{ini}$

$p^0 = \bar{p}^0 = 0,\ s_0 = 10^{30}$

2 – Avanzar el contador: $k = k + 1$

3 – Resolver el sistema: $Mz^k = \rho^{k-1},\ M^T\bar{z}^k = \bar{\rho}^{k-1}$

4 – Calcular:

$$s^k = \bar{\rho}^{k-1}.z^k$$

$$\beta^k = s^k / s^{k-1}$$

$$p^k = z^k + \beta^k p^{k-1}$$

$$\bar{p}^k = \bar{z}^k + \beta^k \bar{p}^{k-1}$$

$$\alpha^k = s^k / (\bar{p}^k.Ap^k)$$

$$\phi^k = \phi^{k-1} + \alpha^k p^k$$

$$\rho^k = \rho^{k-1} - \alpha^k Ap^k$$

$$\bar{\rho}^k = \bar{\rho}^{k-1} - \alpha^k A^T \bar{p}^k$$

5 – Repetir hasta la convergencia.

Algoritmo 3: Método del gradiente biconjugado.

Este algoritmo fue publicado por Fletcher (1976). Tal como se presenta cada iteración requiere el doble de esfuerzo que una iteración del método de gradiente conjugado, y converge en aproximadamente en el mismo número de iteraciones. Su uso no es común en problemas de mecánica de fluidos computacional, pero tiene la ventaja de ser un método muy robusto que permite resolver un amplio espectro de problemas.

4.3. Implementación del algoritmo de Thomas en el MOHID

A continuación se mencionarán algunos aspectos generales de la implementación del algoritmo de Thomas en el código MOHID, únicamente se trata este método por ser el único utilizado en las configuraciones usualmente utilizadas en el IMFIA.

El algoritmo de Thomas se encuentra implementado como una subrutina dentro del módulo ModuleFunctions. Este módulo forma parte de la biblioteca MOHID Base 1.

En el Anexo 1 se adjuntan los códigos de las distintas implementaciones contenidas en la rutina ModuleFunctions.

Existen varias subrutinas que implementan el algoritmo de Thomas:

- subroutine THOMAS_2D
- subroutine THOMAS_3D_NoOpenMP
- subroutine THOMAS_3D_OpenMP
- subroutine THOMAS_3D_i0_j1
- subroutine THOMAS_3D_i1_j0
- subroutine THOMAS_3D_i0_j1_OMP
- subroutine THOMAS_3D_i1_j0_OMP
- subroutine THOMAS_Z_NoOpenMP
- subroutine THOMAS_Z_NoOpenMP
- subroutine THOMAS_Z_OpenMP

Esencialmente el código sigue el Algoritmo 1 presentado anteriormente. Las distintas implementaciones tienen sutiles diferencias para los casos bidimensionales y tridimensionales, así como también existen para las implementaciones 3D versiones paralelizadas.

En el Anexo I se encuentran los códigos de todas estas subrutinas, sin embargo aquí solo analizará una de ellas, la subrutina THOMAS_2D. Los comentarios sobre esta subrutina aplican al resto y de hecho su contenido es la base de las demás. Luego se realizan algunos comentarios sobre las diferencias que el resto de las subrutinas presenta respecto a la 2D.

A continuación se muestra el código de la subrutina THOMAS_2D.

```
subroutine THOMAS_2D (IJmin, IJmax,JImin, JImax, di, dj,                    &
                      DCoef_2D, ECoef_2D, FCoef_2D, TiCoef_2D,      &
                                ANSWER, VECG, VECW)

    !Arguments----------------------------------------------------------------

    integer,                  intent(IN) :: IJmin, IJmax
    integer,                  intent(IN) :: JImin, JImax
    integer,                  intent(IN) :: di,    dj
    real,    dimension(:,:), pointer         :: DCoef_2D, FCoef_2D, TiCoef_2D
    real(8), dimension(:,:), pointer          :: ECoef_2D
    real,    dimension(:,:), pointer         :: ANSWER
    real(8), dimension(:  ), pointer          :: VECG, VECW

    !Local---------------------------------------------------------------------
    integer :: IJ, JI, II, MM, I, J
    !--------------------------------------------------------------------------

do2 :   do IJ = IJmin, IJmax

        I = IJ*dj + di

        J = IJ*di + dj

        VECW(JImin) = -FCoef_2D (I, J)/ECoef_2D(I, J)

        VECG(JImin) =  TiCoef_2D(I, J)/ECoef_2D(I, J)

do3 :      do JI=JImin+1,JImax+1

           I      = IJ*dj + JI*di

           J      = IJ*di + JI*dj

           VECW(JI) = -FCoef_2D(I,J) / (ECoef_2D(I,J) + DCoef_2D(I,J) * VECW(JI-1))

           VECG(JI) = (TiCoef_2D(I,J) - DCoef_2D(I,J) * VECG(JI-1))/   &

                  (ECoef_2D (I,J) + DCoef_2D(I,J) * VECW(JI-1))

        end do do3

        I = IJ * dj + (JImax+1) * di

        J = IJ * di + (JImax+1) * dj

        ANSWER(I, J) = VECG(JImax1)

do1 :      do II = JImin+1, JImax+1

           MM = JImax+2-II

           I  = IJ*dj + MM*di

           J  = IJ*di + MM*dj

           ANSWER(I,J) = VECW(MM) * ANSWER(I+di,J+dj) + VECG(MM)

        end do do1

    end do do2

  end subroutine THOMAS_2D
```

Figura 12: Código de la subrutina THOMAS_2D

Lo primero a destacar es la forma en que el MOHID almacena la matriz a resolver. Los sistemas que el modelo deber resolver tiene tantas ecuaciones como nodos, así si la malla tiene $IJ_{\max}$ elementos en la dirección x y $JI_{\max}$ elementos en la dirección y, se tendrá una matriz tridiagonal como la de la Figura 13 (a) siendo n igual a ($IJ_{\max} * JI_{\max}$). Sin embargo la forma en que el modelo almacena esta matriz es guardando cada una de las diagonales en una matriz con el tamaño de la malla ($IJ_{\max} * JI_{\max}$).

a)-
$$\begin{pmatrix} b_1 & c_1 & 0 & . & . & . & . & 0 \\ a_2 & b_2 & c_2 & 0 & . & . & . & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & . & . & 0 \\ & & & . & & & & \\ & & & . & & & & \\ & & & . & & & & \\ 0 & 0 & 0 & . & . & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & . & . & . & a_n & b_n \end{pmatrix}$$

b)-
$$\begin{pmatrix} 0 & a_2 & . & a_{IJ\max} \\ a_{IJ\max+1} & . & . & \\ & . & . & \\ & & & a_n \end{pmatrix} \begin{pmatrix} b_1 & b_2 & . & b_{IJ\max} \\ b_{IJ\max+1} & . & . & \\ & . & . & \\ & & & b_n \end{pmatrix} \begin{pmatrix} c_1 & c_2 & . & c_{IJ\max} \\ c_{IJ\max+1} & . & . & \\ & . & . & \\ & & c_{n-1} & 0 \end{pmatrix}$$

Figura 13: Esquema de almacenamiento de matrices del MOHID.

El Algoritmo 1 fue presentado con una nomenclatura distinta a la que se muestra en el código de la Figura 12, la equivalencia es la siguiente:

$$ANSWER = x \quad Ti = y$$
$$D = a \quad E = b \quad F = c$$
$$W = \gamma \quad G = \beta \quad n = IJ_{\max} * JI_{\max}$$

Teniendo en cuenta estas equivalencias se puede ver que el algoritmo es el mismo con la diferencia de los loops que realiza por la forma en que almacena las diagonales. Otra observación a realizar es el rol de los parámetros di y dj, estos parámetros valen uno o cero alternadamente entre sí y esto tiene que ver con la dirección que se resuelve implícitamente en el método ADI como se mostró en el capítulo anterior.

En el caso de las implementaciones 3D el algoritmo es el mismo solo que se almacenan las diagonales en matrices 3D con las dimensiones de la malla tal como para el caso 2D, y por esta razón en los códigos esencialmente se agrega un loop externo que recorre cada capa.

Las versiones paralelizadas utilizan OpenMP y básicamente paralelizan en ese último loop que se menciona en el párrafo anterior.

4.4. Implementación de modelo de prueba

Se realizaron simulaciones de prueba a los efectos de evaluar donde el modelo resuelve sistemas lineales, como la hace y que fracción del tiempo de la simulación representa su resolución.

Se instrumentó el código fuente de manera de poder identificar la ruta dentro del mismo que conduce a la ejecución de las subrutinas que resuelven sistemas lineales. En este punto se describe la implementación del modelo utilizada para las pruebas mencionadas.

Esta implementación está siendo utilizada en el marco de desarrollo de un modelo de pronóstico de niveles para el Río de la Plata y su frente marítimo.

Para la aplicación del MOHID a un área determinada se deben introducir una serie de datos iniciales y de contorno particulares del área que se pretende simular. El primer paso para realizar una simulación es generar una malla de cálculo y conocer la batimetría de la región delimitada por la malla. Otros datos necesarios para realizar una simulación son las variables que se deben especificar en las fronteras, como por ejemplo la elevación de la superficie libre en los bordes abiertos, el caudal de ingreso en las secciones de aporte fluvial o el coeficiente de rozamiento del fondo.

Dado que no se justifica realizar una modelación hidrodinámica tridimensional baroclínica para simular el efecto de la marea a escala del Atlántico sur se utilizó un esquema simplificado del modelo hidrodinámico, bidimensional y barotrópico en el cual no se consideran las variaciones en la dirección vertical ni los gradientes de salinidad y temperatura.

4.4.1. Modelo padre

4.4.1.1. Malla de cálculo

La malla horizontal se estructuró bajo el sistema de coordenadas latitud-longitud y se extiende desde -60º latitud al sur, -20º longitud al este, -22º latitud al norte y -70º latitud al oeste. El límite oceánico sur se corresponde con un punto ubicado aproximadamente a 733 km al sur de Tierra del Fuego y la frontera norte se ubica aproximadamente a 166 km al norte de San Pablo. La frontera oceánica este se extiende aproximadamente 2.320 km desde el límite costero norte. Esta malla posee 150.131 nodos activos y presenta un paso espacial constante igual a 0.1º (11,1 km). En la Figura 14 se presenta una imagen global de la malla de cálculo con líneas de división definidas cada 5 celdas para mejorar la visualización de la malla.

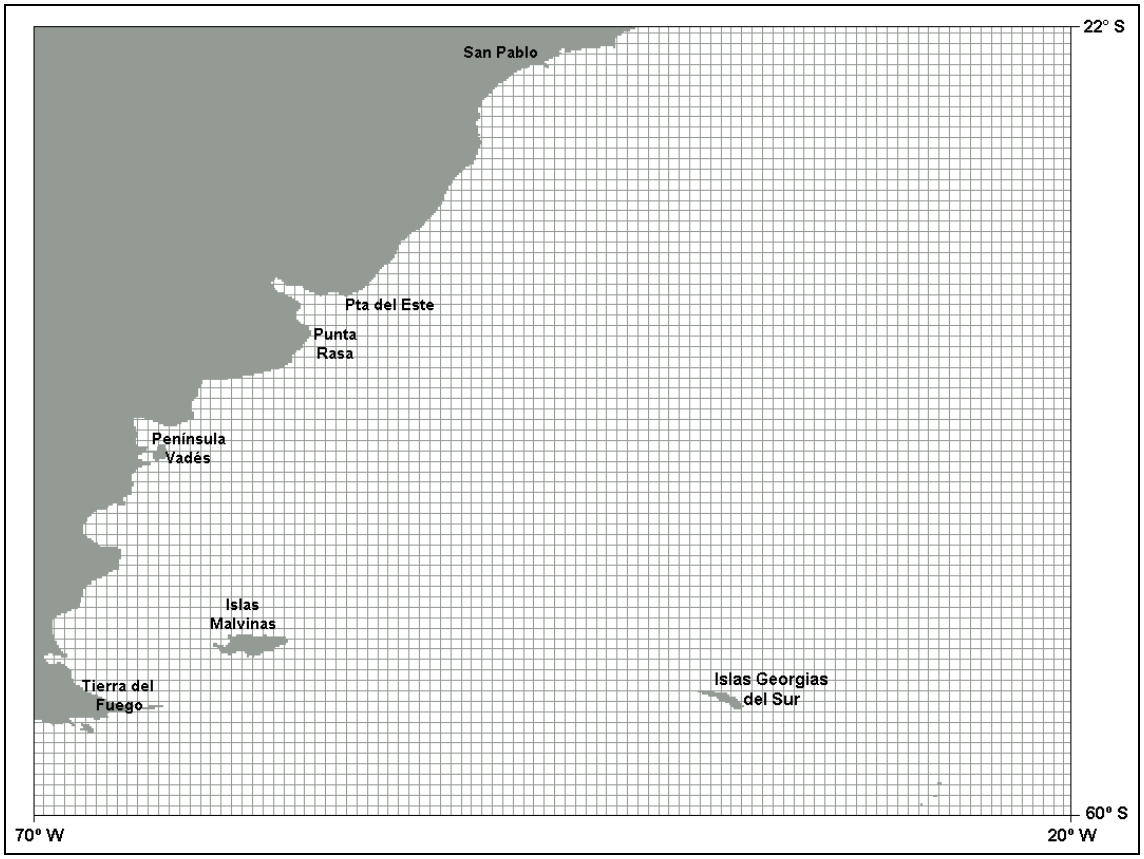

Figura 14: Malla de cálculo con líneas de división definidas cada 5 celdas.

Una vez definida la grilla horizontal, se seleccionó la coordenada vertical tipo sigma para discretizar el dominio en la dirección vertical, la cual permite una mejor resolución en un dominio con importantes variaciones topográficas. La profundidad con esta coordenada varía entre 0 y 1, siendo igual a 1 en la superficie libre e igual a 0 en el fondo. En todas las simulaciones realizadas con marea astronómica se utilizó una única capa sigma ya que como fue mencionado anteriormente la representación de la marea astronómica a escala del Atlántico sur no justifica una modelación tridimensional.

4.4.1.2. Batimetría y línea de costa

Luego de estructurar la malla de cálculo se debe conocer la batimetría de la región y la línea de costa. Para representar la batimetría del área de interés se utilizaron datos batimétricos digitalizados extraídos del General Bathymetric Chart of the Ocean (GEBCO) para todo el dominio con una resolución de 0.06º y una serie de datos de batimetría provistos por el Servicio de Hidrografía Naval de la Argentina para la zona del Río de la Plata con una mayor resolución (Dragani, 2002.

Para representar la línea de costa de la región de estudio se utilizaron datos extraídos del NOAA/NGDC Marine Geology and Geophysics Division.

Una vez ingresados los datos batimétricos y la línea de costa, se genera una matriz bidimensional con los valores de profundidad de cada una de las celdas de cálculo con ayuda de la herramienta Create Digital Terrain del MOHID GIS. Para

calcular la profundidad de una determinada celda, esta herramienta realiza un promedio de los datos de batimetría de los puntos ubicados dentro de la misma o en el caso de que la celda no contenga datos de batimetría realiza una interpolación triangular utilizando los valores de profundidad de las celdas adyacentes. En nuestro caso, dado que la resolución de los datos batimétricos es mayor a la resolución de la malla, no fue necesario utilizar la interpolación triangular. Luego de obtener la matriz de profundidades con el Create Digital Terrain, se aplicó a la misma un filtro de forma de suavizar las diferencias de profundidad entre celdas adyacentes. Finalmente, en la zona costera, algunos de los valores de la matriz de profundidades fueron corregidos manualmente con el objetivo de eliminar profundidades muy pequeñas y morfologías costeras muy complejas, como ser las lagunas costeras. La aplicación del filtro y la posterior corrección manual de la matriz de profundidades son procedimientos recomendados para evitar la ocurrencia de inestabilidades en el algoritmo de resolución del modelo y que además no modifican la calidad de representación del flujo en este primer nivel de resolución.

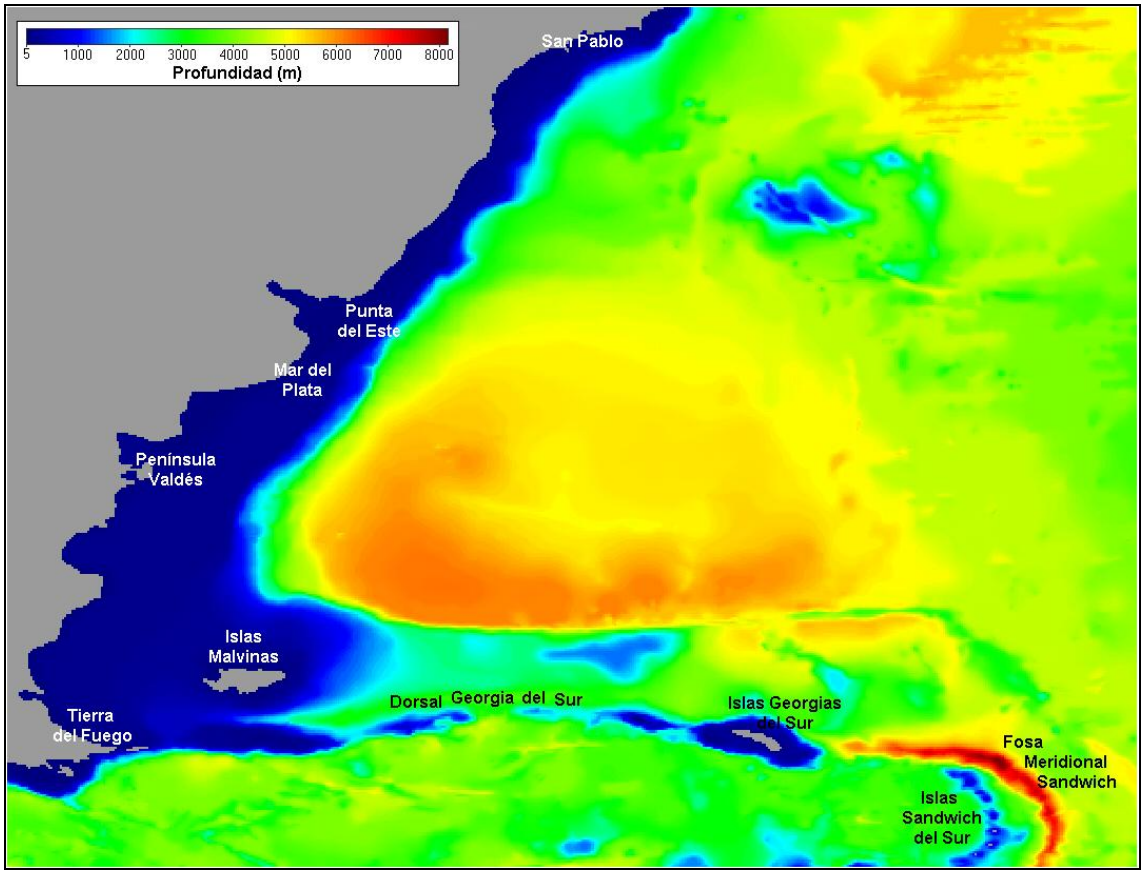

Figura 15: Batimetría del dominio de cálculo.

En la Figura 15 se presenta la batimetría de todo el dominio de cálculo ingresada al modelo en la cual puede observarse la correcta representación de las zonas costeras así como de las unidades morfológicas más importantes de la zona, como ser la dorsal Georgia del Sur y la fosa meridional Sándwich en la cual se alcanza la mayor profundidad de todo el dominio (8.015 m).

4.4.1.3. Condiciones iniciales y de borde

El sistema de ecuaciones en que se basa el modelo para el cálculo de la elevación de la superficie libre y de la velocidad es de naturaleza parabólica, por lo que son necesarias condiciones iniciales en todo el dominio y condiciones de borde durante toda la simulación para resolver el sistema.

Condiciones de borde

Elevación de la superficie libre

En la frontera oceánica del dominio de cálculo se debe especificar la elevación de la superficie libre. La elevación de la superficie libre puede imponerse como un valor constante en todo el período de simulación o puede especificarse como una serie temporal de nivel variable.

En las simulaciones realizadas para este trabajo, la condición de borde en la frontera oceánica fue utilizada para imponer el efecto de la marea astronómica sobre el dominio de cálculo. Para imponer el efecto de la marea astronómica se calcula la elevación de la superficie libre en 63 nodos, distribuidos en toda la frontera oceánica, a partir de la superposición de 13 componentes armónicas de la marea: M2, N2, S2, K2, 2N2, O1, Q1, K1, P1, Mf, Mm, Mtm, MSqm. En la Figura 16 se muestra la distribución de estos 63 nodos en la frontera oceánica: 2 ubicados en la frontera oeste, 22 en la frontera sur, 28 en la frontera este y 11 en la frontera norte.

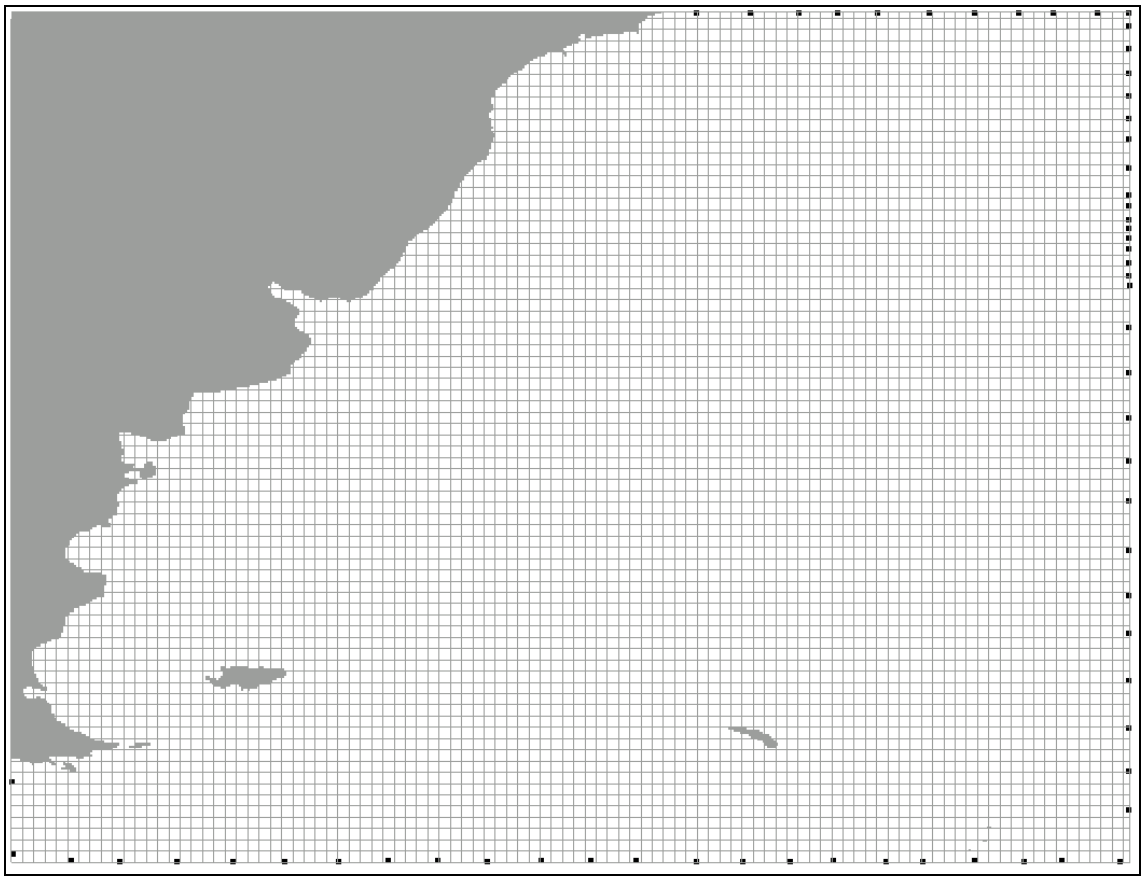

Figura 16: Ubicación de los 63 nodos de ingreso de la marea astronómica en la frontera oceánica.

Los valores de las amplitudes y fases de las componentes de la marea astronómica que se ingresan en los 63 nodos de la frontera oceánica se obtienen con ayuda de una herramienta del MOHID denominada MOHID-tide, la cual extrae la solución del atlas de mareas FES2004 en dichos puntos de la frontera. Estos valores son los que deben ajustarse en la calibración del modelo para representar correctamente la onda de marea en el dominio.

A partir de los valores de nivel calculados en los 63 nodos de la frontera oceánica, el modelo interpola linealmente para cada paso de tiempo calculando así el nivel en los nodos restantes de dicha frontera.

Caudal fluvial

La frontera oeste del dominio de cálculo incluye una sección de aporte de caudal fluvial de los ríos Paraná y Uruguay. En el modelo MOHID dicho aporte es representado como una condición de borde correspondiente a un caudal de entrada determinado. Este ingreso de caudal se divide en dos secciones diferentes, una correspondiente al ingreso del flujo de los ríos Paraná – Guazú y Uruguay y otra sección por donde ingresa el caudal del río Paraná – Las Palmas.

El ingreso de caudal puede especificarse como un valor constante a lo largo del período de simulación o como una serie temporal variable a lo largo del tiempo. En las simulaciones realizadas para este trabajo se adoptó un caudal de ingreso constante en cada una de las secciones de aporte correspondiente al caudal medio anual de los ríos Paraná – Guazú, Uruguay y Paraná – Las Palmas. En la Tabla 1 se presentan los caudales fluviales de las tres secciones de aporte.

Río	**Q (m^3/s)**
Paraná – Guazú y Uruguay	20.000
Paraná - Las Palmas	5.000

Tabla 1: Caudales de ingreso en las secciones de aporte de caudal fluvial.

Condiciones iniciales

Además de las condiciones de borde, el modelo requiere la especificación de una condición inicial para las variables del sistema que son el nivel de la superficie libre y la velocidad.

Las condiciones iniciales utilizadas son del tipo de Dirischlet, y se imponen especificando directamente los valores de las variables en todos los puntos del dominio durante el instante inicial. Para la elevación de la superficie libre se utiliza un valor de referencia uniforme en todo el dominio de cálculo, igual a 0,91 m mientras que para el campo de velocidades la condición inicial es el reposo y solamente es necesario un tiempo corto de arranque para que las velocidades se ajusten a la acción de los forzantes considerados.

4.4.2. Modelo hijo

4.4.2.1. Malla de cálculo

El dominio utilizado en el segundo nivel de resolución definido para este trabajo es de aproximadamente 262.263 km^2 y se presenta en la Figura 17, marcado sobre el dominio del modelo global.

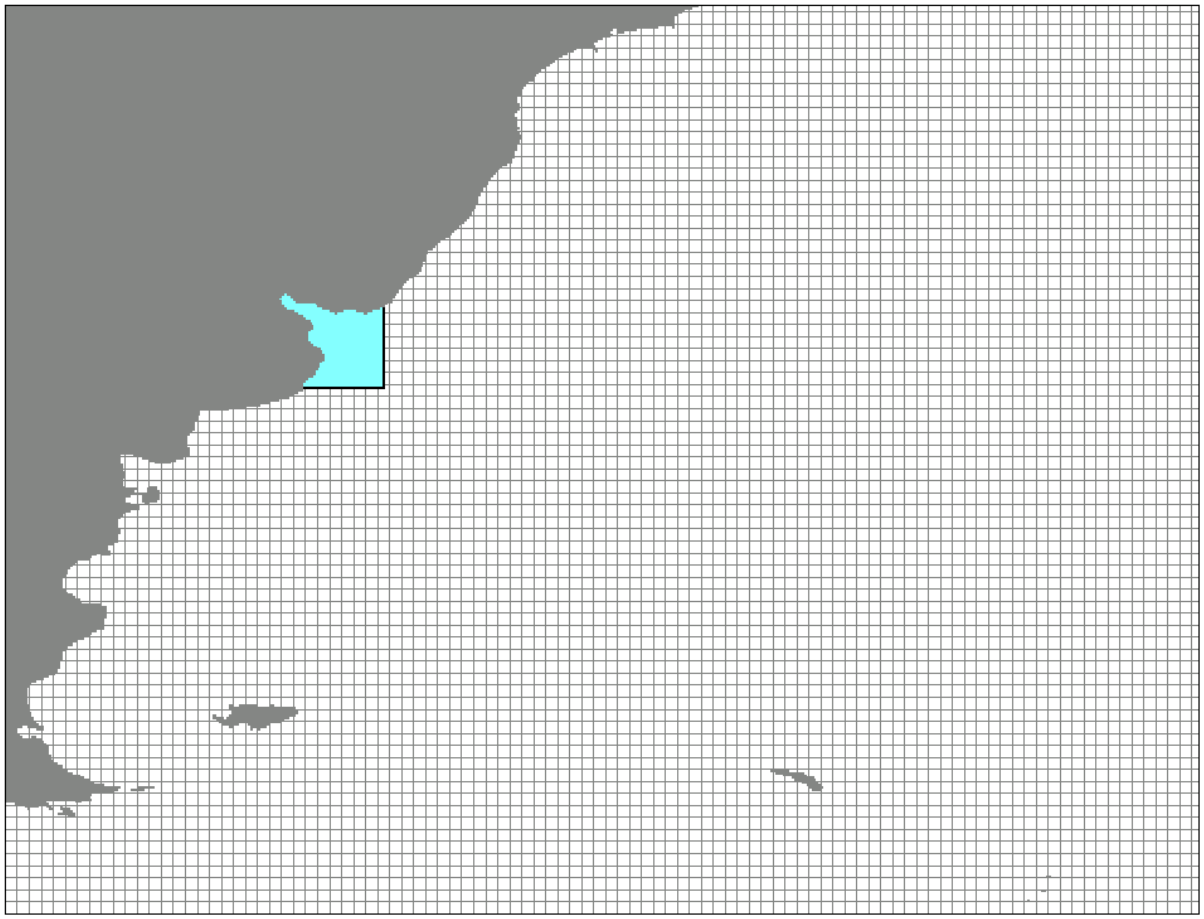

Figura 17: Dominio de cálculo simulado en el nivel intermedio.

La malla horizontal utilizada en el segundo nivel de resolución definido para este trabajo de modelación se estructuró bajo el sistema de coordenadas latitud-longitud y se extiende desde -38,09º latitud al sur, -54,10º longitud al este, -33,76º latitud al norte y -59,00º latitud al oeste. El límite sur interseca la costa argentina en Mar del Plata y se extiende aproximadamente 382 km al este de dicha ciudad. La frontera este pasa por La Paloma en la costa uruguaya y se extiende aproximadamente 388 km al sur de dicho balneario. La malla de cálculo horizontal utilizada para simular este nivel intermedio está formada por 9.600 celdas activas y presenta un paso espacial constante igual a 0.033º (3,7 km), un tercio del paso espacial de la malla definida para implementar el modelo global. En la Figura 18 se presenta una imagen de la malla de cálculo utilizada en el nivel intermedio.

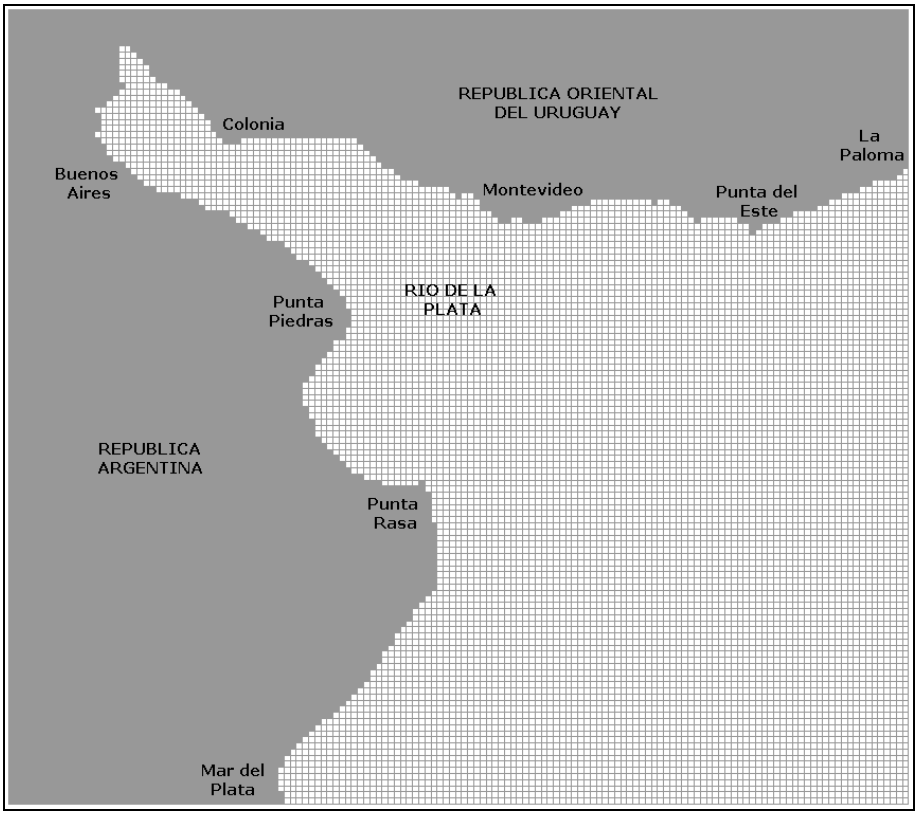

Figura 18: Malla de cálculo con líneas del modelo intermedio.

Para discretizar el dominio en la dirección vertical se seleccionó la coordenada vertical tipo sigma, la cual permite una mejor resolución en un dominio con importantes variaciones topográficas. La profundidad con esta coordenada varía entre 0 y 1, siendo igual a 1 en la superficie libre e igual a 0 en el fondo. En todas las simulaciones realizadas se utilizó una única capa sigma.

4.4.2.2. Batimetría y línea de costa

Se utilizaron las mismas fuentes de información que para el modelo padre. En la Figura 19 se presenta la batimetría ingresada al modelo intermedio.

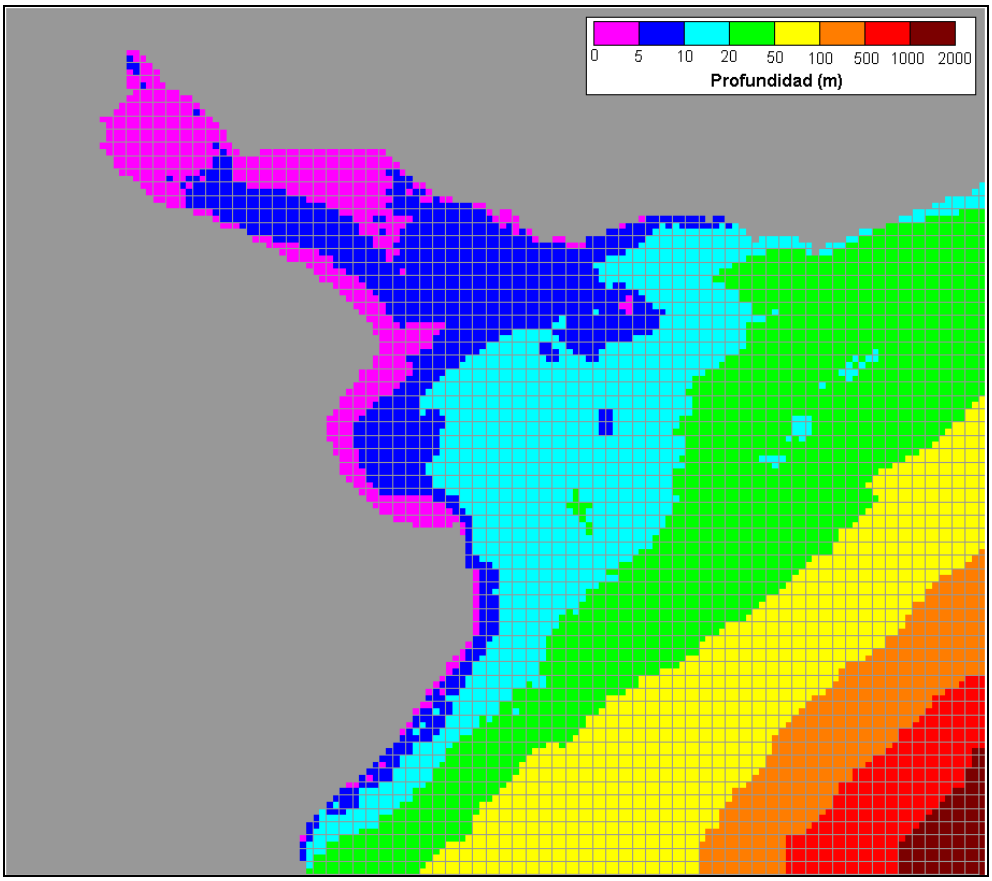

Figura 19: Batimetría del dominio de cálculo.

4.4.2.3. Condiciones iniciales y de borde

Condiciones de borde

En la frontera abierta del dominio de cálculo se debe especificar la elevación de la superficie libre. El modelo intermedio trabaja de forma acoplada con el modelo global y recibe la información calculada en dicho nivel a través de los bordes del nivel intermedio. De esta forma, la elevación de la superficie libre y la velocidad de flujo en la frontera abierta del modelo intermedio son los valores de nivel y velocidad de flujo calculados para cada instante de tiempo en las celdas del nivel global correspondientes a la frontera del nivel intermedio. Debido a que los tamaños de las celdas de las mallas del modelo global y del modelo intermedio son diferentes, el modelo realiza una interpolación lineal entre los valores calculados en las celdas del nivel global para determinar los valores en las celdas del nivel intermedio.

Para simular la marea astronómica en el modelo intermedio, se ingresa como condición de borde la solución del modelo global obtenida al considerar como único forzante a la marea astronómica. Por otro lado, para simular la marea meteorológica en el modelo intermedio, se utiliza la solución del modelo global

obtenida al considerar como forzantes a la marea astronómica, al viento y a la presión atmosférica.

Condición de radiación

A la condición de borde recibida del modelo global se agrega una condición de radiación de forma de permitir que las ondas salientes atraviesen la frontera oceánica del dominio sin reflejarse. En todas las simulaciones realizadas con el modelo intermedio se utilizó la condición de radiación de Flather, que consiste en la combinación de la condición de Sommerfeld y la ecuación de continuidad:

$$v = v_r(t) \pm \frac{c}{H}\left(\eta - \eta_r(t)\right) \quad \textbf{Ecuación 121}$$

Done η es el nivel de la superficie libre calculado con el modelo, η_r es el nivel de la superficie libre impuesto por la solución de referencia, c es la velocidad de fase de la onda, H es la profundidad de la columna de agua y v_r es la velocidad de flujo normal a la frontera impuesta por la solución de referencia. La solución de referencia utilizada para implementar esta condición de radiación fue la obtenida con el modelo global en las celdas correspondientes a la frontera abierta del modelo intermedio.

Caudal fluvial

El mayor aporte de caudal fluvial que recibe el Río de la Plata proviene de los ríos Paraná y Uruguay. En el modelo MOHID dicho aporte es representado como una condición de borde correspondiente a un caudal de entrada determinado. En el nivel intermedio, este ingreso de caudal se divide en tres secciones, dos correspondientes al ingreso del flujo de los ríos Paraná – Guazú y Uruguay y otra sección por donde ingresa el caudal del río Paraná – Las Palmas.

El ingreso de caudal puede especificarse como un valor constante a lo largo del período de simulación o como una serie temporal variable a lo largo del tiempo. En las simulaciones realizadas para este trabajo se adoptó un caudal de ingreso constante en cada una de las secciones de aporte correspondiente al caudal medio anual de los ríos Paraná – Guazú, Uruguay y Paraná – Las Palmas. En la Tabla 2 se presentan los caudales fluviales de las tres secciones de aporte.

Río	**Q (m^3/s)**
Paraná – Guazú y Uruguay (1)	10.000
Paraná – Guazú y Uruguay (2)	10.000
Paraná - Las Palmas	5.000

Tabla 2: Caudales de ingreso en las secciones de aporte de caudal fluvial.

Superficie libre

Para simular la marea meteorológica en el modelo intermedio se agrega además la acción de los forzantes atmosféricos sobre el dominio de estudio. Para ello, se introduce el campo de vientos a 10 m y el campo de presión atmosférica

obtenido con el WRF a través de la condición de borde en la superficie libre. Se utiliza el mismo archivo de vientos y presiones ingresado al modelo global para simular la marea meteorológica (en formato HDF5) interpolado a la malla de cálculo del modelo intermedio con ayuda de la herramienta ConvertToHDF5.

Condiciones iniciales

Además de las condiciones de borde, el modelo requiere la especificación de una condición inicial para las variables del sistema que son el nivel de la superficie libre y la velocidad.

Las condiciones iniciales utilizadas son del tipo de Dirischlet, y se imponen especificando directamente los valores de las variables en todos los puntos del dominio durante el instante inicial. Para la elevación de la superficie libre se utiliza un valor de referencia uniforme en todo el dominio de cálculo, igual a 0,91 m mientras que para el campo de velocidades la condición inicial es el reposo y solamente es necesario un tiempo corto de arranque para que las velocidades se ajusten a la acción de los forzantes considerados. Para la salinidad, la condición inicial es un campo uniforme con el valor de salinidad oceánica considerada en este trabajo de modelación, igual a 38 ppm.

4.4.3. Período a simular

Para la realización de estas pruebas únicamente se simularon 6 horas, utilizando 60 segundos como paso de tiempo para el modelo padre y 20 segundos para el modelo hijo.

4.4.4. Características de la plataforma de ejecución

Las corridas realizadas se llevaron a cabo en el CLUSTER FING, más precisamente en uno de sus nodos con las siguientes caracteríticas: HP Proliant DL180 G6, dos procesadores Quad Core Intel Xeon serie E5520, 2.26 GHz., memoria RAM 24 GB, DDR3 1066 MHz., 2 puertos Gigabit Ethernet, almacenamiento local 60 GB SATA2.

4.5. Resultados

Con las características de la implementación descrita anteriormente el modelo únicamente utiliza el algoritmo de Thomas para la resolución de sistemas lineales (otras implementaciones podrían llegar a utilizar otros métodos).

4.5.1. Ruta de ejecución

A continuación se muestra cómo las subrutinas que implementan el algoritmo de Thomas son invocadas. Partiendo del código Main (base del MOHIDWater) se muestran en verdes las subrutinas que son invocadas y terminan en la ejecución de los algoritmos de Thomas.

Main (Main.F90)

```
...
if (RunInParallel) then
        call ConstructMohidWaterMPI
else
        call ConstructMohidWater
endif

        call ModifyMohidWater

if (RunInParallel) then
        call  KillMohidWaterMPI
else
        call KillMohidWater
endif
...
```

Subroutine ModifyMohidWater (Main.F90)

```
...
call SearchMinMaxTimeStep
do while (Running)
        GlobarlCurrentTime = GlobalCurrentTime + Dtmin
        if (RunInParallel) then
            do while (associated(CurrentModel))
                if (CurrentModel%MPI_ID == myMPI_ID) then
                    call UpdateTimeandMapping
                    if (DoNextStep) then
                        if (associated(CurrentModel%FatherModel)) then
                            if (CurrentModel%CurrentTime == CurrentModel%InfoTime) then
                                call ReceiveInformationMPI (CurrentModel)
                            else if (CurrentModel%CurrentTime < CurrentModel%InfoTime) then
                                call UpdateSubModelValues (CurrentModel)
                            else if (CurrentModel%CurrentTime > CurrentModel%InfoTime) then
                                stop 'ModifyMohidWater - MohidWater - ERR02a'
                            endif
                        endif
                        call RunModel
                        call SendInformationMPI (CurrentModel)
                    endif
                endif
                CurrentModel => CurrentModel%Next
            enddo
        else
            CurrentModel => FirstModel
            do while (associated(CurrentModel))
                call UpdateTimeAndMapping
                if (DoNextStep) then
                    call SubModelComunication (CurrentModel)
                    call RunModel
                endif
                CurrentModel => CurrentModel%Next
            enddo
        endif
        call SearchMinMaxTimeStep (DTmin, DTmax)
        if (abs(GlobalCurrentTime - GlobalEndTime) > DTmin / 10.0) then
            Running = .true.
        else
            Running = .false.
        endif
enddo
```

Subroutine RunModel (ModuleModel.F90)

```
...
call Ready(ModelID, ready_)
if (ready_ .EQ. IDLE_ERR_) then
        #ifndef _AIR_
            call GetWaterPoints2D
            call ModifyAtmosphere
            call UnGetHorizontalMap
            call ModifyInterfaceWaterAir
        #endif
        #ifndef _WAVES_
            if(Me%RunWaves)then
                call ModifyWaves
            endif
        #endif
        call GetDensity
        call GetSigma
        if (Me%ExternalVar%NeedsTempSalinity) then
            call GetConcentration
        endif
        call GetHorizontalVelocity
        call GetVerticalVelocity
        call GetChezy
        call Turbulence
        call UnGetHydrodynamic
        call UnGetHydrodynamic
        call UnGetHydrodynamic
        PredictedDT = Me%DT
        call Modify_Hydrodynamic
        call UnGetWaterProperties
        call UnGetWaterProperties
        call UnGetWaterProperties
        call WaterProperties_Evolution
        #ifndef _SEDIMENT_
            if(Me%RunSediments)then
                call ModifyConsolidation
                call SedimentProperties_Evolution
            endif
        #endif
        #ifndef _LAGRANGIAN_
            if (Me%ObjLagrangian /= 0) then
                call ModifyLagrangian
            endif
```

Subroutine Modify_Hydrodynamic (Module_Hydrodinamic.

```
call GetComputeCurrentTime
call Actualises_Hydrodynamic
call ReadLock_External_Modules
call ReadLock_ModuleTurbulence
call One_Iteration
call ComputeBoxesWaterFluxes
call ComputeResidualFlowProperties
call ComputeSystemEnergy
call Hydrodynamic_OutPut
if (Me%ComputeOptions%Recording) then
        call ModifyHydrodynamicFile
endif
call GetVariableDT
if (VariableDT) then
          call CalcNewDT (NewDT)
endif
call ReadUnLock_ModuleTurbulence
...
```

Subroutine OneIteration (Module_Hydrodinamic.F90)

```
call Actualize_HydrodynamicTimeStep
if (FirstIteration) then
        call Bottom_Boundary
endif
Evolution = Me%ComputeOptions%Evolution
if (Evolution == Solve_Equations_) then
          call MomentumMassConservation
else if (Evolution == Read_File_) then
        call ReadHydrodynamicFile
        call Bottom_Boundary
else if (Evolution == No_hydrodynamic_
                .or. Evolution == Residual_hydrodynamic_) then
        call New_Geometry
        call Bottom_Boundary
else if (Evolution == ImposedSolution_) then
        call ReadImposedSolution
        call Modify_HorizontalWaterFlow
        call ChangeDirection
        call Modify_HorizontalWaterFlow
        call New_Geometry
        call Bottom_Boundary
else if (Evolution == Vertical1D_) then
        call AssociateDirectionX
        call Bottom_Boundary
        call Explicit_Forces
        call Compute_Velocity
        call AssociateDirectionY
        call Bottom_Boundary
        call Explicit_Forces
        call Compute_Velocity
else
        Stop 'Sub. One_Iteration - ModuleHydrodynamic - Err04'
endif
```

Subroutine MomentumMassConservation (Module_Hydrodinamic.

```
call ModifyWaterDischarges
if (Me%ComputeOptions%Vertical_AxiSymmetric_Model == 0) ther
        if     (Num_Discretization == Abbott    ) then
                call Abbott_Scheme
        else if (Num_Discretization == Leendertse) then
                call Leendertse_Scheme
        else
                Stop 'Sub. MomentumMassConservation
                        - ModuleHydrodynamic - Err01.'
        endif
else
        call Implicit_1DScheme
endif
if (Me%NonHydrostatic%ON) then
        call VerticalMomentum
        call NonHydroStaticCorrection
        call Modify_HorizontalWaterFlow
        call NullifyAuxiliarPointers
endif
Grid = Fix
call New_VerticalHydrodynamic( Grid)
call New_Geometry
Grid = Variable
call New_VerticalHydrodynamic( Grid)
```

F90)

Subroutine Abbott_Scheme

```
call ChangeDirection
cal Bottom_Boundary
call Explicit_Forces
call Compute_WaterLevel
call Compute_Velocity
Me%WaterFluxes%New_Old = 0.5
call Modify_HorizontalWaterFlow
```

n

Subroutine Leendertse_Scheme

```
call MaintainDirection
call Explicit_Forces
call Compute_Velocity
call Modify_HorizontalWaterFlow
call ChangeDirection
call Bottom_Boundary
call Explicit_Forces
call Compute_WaterLevel
call Compute_Velocity
Me%WaterFluxes%New_Old = 1.
```

Subroutine Compute WaterLevel (Module_Hydrodinamic.F90)

```
...
Inicia variables (Dcoef_2D=Fcoef_2D=0, Ecoef_2D=1
                  , TiCoef_2D=WaterLevel_Old)
call WaterLevel_BottomFriction
call WaterLevel_BarotropicPressure
call WaterLevel_ExplicitForces
call WaterLevel_WaterFluxes
call WaterLevel_OpenBoundary
call WaterLevelDischarges
if (.not. CyclicBoundary ON)
        call THOMAS_2D
else
        call Water_Level_CyclicBoudary
endif
if (Me%Relaxation%WaterLevel)
        call WaterLevelRelaxationAltimetry
nullify (DCoef_2D, ECoef_2D, FCoef_2D, TiCoef_2D)
nullify (WaterLevel_Old, WaterLevel_New)
```

Subroutine Compute_Velocity (Module_Hydrodinamic.F90)

```
...
Incia variables
call Velocity_ExplicitForces
call VelVerticalDiffusionBoundaries
if (KUB > 1) then
        if (Me%ComputeOptions%VerticalDiffusion)
                call Velocity_VerticalDiffusion
        if (Me%ComputeOptions%VerticalAdvection)
                call Velocity_VerticalAdvection
        endif
endif
if (Me%ComputeOptions%WaveStress) then
        call Velocity_WaveStress
endif
do  k = KLB, KUB
    do  j = JLB, JUB
        do  i = ILB, IUB
            Modifica el TiCoef_3D, correcciones en la frontera,...
        enddo
    enddo
enddo
if (KUB == 1)
        call THOMAS_3D
else
        call THOMASZ
endif
call InstantMixingSmallDepths
call Velocity_OpenBoundary
if (Me%CyclicBoundary%ON)  then
        .
        .
        .
endif
if (Me%Relaxation%Velocity)
        call VelocityRelaxation
nullify (DCoef_3D, ECoef_3D, FCoef_3D, TiCoef_3D)
nullify (WaterPoints3D, LandBoundaryFacesUV)
nullify (Velocity_UV_Old, Velocity_UV_New)
```

El esquema de la Figura 20 muestra en forma resumida la ruta de ejecución hasta invocar los algoritmos encargados de la resolución de los sistemas lineales. Vale la pena destacar que la subrutina ComputeVelocity en esta versión del código (correspondiente a Junio de 2007) paradójicamente cuando la implementación del modelo es 2D ejecuta la subrutina THOMAS_3D, mientras que en implementaciones 3D llama a la subrutina THOMAS_2D.

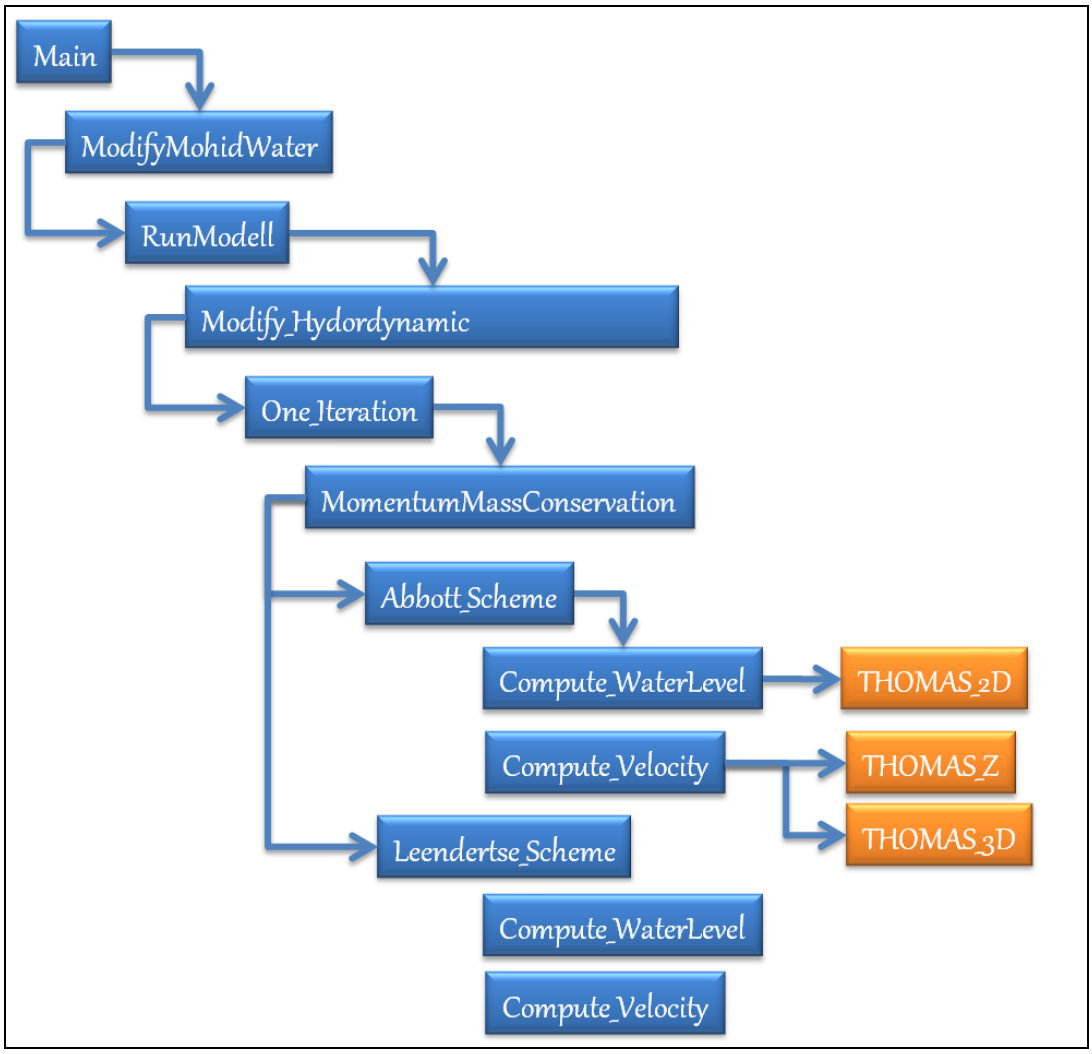

Figura 20: Esquema de la ruta de ejecución hasta alcanzar los algoritmos de resolución de sistemas lineales.

4.5.2. Tiempos de ejecución

Se evaluaron los tiempos de ejecución de las rutinas y subrutinas descritas en el punto anterior. Para ello fueron introducidas en el código fuente sentencias basadas en el comando MCLOCK. La precisión obtenida con este comando es de 1 milisegundo.

En la Figura 21 se presenta un resumen de los tiempos de ejecución evaluados. Como se mencionó anteriormente se simuló un período de seis horas, con un DT de 60s para el modelo padre y 20 s para el modelo hijo, lo cual implica la realización de 360 y 1080 iteraciones respectivamente. La simulación se llevó a cabo en 493,5 seg, (poco más de 8 minutos).

Los tiempos que se presentan en la Figura 21 son lo correspondientes a las subrutinas que fueran identificadas en el punto anterior (Figura 20). Se presentan los tiempos totales en segundos, y luego el tiempo en milisegundos consumido en cada invocación de la subrutina.

Corrida con implementación PDT

Período simulado:	01/05/2007 00:00 - 01/05/2007 06:00 --> 6 horas			
DT modelo padre:	60 segundos	---->	Nº de Iteraciones	360
DT modelo hijo:	20 segundos	---->	Nº de Iteraciones	1080

Tiempo total de ejecución (seg.)	493,5

		Tiempo total (s)	Tiempo por iteración (ms)
Modelo Padre	Modify_Hydrodynamic	239,7	665,7
	One_Iteration	218	605,5
	MomentumMassConservation	217,9	605,4
	Leendertse_Scheme	192,5	534,6
	Compute_Velocity	24,33	33,8
	Compute Water Level	7,92	22
	Compute_Velocity	24,33	33,8

		Tiempo total (s)	Tiempo por iteración (ms)
Modelo Hijo	Modify_Hydrodynamic	39,42	36,50
	One_Iteration	34,82	32,24
	MomentumMassConservation	34,19	31,66
	Leendertse_Scheme	27,83	25,77
	Compute_Velocity	4,62	2,14
	Compute Water Level	8,24	7,63

Tiempo total (s)	**279,12**	(Suma de ambos
Porcentaje del tiempo de corrida asociado (%):	**56,6**	Modify_Hydrodynamic)

Figura 21: Resumen de tiempos de ejecución evaluados en la simulación de prueba.

Se puede ver que las subrutinas Modify_Hydrodynamic insumen casi el 57% del tiempo total requerido para llevar a cabo la simulación. Dentro de ésta subrutina la subrutina One_Iteration consume la mayor parte del tiempo. Luego se invoca la subrutina MomentumMassConservation la cual es responsable de casi la totalidad del consumo de tiempo de la subrutina One_Iteration. Dentro de la subrutina MomentumMassConservation la invocación de la subrutina Leendertse_Scheme consume una porción importante del tiempo. Ésta última invoca las subrutinas Compute_Velocity y Compute_WaterLevel (dos veces la primera) las cuales finalmente ejecutan los algoritmos de Thomas. Sin embargo, en la subrutina Leendertse_Scheme la invocación del los algoritmos mencionados representa solo un 30% del tiempo de su ejecución. Por esta razón si se observa el porcentaje de tiempo que insume la resolución de los sistemas lineales respecto al tiempo total de simulación (utilizando la suma de los tiempos de las subrutinas Compute_Velocity y Compute_WaterLevel del modelo padre e hijo como indicador), se puede ver que representa un 15 % del tiempo total.

La Figura 22 resume los tiempos de ejecución de cada subrutina antes mencionados.

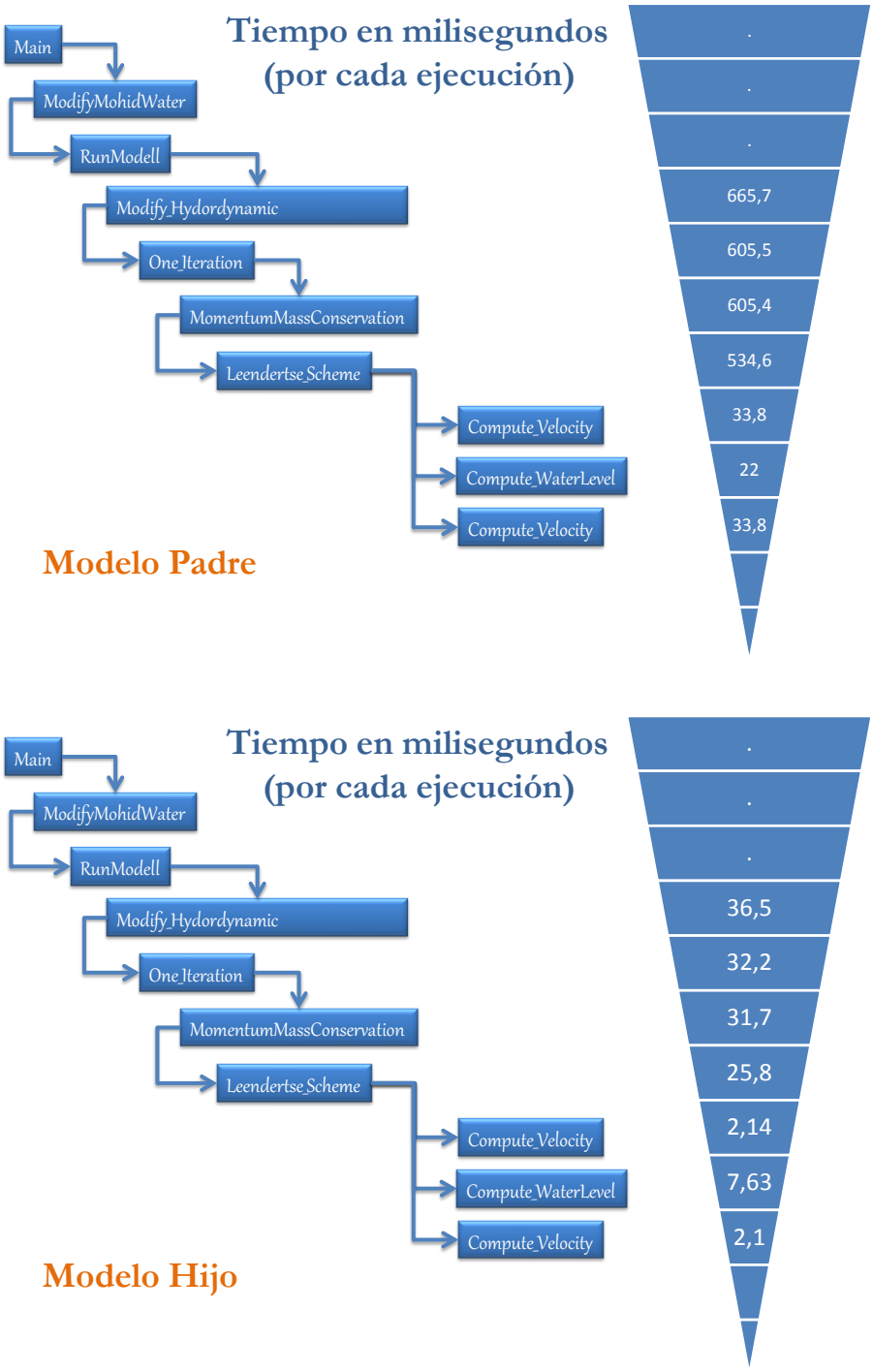

Figura 22: Resumen de tiempos insumidos por cada subrutina (mseg).

4.6. Sintesis

En este capítulo se han presentado una breve descripción teórica de los métodos implementados en el modelo MOHID para la resolución de sistemas lineales. Posteriormente el código fuente de la versión correspondiente a junio de 2007 fue instrumentado a los efectos de evaluar los tiempos de ejecución en una aplicación típica de las llevadas a cabo en el IMFIA.

La instrumentación del código permitió identificar claramente la ruta seguida por el modelo durante su ejecución hasta invocar a las rutinas encargadas de la resolución de sistemas lineales. Esta tarea permitió fortalecer el conocimiento y dominio que se tiene del código fuente del MOHID el cual presenta una complejidad importante.

A partir de la simulación de prueba se observa que la invocación a la subrutina Modify_Hydrodynamic insumen casi el 57% del tiempo total requerido para llevar a cabo la simulación. Dentro de esta subrutina la invocación de las subrutinas Compute_Velocity y Compute_WaterLevel, las cuales ejecutan los algoritmos de Thomas, representa aproximadamente un 25% del tiempo.

Vale la pena destacar que dicha versión del código fuente varias secciones paralelizadas con OpenMP no habían sido terminadas de implementar, incluyendo entre éstas el algoritmo de Thomas. El siguiente capítulo se centra en la evaluación del rendimiento empleando paralelización para lo cual se configuró y compiló una nueva versión del código del MOHID Water correspondiente a mayo 2011.

5. Paralelización del modelo MOHID

Las diversas aplicaciones del MOHID realizadas en el IMFIA para resolver diversos aspectos del flujo en el Río de la Plata y zonas adyacentes requieren de una gran capacidad de cálculo para resolver los sistemas de ecuaciones resultantes. En consecuencia, siguiendo el camino de otros usuarios de este tipo de modelos, el presente capítulo pretende trabajar en la implementación de versiones paralelas del código, posibilitando su ejecución en plataformas paralelas como ser el cluster FING.

La necesidad histórica de reducir el tiempo de cálculo de los modelos numéricos se convirtió en una prioridad para el equipo de desarrollo del MOHID con la implementación de un modelo operacional capaz de pronosticar la hidrodinámica y la calidad del agua de la desembocadura del Tajo, en Lisboa, Portugal. Es así que en el año 2003 se comenzó a implementar el procesamiento en paralelo en el MOHID Water.

Si bien el código del MOHID ya incluye secciones paralelizadas con OpenMP, la compilación de este modelo con OpenMP no está siendo utilizada actualmente en el IMFIA. Esto se debe a que una evaluación preliminar del ejecutable con OpenMP detectó mejoras marginales en el tiempo de ejecución con respecto a la versión serial del modelo.

5.1. Paralelización con OpenMP

El procesamiento en paralelo usando OpenMP se está aplicando actualmente en el MOHID mediante la definición de directivas para optimizar los loops. Estas directivas se definen como comentarios en el código y por lo tanto son leídas solo si se usan opciones especiales de compilación.

El procesamiento paralelo con OpenMP se puede utilizar en procesadores con múltiples núcleos presentes en el mismo equipo. No se puede utilizar para procesamiento en paralelo procesadores instalados en varios equipos dispuestos en un clúster.

Los loops de optimización se introdujeron en una primera fase en loops referidos a variables de la grilla (índices k, j, i), ubicados en la sección “Modifier” de los módulos del MOHID. Estos loops se utilizan varias veces e involucran una gran asignación de recursos en las simulaciones del MOHID, por lo tanto, estos son los lugares con mayores ganancias potenciales de los recursos involucrados en la paralelización.

5.2. Identificación de los procesos paralelizados con OpenMP en el código del MOHID

En las siguientes tablas se presentan las rutinas del código del MOHID Water que contienen alguna sección paralelizada con OpenMP (sombreadas en verde). Esta información se obtuvo buscando “!$OMP” (presente en el comienzo de toda directiva de OpenMP) en el código de los distintos módulos que intervienen en la ejecución del MOHID Water.

Subrayadas se presentan aquellas rutinas cuyo tiempo de cálculo queda registrado en el archivo Outwatch.dat. Estos tiempos son calculados con las rutinas StartWatch y StopWatch definidas en el módulo StopWatch. Las mismas utilizan los comandos CPU_TIME y DATE_AND_TIME de fortran para calcular el tiempo de cálculo consumido por cada rutina. Estos tiempos serán utilizados para comparar la performance de la ejecución en serie y en paralelo variando la cantidad de threads.

Main.F90	ModuleModel.F90

#ifdef _USE_MPI
RunInParallel = .true.
#else _USE_MPI
RunInParallel = .false.
#endif _USE_MPI
 	
!Subroutines———————————————
!Constructor
if (RunInParallel) then
ConstructMohidWaterMPI
GetSystemTime
ConstructModelList
AddNewModel
SetFilesName (Folder)
ConvertStrToInt
ConvertIntToStr
else
ConstructMohidWater
GetSystemTime
ConstructModelList
AddNewModel
SetFilesName (Folder)
ConstructModelOverlapping
endif
!Modifier
ModifyMohidWater
SearchMinMaxTimeStep
UpdateTimeStepsByMPI
DoOneTimeStep
SearchMinMaxTimeStep
if (RunInParallel) then
ReceiveInformationMPI
UpdateSubModelValues
SendInformationMPI
else
SubModelComunication
OverlapModelCommunication
endif
!Destructor
if (RunInParallel) then
KillMohidWaterMPI
GetSystemTime
else
KillMohidWater
GetSystemTime
endif
	!Subroutines———————————————
	!Constructor
	public :: ConstructModel
	#ifdef OVERLAP
	public :: ConstructOverlapping
	!Modifier
	public :: UpdateTimeAndMapping
	public :: RunModel
	RunOneModel
	!Destructor
	public :: KillModel
	
	!Selector
	public :: GetModelTimeLimits
	public :: GetModelTimeStep
	public :: GetModelCurrentTime
	public :: GetModelInstanceIDs
	public :: GetSubModelWindow
	public :: GetSubModelWindowON
	
	#ifdef OVERLAP
	public :: GetModelOverlap
	public :: GetModelOverlapInfo
	#endif OVERLAP
	
	!Management
	private :: AllocateInstance
	private :: DeallocateInstance
	private :: Read_Lock
	private :: Read_Unlock
	private :: Ready

Tabla 3: Rutinas del Main (izq) y del módulo Model (der).

ModuleHydrodynamic.F90

```
!Subroutines----------------------------------------
    Bottom_Boundary
    Modify_ChezyZ
    Modify_ChezyVelUV

    !Constructor
    StartHydrodynamic
      ...
    !Modifier
    Modify_Hydrodynamic
        Actualises_Hydrodynamic
        ...
        One_Iteration
            if Evolution == Solve_Equations
            call MomentumMassConservation
                if Num_Discretization == Abbott
                call Abbott_Scheme
                if Num_Discretization == Leendertse
                call Leendertse_Scheme
                    call ChangeDirection
                        call AssociateDirectionX
                        call AssociateDirectionY
                    call Bottom_Boundary
                    call Explicit_Forces
                        call Modify_Horizontal_Transport
                                Modify_Advection_UX_VY
                                Modify_Advection_UY_VX
                                Modify_Advection_Bound
                                Modify_UX_VY_Boundary
                                Modify_UY_VX_Boundary
                                Modify_UX_VY_SubModel
                                Modify_UY_VX_SubModel
                                ModifyRelaxHorizAdv
                                Modify_InertiaForces
                                Modify_ROX3
                                ModifyTidePotential
                                ModifyRelaxAceleration
                                ModifyAltimAceleration
                    call Compute_WaterLevel
                        WaterLevel_BarotropicPressure
                        WaterLevel_BottomFriction
                        WaterLevel_ExplicitForces
                        WaterLevel_WaterFluxes
                        WaterLevel_OpenBoundary
                                WaterLevel_ImposedWave
                                WaterLevel_FlatherWindWave
                                WaterLevel_FlatherLocalSolution
                                WaterLevel_BlumbergKantha
                        WaterLevelDischarges
                        WaterLevelRelaxation
                        WaterLevelRelaxationAltimetry
                        WaterLevelCorrection
                        WaterLevelMaxMin
                        Waterlevel_CyclicBoundary
                                CyclicCoef2D
                    call Compute_Velocity
                        Velocity_ExplicitForces
                        Velocity_VerticalAdvection
                        Velocity_VerticalDiffusion
                        VelVerticalDiffusionBoundaries
                        Velocity_OpenBoundary
                                Compute_BaroclinicHorVelocity
                                VelTangentialOpenBoundary
                                VelNormalOpenBoundary
                                VelSubModelNormalOB
                                VelSubModelTangentialOB
                                CyclicBoundVectTangential
                                CyclicBoundVectNormal
                                CyclicBoundVertical
                    call Modify_HorizontalWaterFlow
                call VerticalMomentum
                call NonHydroStaticCorrection
                call Modify_HorizontalWaterFlow
                call New_Geometry
                call New_VerticalHydrodynamic
                    Modify_VerticalWaterFlow
                    Filter_3D_Fluxes
                    Compute_VerticalVelocity
                    ComputeCartesianNH
                    ComputeCartesianVertVelocity_
                    Boundary_VerticalFlow
                        ComputeBaroclinicVertVelocity
                        Compute_BoundaryVertFlux
                    call ModifyWaterDischarges
            if Evolution == Read_File
            call ReadHydrodynamicFile
            call Bottom_Boundary
            if Evolution == ImposedSolution
            call ReadImposedSolution
            ...
        ComputeResidualFlowProperties
        ComputeSystemEnergy
        Hydrodynamic_OutPut
            Write_HDF5_Format
                CenterVelocity
            Write_Surface_HDF5_Format
            ModifyMatrixesOutput
                CenterVelocity
        ...
    !Destructor
    KillHydrodynamic
    ...
```

Tabla 4: Rutinas del módulo Hydrodynamic.

ModuleWaterProperties.F90	ModuleAdvectionDiffusion.F90
!Subroutines—————————————— !Constructor Construct_WaterProperties ... !Modifier WaterProperties_Evolution ModifyPropertiesFromFile HydroIntegration_Processes Advection_Diffusion_Processes WaterPropDischarges FreeVerticalMovements_Processes Bottom_Processes SetLimitsConcentration Filtration_Processes Reinitialize_Solution Surface_Processes MacroAlgae_Processes Partition_Processes InstantaneouslyMixing DataAssimilationProcesses OutPut_Results_HDF OutPut_TimeSeries Actualize_Time_Evolution ModifyDensity ModifySpecificHeat ModifySolarRadiation SetWaterPropFather ActualizeSubModelValues ActualizeSon3DFather2D ActualizeSon3DFather3D !Destructor KillWaterProperties	!Subroutines—————————————— !Constructor StartAdvectionDiffusion ... !Modifier AdvectionDiffusion Set_Internal_State Convert_Dif_Vertical AdvectionDiffusionIteration Convert_Visc_Dif_Horizontal VolumeVariation Discharges HorizontalDiffusion HorizontalDiffusionXX HorizontalDiffusionYY OpenBoundaryCondition FluxAtOpenBoundary HorizontalAdvection HorizontalAdvectionXX HorizontalAdvectionYY VerticalDiffusion VerticalAdvection FinishAdvectionDiffusionIt ... !Destructor KillAdvectionDiffusion ...

Tabla 5: Rutinas del módulo WaterProperties (izq) y del módulo AdvectionDiffusion (der).

ModuleTurbulence.F90	ModuleTurbGOTM.F90
!Subroutines----------	!Subroutines----------
!Constructor **ConstructTurbulence** ... TurbulentViscosity_CellCorner	**!Constructor** **StartTurbGOTM** ... **!Modifier** TurbGOTM **Read_Final_Turbulence_File** **Write_Final_Turbulence_File** **!Destructor** **KillTurbGOTM** ...
!Modifier Turbulence LeendertseeModel BackhausModel PacanowskiModel NihoulModel **TurbulenceEquationModel** Richardson ComputeMixedLayerDepth EstuaryModel SmagorinskyModel **OutPut_Results_HDF** Write_HDF5_Format ... **!Destructor** **KillTurbulence** ...	

Tabla 6: Rutinas del módulo Turbulence (izq) y del módulo TurbGOTM (der).

ModuleAtmosphere.F90	ModuleInterface.F90
!Subroutines----------	!Subroutines----------
!Constructor StartAtmosphere ... **!Modifier** ModifyAtmosphere ModifyPrecipitation ModifySolarRadiation ModifyCloudCover ModifyRandom **!Destructor** KillAtmosphere ...	**!Constructor** ConstructInterface ... **!Modifier** Modify_Interface FillMassTempSalinity InputData PropertyIndexNumber UnfoldMatrix ... **!Selector** GetRateFlux ... **!Destructor** KillInterface ...

Tabla 7: Rutinas del módulo Atmosphere (izq) y del módulo Interface (der).

ModuleInterfaceSedimentWater.F90	ModuleInterfaceWaterAir.F90
!Constructor StartInterfaceSedimentWater **!Modifier** ModifyInterfaceSedimentWater ModifyShearStress ComputeWaveRugosity ComputeWaveTension Benthos_Processes Detritus_Processes **ModifyWaterColumnFluxes** InitializeFluxesToWaterColumn **ModifyErosionFluxes** ModifyErosionCoefficient ModifyDepositionFluxes ModifyDissolvedFluxes **ModifySedimentColumnFluxes** InitializeFluxesToSediment ComputeConsolidation ModifyConsolidatedErosionFluxes **ModifySedimentWaterFluxes** ComputeWaterFlux DissolvedSedimentWaterFluxes ParticulateSedimentWaterFluxes Output_BoxTimeSeries ... **!Destructor** KillInterfaceSedimentWater ...	!Subroutines———————————————————— **!Constructor** StartInterfaceWaterAir ... **!Modifier** ModifyInterfaceWaterAir **ModifyRugosity** ComputeWavesRugosity **ModifyLocalAtmVariables** **ModifyWaterAirFluxes** **ModifyEvaporation** ComputeEvaporation ModifyCarbonDioxideFlux **ModifySurfaceRadiation** ComputeSurfaceRadiation ModifyWindShearVelocity **ModifyTurbulentKE** ComputeTKEWind Output_BoxTimeSeries ... **!Destructor** KillInterfaceWaterAir ...

Tabla 8: Rutinas del módulo InterfaceSedimentWater (izq) y del módulo InterfaceWaterAir (der)

ModuleFunctions.F90	ModuleGeometry.F90
!Matrix Operations SetMatrixValue !Linear tridiagonal sytems solvers THOMAS_2D THOMAS_3D THOMASZ	!Subroutines---------- !Constructor ConstructGeometry ... !Modifier ComputeInitialGeometry ComputeVerticalGeometry ComputeSZZ ... ComputeSigma ... ComputeZCellCenter ComputeDistances ComputeAreas ComputeVolumes StoreVolumeZOld !Destructor KillGeometry ...

Tabla 9: Rutinas del módulo Functions (izq) y del módulo Geometry (der).

ModuleHorizontalMap.F90	ModuleOpenBoundary.F90	ModuleFreeVerticalMovement.F90
!Subroutines---------- !Constructor ConstructHorizontalMap ... !Modifier UpdateComputeFaces2D UpdateOpenPoints2D !Destructor KillHorizontalMap ...	!Subroutines---------- !Constructor ConstructOpenBoundary AllocateInstance !Modifier Modify_OpenBoundary ImposeInvertBarometer ... !Destructor KillOpenBoundary DeallocateInstance	!Subroutines---------- !Constructor Construct_FreeVerticalMovement ... !Modifier Modify_FreeVerticalMovement FreeVerticalMovementIteration VerticalFreeConvection CalcVerticalFreeConvFlux Vertical_Velocity ... !Destructor Kill_FreeVerticalMovement ...

Tabla 10: Rutinas del módulo HorizontalMap (izq), OpenBoundary (medio) y FreeVerticalMovement (der).

5.3. Implementación del MOHID evaluada

Como se ha mencionado diversas aplicaciones del modelo MOHID se han utilizado para realizar diversos estudios de circulación en el Río de la Plata y zonas

adyacentes. En este estudio en particular se utiliza la implementación del modelo hidrodinámico tridimensional aplicado al Río de la Plata. A continuación se describen las principales características de la misma.

5.3.1. Modelo Tridimensional

Este modelo fue desarrollado por el IMFIA con el objetivo de representar el flujo en todo el Río de la Plata y gran parte de su Frente Marítimo (Fossati & Piedra-Cueva, 2006). Para esta aplicación el MOHID fue implementado en su versión tridimensional y baroclínica. En la Tabla 11 se presentan las principales características de implementación del modelo tridimensional utilizadas para simular marea astronómica y marea meteorológica. En la Figura 23 y Figura 24 se presenta la malla de cálculo y la batimetría ingresada al dominio de este modelo.

Malla de cálculo	
Discretización horizontal	5.000 – 9.000 x 5.000 – 9.000 m
Discretización vertical	10 capas sigma
Celdas activas	5.941
Cálculo	
Variables hidrodinámicas	Resuelve las ecuaciones de movimiento
Propiedades del agua	Resuelve el transporte de salinidad y temperatura
Condiciones de borde	
Borde lateral abierto	Marea astronómica (FES200) y distribución del residuo de niveles medidos en Mar del Plata y La Paloma. 23 nodos Salinidad oceánica = 38 ppm
Borde lateral cerrado	Caudal fluvial variable de los ríos Uruguay y Paraná
Superficie libre	Flujo advectivo y difusivo nulo
Condiciones Iniciales	
Variables hidrodinámicas	η = 0,91 m Velocidad = 0 (reposo)
Propiedades del agua	Temperatura = 20^{o}C Salinidad = 38 ppm
Parámetros	
Paso temporal de cálculo	40 s
Viscosidad horizontal	Formulación de Smagorinsky con un factor de 0,4
Viscosidad vertical	0,0008

Tabla 11: Principales características de implementación del modelo 3D

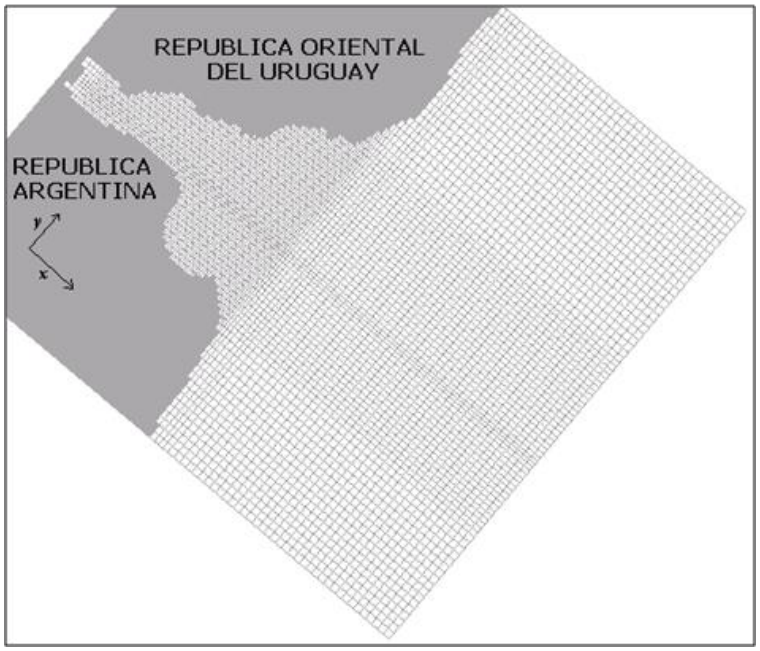

Figura 23: Malla horizontal de cálculo del modelo 3D.

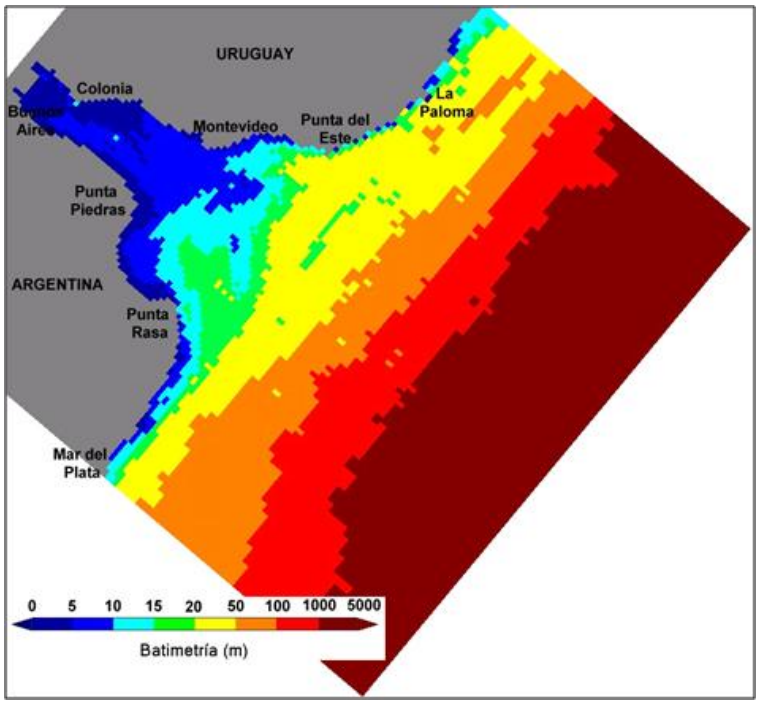

Figura 24: Batimetría ingresada al modelo 3D.

5.4. Evaluación de los tiempos de ejecución

Para medir los tiempos de cálculo se utilizaron las rutinas StartWatch y StopWatch definidas en el módulo StopWatch. Las mismas utilizan los comandos CPU_TIME y DATE_AND_TIME de fortran para calcular el tiempo de cálculo consumido por cada rutina.

Se realizaron cuatro corridas con la implementación evaluada:

- Serial

- Paralela con 2 threads (export OMP_NUM_THREADS=2)
- Paralela con 4 threads (export OMP_NUM_THREADS=4)
- Paralela con 8 threads (export OMP_NUM_THREADS=8)

La cantidad de threads se define antes de lanzar el ejecutable con la variable OMP_NUM_THREADS.

Para la realización de estas pruebas se simuló un período de 6 horas que corresponde a un total de 500 iteraciones.

Las corridas realizadas se llevaron a cabo en el CLUSTER FING, más precisamente en alguno de sus nodos con 8 procesadores.

A continuación se presentan los resultados obtenidos en las pruebas realizadas.

5.5. Módulos principales

En la Tabla 12 se presentan los tiempos de ejecución del modelo 3D. En dicha tabla se presenta el tiempo total de ejecución del modelo, el tiempo consumido por el Main, por el módulo Module y por los principales módulos que componen la ejecución del MOHID Water: Hydrodynamic, WaterProperties y Turbulence. Los mismos valores presentados en la Tabla 12 se grafican en la Figura 25. En dicha figura se puede observar que la distancia entre las tres primeras series (tiempo total, del Main y del módulo Model) se mantiene casi constante. Esto era esperable ya que no se identificaron rutinas paralelizadas en el Main o el módulo Model.

Rutina	**Número Procesadores**			
	1	**2**	**4**	**8**
Total	62.44	43.72	28.44	22.62
Main	60.08	41.30	26.12	20.32
Módulo Model	57.32	39.30	24.63	19.01
Módulo Hydrodynamic	36.86	25.53	16.04	12.23
Módulo WaterProperties	17.74	12.42	7.81	6.20
Módulo Turbulence	2.61	1.27	0.71	0.52

Tabla 12: Tiempos de ejecución de las distintas retinas al variar el número de hilos.

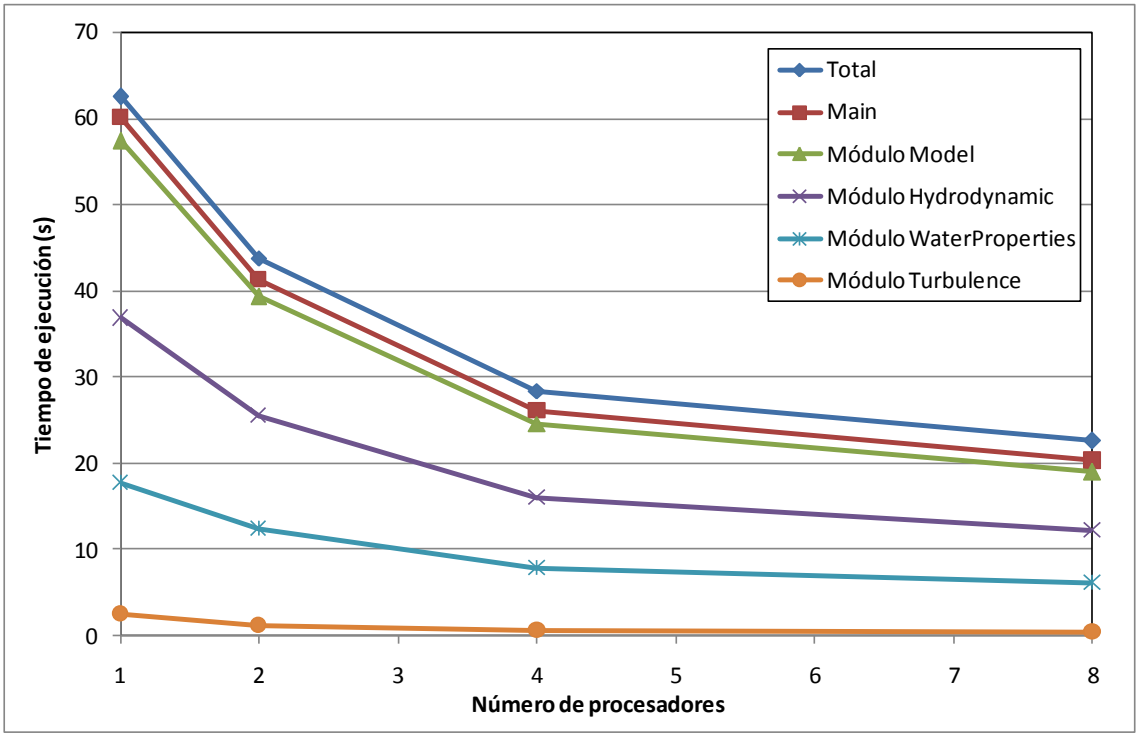

Figura 25: Tiempo de ejecución vs Número de procesadores para el modelo 3D.

En la Figura 26 se presenta un gráfico de torta con la distribución de tiempos de los principales módulos del MOHID Water. En dicha figura se puede observar que el módulo que consume la mayor cantidad de recursos de cálculo es el Hydrodynamic. Su ejecución consume más del 60% del tiempo total de ejecución del modelo. Le sigue el módulo WaterProperties que consume más del 30% del tiempo total de ejecución y por último se ubica el módulo Turbulence que apenas alcanza el 5% del tiempo total de ejecución.

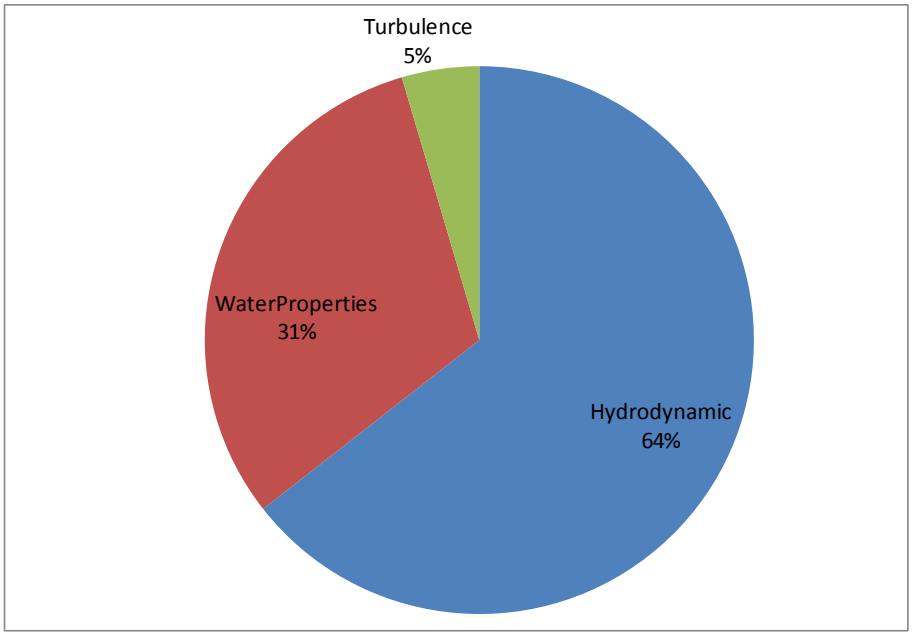

Figura 26: Tiempos de ejecución en versión serial.

En la Figura 27 se presenta un gráfico del Speed up en función de la cantidad de hilos utilizados para ejecutar el programa.

El Speed up es una medida de la mejora de rendimiento de una aplicación al aumentar la cantidad de procesadores (comparado con el rendimiento al utilizar un solo procesador). Siendo T_N el tiempo total de ejecución de una aplicación utilizando N procesadores, se define el Speedup algorítmico como:

$$S_N = T1 / T_N$$

Siendo T1 el tiempo en un procesador (serial) y T_N el tiempo paralelo.

La situación ideal es lograr el speedup lineal. Al utilizar p procesadores obtener una mejora de factor p. La realidad indica que es habitual obtener speedup sublineal. Utilizar p procesadores no garantiza una mejora de factor p. Algunos de los motivos que impiden el crecimiento lineal del Speedup son: el overhead en intercambio de datos, el overhead en trabajos de sincronización, la existencia de tareas no paralelizables (Ley de Amdahl), etc. Los factores mencionados incluso pueden producir que el uso de más procesadores sea contraproducente para la performance de la aplicación.

En la Figura 27 se puede observar que el módulo que presenta el mejor desempeño en paralelo es el Turbulence. Se podría decir que dicho módulo presenta un speed-up lineal con 2 y 4 procesadores, presentando un speed-up sublineal al aumentar el número de hilos a 8.

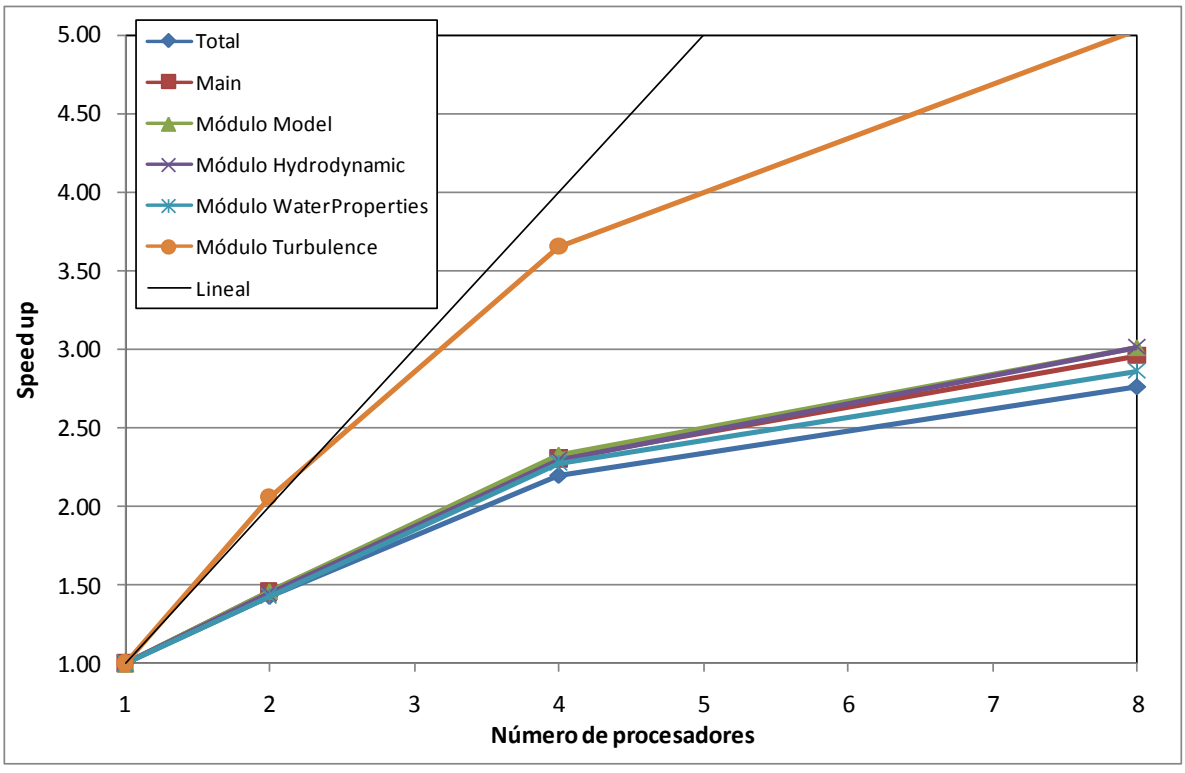

Figura 27: Speed up del modelo 3D.

5.5.1.1. Módulo Hydrodynamic

En la Figura 28 se presenta el tiempo de cálculo consumido por las principales rutinas que integran el módulo Hydrodynamic. En dicha figura se puede observar que la rutina que presenta el mayor costo computacional es la Modify_Hydrodynamic. Dentro de ésta, la rutina One_Iteration consume la mayor parte del tiempo. Dentro de la subrutina One_Iteration, la rutina Leendertse_Scheme consume una porción importante del tiempo. Ésta última invoca a las subrutinas Explicit_Forces, Compute_Velocity y Compute_WaterLevel las cuales finalmente ejecutan los algoritmos de Thomas, haciendo uso del módulo Functions.

En la Figura 29 se puede observar que las principales rutinas del módulo Hydrodynamic están paralelizadas ya que su tiempo de ejecución disminuye a medida que aumentamos el número de procesadores. Sin embargo, a medida que aumentamos la cantidad de hilos de ejecución, la disminución observada en los tiempos de cálculo es menor.

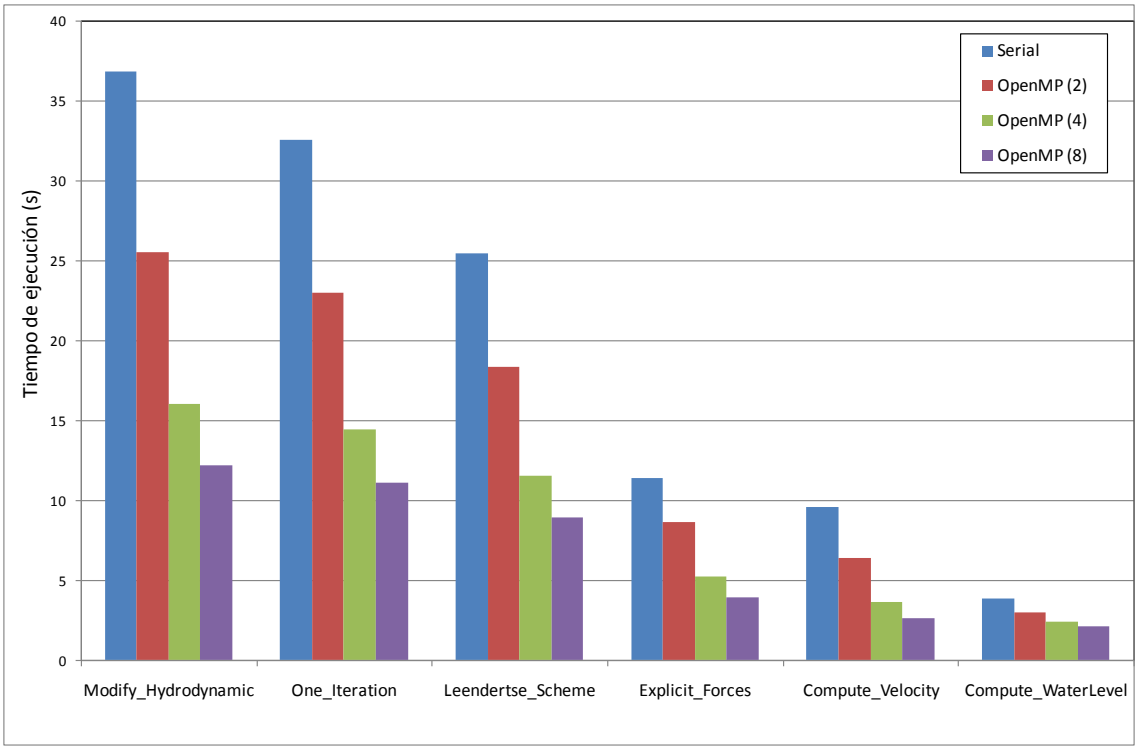

Figura 28: Tiempos de ejecución de las principales rutinas incluidas en el módulo Hydrodynamic.

5.5.1.2. Módulo WaterProperties

En la Figura 29 se presentan los tiempos de cálculo de algunas de las rutinas incluidas en el módulo WaterProperties. En dicha figura se puede observar que la rutina que consume la mayor parte del tiempo es la denominada WaterProperties_Evolution. Dentro de ésta, se ubica la rutina AdvectionDiffusion_Processes que es la responsable de consumir la mayor parte del tiempo. Esta rutina invoca a la rutina AdvectionDiffusion ubicada en el módulo AdvectionDiffusion. La rutina AdvectionDiffusion se encuentra paralelizada, lo que

explica la disminución en los tiempos de cálculo de las rutinas AdvectionDiffusion_Processes y WaterProperties_Evolution al aumentar el número de threadas. Por otro lado, en la Figura 29 se puede observar claramente que la rutina OutPut_Results_HDF5 no está paraleliza dado que su costo computacional se mantiene invariante al variar el número de hilos de ejecución. Esto coincide con los resultados alcanzados durante la etapa de identificación de las rutinas paralelizadas con OpenMP, presentados en el capítulo 5.

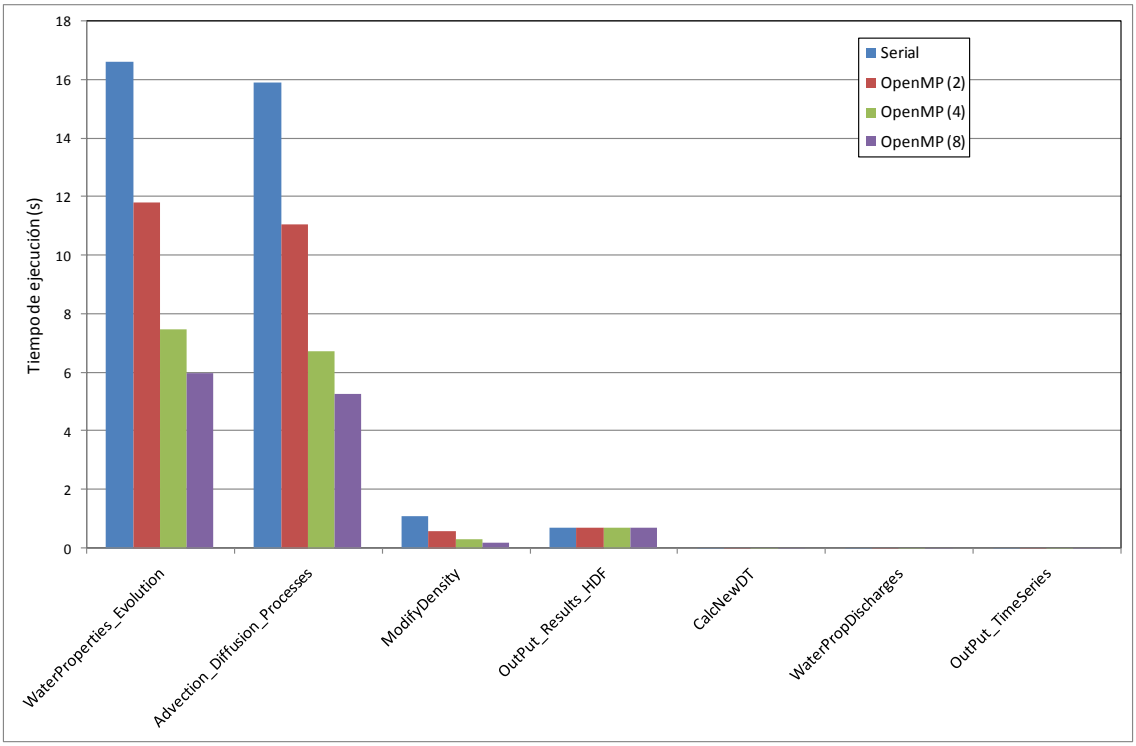

Figura 29: Tiempos de ejecución de las rutinas incluidas en el módulo WaterProperties.

5.6. Otros módulos

En la Figura 30 y Figura 31 se presentan los tiempos de ejecución y el speed-up alcanzado por los módulos AdvectionDiffusion, Functions, Geometry y OpenBoundary al variar el número de procesadores. Dichas figuras muestran que el módulo que presenta un mejor desempeño al aumentar el número de hilos de ejecución es el Functions, seguido del Geometry y luego del AdvectionDiffusion. Esto se debe principalmente a que las rutinas de mayor costo computacional del módulo Functions son las correspondientes a la resolución de los sistemas lineales (algoritmos de Thomas) y éstas se encuentran paralelizadas con directivas OpenMP.

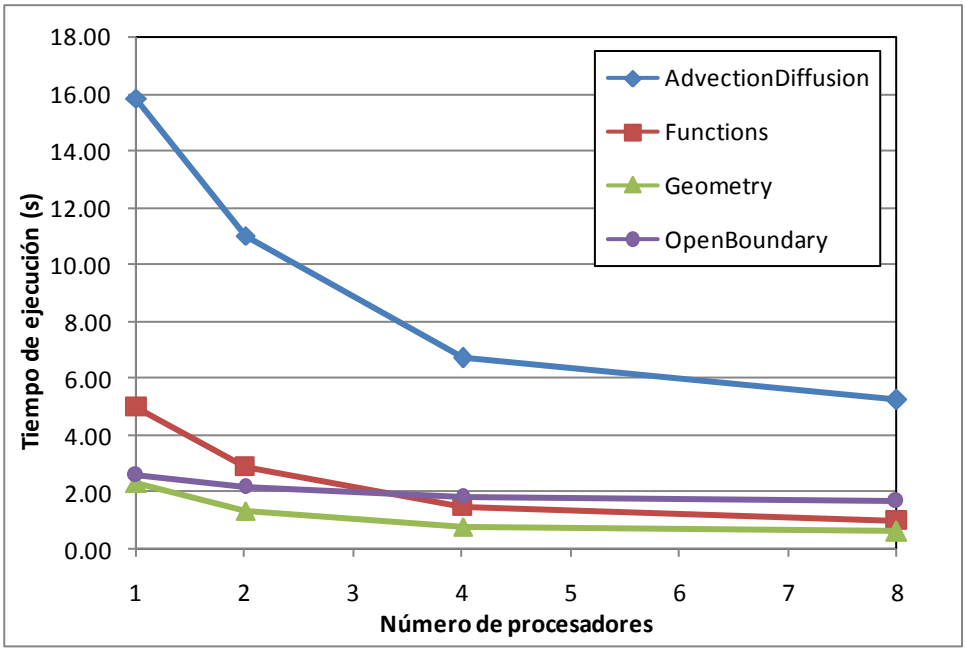

Figura 30: Tiempos de ejecución de los módulos secundarios.

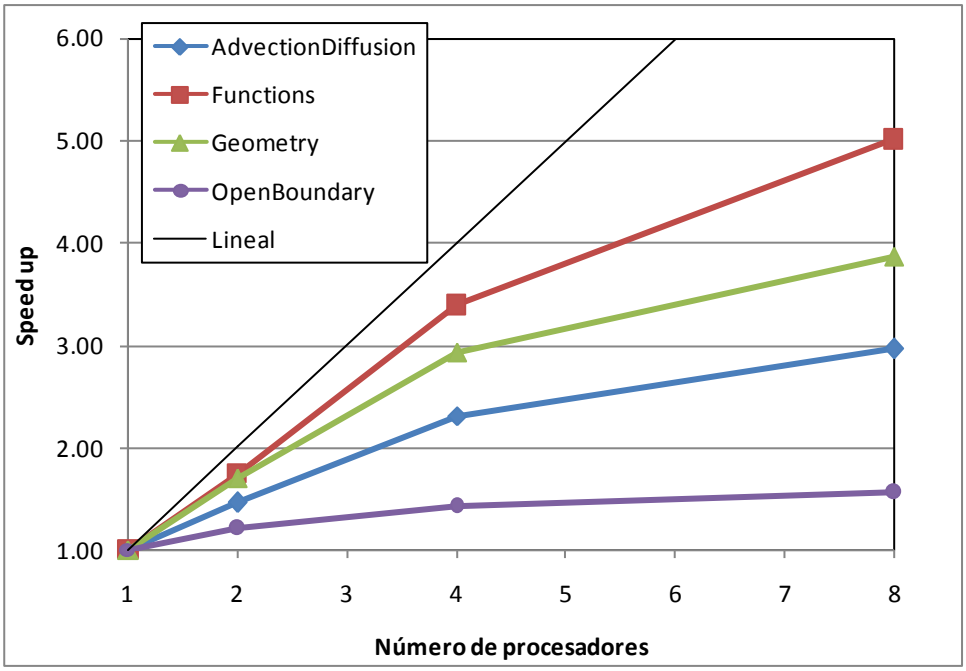

Figura 31: Speed up de los módulos secundarios.

5.7. Síntesis

En este capítulo se presentó un estudio específico realizado con el objetivo de identificar las rutinas del código del MOHID que incluyen secciones paralelizadas con OpenMP y que intervienen en la implementación del modelo a

ser evaluada. Además, se realizó una evaluación de los tiempos de ejecución de la versión serial y paralela variando el número de threads.

Como resultado se ha logrado identificar aquellas secciones del código que utilizan directivas OpenMP para paralelizar la ejecución del modelo y se ha evaluado el desempeño del código paralelizado en comparación con el código serial. Estas pruebas se han realizado en el cluster FING con una implementación del modelo realizada por el IMFIA. De estas pruebas se concluye que si bien el código paralelizado presenta un speed-up sublineal, al pasar de la versión serial a la versión paralela con 2 y 4 hilos de ejecución, se logra disminuir significativamente los tiempos de cálculo. Sin embargo, en la implementación evaluada, la disminución del tiempo de ejecución no es tan pronunciada al pasar de 4 a 8 threads.

6. Síntesis Final

En este reporte se ha presentado una detallada descripción del modelo hidrodinámico MOHID, desde las ecuaciones que gobiernan el modelo y la metodología para su resolución numérica hasta la arquitectura y principales rutinas del código que la implementan.

Fue descrita en detalle la metodología empelada por el modelo para la resolución de sistemas lineales por ser una de las componentes claves en los tiempos de ejecución. A partir de una implementación serial bidimensional típica para ciertas aplicaciones que se llevan a cabo en el IMFIA se evaluaron los tiempos de ejecución de las distintas rutinas del modelo permitiendo identificar aquellas que son determinantes en el tiempo de ejecución total.

El trabajo mencionado anteriormente fue realizado con una versión del código correspondiente a junio de 2007, posteriormente se configuró y compiló una nueva versión del código del MOHID Water con la opción de paralelización OpenMP. En esta nueva versión del código (mayo 2011) se han implementado varias secciones paralelizadas con OpenMP que el código anterior no había terminado de implementar, incluyendo entre éstas el algoritmo de Thomas. En base a lo anterior se recomienda fuertemente realizar una actualización periódica del código del MOHID en el cluster.

En base a esta nueva versión del modelo se identificaron aquellas secciones del código que utilizan directivas OpenMP para paralelizar la ejecución del modelo y se ha evaluado el desempeño del código paralelizado en comparación con el código serial. Estas pruebas se han realizado en el cluster FING con una implementación tridimensional del modelo realizada por el IMFIA. De estas pruebas se concluye que si bien el código paralelizado presenta un speed-up sublineal, al pasar de la versión serial a la versión paralela con 2 y 4 hilos de ejecución, se logra disminuir significativamente los tiempos de cálculo. Sin embargo, en la implementación evaluada, la disminución del tiempo de ejecución no es tan pronunciada al pasar de 4 a 8 threads.

7. Referencias

Abbott, M., A. Damsgaard & G. Rodenhuis. *System 21 Jupiter, A design system for two-dimensional nearly-horizontal flows.* H. Hyd. Res. 1_1-28, 1973.

Barreto, I.; Ezzati, P.; Fossati, M. Estudio Inicial del modelo MOHID (2009). Reporte Técnico RT 09-10 del PEDECIBA Informática. INCO-FING, UdelaR, Uruguay.

Ferziger, J., Períc, M., *Computational Methods for Fluid Dynamics*. Thrid rev. edition, Springer, 423 p., 2002.

Fletcher, R., *Conjugate gradient methods for indefinite systems. Lecutre Notes in Mathematics*, 506, 773-789, 1976.

Fossati, M. y Piedra-Cueva, I. (2006). Modelación tridimensional de la circulación en el Río de la Plata. XXII Congreso Latinoamericano de Hidráulica, Ciudad Guayana, Venezuela.

Golub, G.H., van Loan, C., Matrix computations. Johns Hopkins Univ. Press, Baltimore (1990).

Hermanns, M. Parallel Programming in Fortran 95 using OpenMP (2002). School of Aeronautical Engineering, Universidad Politécnica de Madrid, España.

Leitão, P. C.. *Modelo de Dispersão Lagrangeano Tridimensional.* Ms. Sc. Thesis, Universidade Técnica de Lisboa, Instituto SuperiorTécnico, 1996.

Martins, F.. *Modelação Matemática Tridimensional de Escoamentos Costeiros e Estuarinos usando uma Abordagem de Coordenada Vertical Genérica.* Ph. D, Thesis, Universidade Técnica de Lisboa, Instituto Superior Técnico, 1999.

Martins, F., R. Neves, P. Leitão & A. Silva. *3-D modeling in the Sado estuary using a new generic coordiante approach.* J. Geophys. Res., 1999.

MOHID. www.MOHID.com

MOHID. MOHID description (2003).

MOHID. Hydrodynamic Module User Guide (2006). Instituto Superior Tecnico – MARETEC.

MOHID Wiki, www.MOHID.com/wiki

Montero, P.. Estudio de la hidrodinámica de la Río de Vigo mediante un modelo de volúmenes finitos. Tesis de doctorado, Universidad Santiago de Compostela, Facultad de Física, 1999.

Mourik T. Fortran 90/95 Programming Manual (2005). Chemistry Department, University College London.

Leendertse, J., *Aspects of a computational model for long water wave propagation.* Rand Corporation Memorandum RH-5299-RR, Santa Monica, 1967.

Padman, R. Programming in Fortran 95 - Self-study guide 2 (2007). Computational Physics, Department of Physics, University of Cambridge.

OpenMP. www.openmp.org/drupal/

Stone, H.L., Iterative solution of implicit approximations of multidimensional partial differential equations. SIAM J. Numer. Anal., 5, 530-558, 1968.

Taboada, J., Prego, R, Ruiz-Villarreal, M., Montero, P., Gómez- Gesteira, M., Santos, A. y Pérez-Villar, V.. *Evaluation of the seasonal variations in the residual patterns in the Ría de Vigo (NWSpain) by means of a 3D baroclinic model.* Estuarine Coastal and Shelf Science 47, pp. 661-670, 1998.

Villarreal, M.. *Parametrization of turbulence in the ocean and application of a 3D baroclinic model to the Ria de Pontevedra.* Ph. D. Dissertation, Universidad de Santiago de Compostela, 2000.

Anexo I: Códigos de las implementaciones del TDMA

A1.1. Thomas_2D

```
subroutine THOMAS_2D (IJmin, IJmax,                          &
                      JImin, JImax,                          &
                      di,    dj,                             &
                      DCoef_2D, ECoef_2D, FCoef_2D, TiCoef_2D,  &
                      ANSWER,                                &
                      VECG, VECW)
 !Arguments---------------------------------------------------------------
double precision :: RTC
integer :: MCLOCK
integer,                intent(IN) :: IJmin, IJmax
integer,                intent(IN) :: JImin, JImax
integer,                intent(IN) :: di,    dj
real,    dimension(:,:), pointer       :: DCoef_2D, FCoef_2D, TiCoef_2D
real(8), dimension(:,:), pointer       :: ECoef_2D
real,    dimension(:,:), pointer       :: ANSWER
real(8), dimension(:  ), pointer       :: VECG, VECW

!Local--------------------------------------------------------------------
integer :: IJ, JI, II, MM, I, J
!-------------------------------------------------------------------------

do2 :   do IJ = IJmin, IJmax
          I = IJ*dj + di
          J = IJ*di + dj
          VECW(JImin) = -FCoef_2D (I, J)/ECoef_2D(I, J)
          VECG(JImin) =  TiCoef_2D(I, J)/ECoef_2D(I, J)

do3 :      do JI=JImin+1,JImax+1
             I      = IJ*dj + JI*di
             J      = IJ*di + JI*dj
             VECW(JI) = -FCoef_2D(I,J) / (ECoef_2D(I,J) + DCoef_2D(I,J) * VECW(JI-1))
             VECG(JI) = (TiCoef_2D(I,J) - DCoef_2D(I,J) * VECG(JI-1))/  &
                    (ECoef_2D (I,J) + DCoef_2D(I,J) * VECW(JI-1))
          end do do3
          I = IJ * dj + (JImax+1) * di
```

```
          J = IJ * di + (JImax+1) * dj
          ANSWER(I, J) = VECG(JImax+1)

do1 :     do II = JImin+1, JImax+1
            MM = JImax+2-II
            I  = IJ*dj + MM*di
            J  = IJ*di + MM*dj
            ANSWER(I,J) = VECW(MM) * ANSWER(I+di,J+dj) + VECG(MM)
          end do do1
        end do do2
      end subroutine THOMAS_2D
```

A1.2. Thomas_3D_NoOpenMP

```
subroutine THOMAS_3D_NoOpenMP(IJmin, IJmax,                    &
                              JImin, JImax,                    &
                              Kmin,  Kmax,                     &
                              di,    dj,                       &
                              DCoef_3D, ECoef_3D, FCoef_3D, TiCoef_3D,  &
                              ANSWER,                          &
                              VECG, VECW)

    !Arguments---------------------------------------------------------------
    integer,                  intent(IN) :: IJmin, IJmax
    integer,                  intent(IN) :: JImin, JImax
    integer,                  intent(IN) :: Kmin , Kmax
    integer,                  intent(IN) :: di,    dj
    real,    dimension(:,:,:), pointer       :: DCoef_3D, FCoef_3D, TiCoef_3D
    real(8), dimension(:,:,:), pointer       :: ECoef_3D
    real,    dimension(:,:,:), pointer       :: ANSWER
    real(8), dimension(:  ), pointer         :: VECG, VECW
    !Local--------------------------------------------------------------------
   if (di == 0 .and. dj == 1) then
      call THOMAS_3D_i0_j1(IJmin, IJmax,                       &
                           JImin, JImax,                       &
                           Kmin,  Kmax,                        &
                           DCoef_3D, ECoef_3D, FCoef_3D, TiCoef_3D,  &
                           ANSWER,                             &
                           VECG, VECW)
```

```
      else if (di == 1 .and. dj == 0) then
        call THOMAS_3D_i1_j0(IJmin, IJmax,                    &
                             JImin, JImax,                    &
                             Kmin,  Kmax,                     &
                             DCoef_3D, ECoef_3D, FCoef_3D, TiCoef_3D,  &
                             ANSWER,                          &
                             VECG, VECW)
      else
                stop 'THOMAS_3D_NoOpenMP - Module Functions - ERR01'
      endif
    end subroutine THOMAS_3D_NoOpenMP
```

A1.3. Thomas_3D_io_j1

```
    subroutine THOMAS_3D_i0_j1 (IJmin, IJmax,                 &
                             JImin, JImax,                    &
                             Kmin,  Kmax,                     &
                             DCoef_3D, ECoef_3D, FCoef_3D, TiCoef_3D,  &
                             ANSWER,                          &
                             VECG, VECW)

      integer,                intent(IN) :: IJmin, IJmax
      integer,                intent(IN) :: JImin, JImax
      integer,                intent(IN) :: Kmin , Kmax
      real,    dimension(:,:,:), pointer      :: DCoef_3D, FCoef_3D, TiCoef_3D
      real(8), dimension(:,:,:), pointer      :: ECoef_3D
      real,    dimension(:,:,:), pointer      :: ANSWER
      real(8), dimension(:  ), pointer        :: VECG, VECW

    !Local-----------------------------------------------------------------
      integer                         :: IJ, JI, II, K,
    !Begin-----------------------------------------------------------------
    do4:   do K  = Kmin, Kmax
    do2 :  do IJ = IJmin, IJmax

        VECW(JImin) =-FCoef_3D (IJ, 1, K)/ECoef_3D(IJ, 1, K)
        VECG(JImin) = TiCoef_3D(IJ, 1, K)/ECoef_3D(IJ, 1, K)
```

```
do3 :     do JI=JImin+1,JImax+1
            VECW(JI) = - FCoef_3D(IJ, JI, K) / (ECoef_3D(IJ, JI, K) +          &
                  DCoef_3D(IJ, JI, K) * VECW(JI-1))
            VECG(JI) =  (TiCoef_3D(IJ, JI, K) - DCoef_3D(IJ, JI, K) * VECG(JI-1))/    &
                  (ECoef_3D (IJ, JI, K) + DCoef_3D(IJ, JI, K) * VECW(JI-1))
          end do do3
          ANSWER(IJ, (JImax+1), K) = VECG(JImax+1)
do1 :     do II = JImin+1, JImax+1
            ANSWER(IJ, JImax+2-II, K) = VECW(JImax+2-II) * ANSWER(IJ, JImax+2-II+1, K) +
VECG(JImax+2-II)
          end do do1
        end do do2
        end do do4
      end subroutine THOMAS_3D_i0_j1
```

A1.4. Thomas_3D_i1_j0

```
subroutine THOMAS_3D_i1_j0(IJmin, IJmax,                          &
                           JImin, JImax,                          &
                           Kmin,  Kmax,                       &
                           DCoef_3D, ECoef_3D, FCoef_3D, TiCoef_3D,    &
                           ANSWER,                                &
                           VECG, VECW)

    !Arguments---------------------------------------------------------------
    integer,                intent(IN) :: IJmin, IJmax
    integer,                intent(IN) :: JImin, JImax
    integer,                intent(IN) :: Kmin , Kmax
    real,    dimension(:,:,:), pointer      :: DCoef_3D, FCoef_3D, TiCoef_3D
    real(8), dimension(:,:,:), pointer      :: ECoef_3D
    real,    dimension(:,:,:), pointer      :: ANSWER
    real(8), dimension(:  ), pointer        :: VECG, VECW

    !Local--------------------------------------------------------------------
    integer                         :: IJ, JI, II, K,
    !Begin--------------------------------------------------------------------
do4:   do K  = Kmin, Kmax
do2 :  do IJ = IJmin, IJmax
```

```
            VECW(JImin) =-FCoef_3D (1, IJ, K)/ECoef_3D(1, IJ, K)
            VECG(JImin) = TiCoef_3D(1, IJ, K)/ECoef_3D(1, IJ, K)

      do3 :     do JI=JImin+1,JImax+1
              VECW(JI) = - FCoef_3D(JI, IJ, K) / (ECoef_3D(JI, IJ, K) +          &
                    DCoef_3D(JI, IJ, K) * VECW(JI-1))
              VECG(JI) = (TiCoef_3D(JI, IJ, K) - DCoef_3D(JI, IJ, K) * VECG(JI-1))/    &
                   (ECoef_3D (JI, IJ, K) + DCoef_3D(JI, IJ, K) * VECW(JI-1))
            end do do3
            ANSWER(JImax+1, IJ, K) = VECG(JImax+1)

      do1 :     do II = JImin+1, JImax+1
              ANSWER(JImax+2-II, IJ, K) = VECW(JImax+2-II) * ANSWER(JImax+2-II+1, IJ, K) +
VECG(JImax+2-II)
            end do do1
          end do do2
          end do do4
      end subroutine THOMAS_3D_i1_j0
```

A1.5. Thomas_3D_OpenMP

```
      subroutine THOMAS_3D_OpenMP  (IJmin, IJmax,                    &
                  JImin, JImax,                                      &
                  Kmin,  Kmax,                                       &
                  di,    dj,                                         &
                  DCoef_3D, ECoef_3D, FCoef_3D, TiCoef_3D,           &
                  ANSWER,                                            &
                  VECG, VECW)

        !Arguments---------------------------------------------------------
        integer,                intent(IN) :: IJmin, IJmax
        integer,                intent(IN) :: JImin, JImax
        integer,                intent(IN) :: Kmin , Kmax
        integer,                intent(IN) :: di,    dj
        real,    dimension(:,:,:), pointer      :: DCoef_3D, FCoef_3D, TiCoef_3D
        real(8), dimension(:,:,:), pointer      :: ECoef_3D
        real,    dimension(:,:,:), pointer      :: ANSWER
        real(8), dimension(:,:  ), pointer      :: VECG, VECW
```

```
!Local----------------------------------------------------------------

!Begin----------------------------------------------------------------
if (di == 0 .and. dj == 1) then

  call THOMAS_3D_i0_j1_OMP(IJmin, IJmax,                    &
                JImin, JImax,                               &
                Kmin,  Kmax,                                &
                DCoef_3D, ECoef_3D, FCoef_3D, TiCoef_3D,    &
                ANSWER,                                     &
                VECG, VECW)

else if (di == 1 .and. dj == 0) then

  call THOMAS_3D_i1_j0_OMP(IJmin, IJmax,                    &
                JImin, JImax,                               &
                Kmin,  Kmax,                                &
                DCoef_3D, ECoef_3D, FCoef_3D, TiCoef_3D,    &
                ANSWER,                                     &
                VECG, VECW)

else
        stop 'THOMAS_3D_OpenMP - Module Functions - ERR01'
endif
end subroutine THOMAS_3D_OpenMP
```

A1.6. Thomas_3D_i0_j1_OMP

```
subroutine THOMAS_3D_i0_j1_OMP (IJmin, IJmax,               &
                JImin, JImax,                               &
                Kmin,  Kmax,                                &
                DCoef_3D, ECoef_3D, FCoef_3D, TiCoef_3D,    &
                ANSWER,                                     &
                VECG, VECW)

  !Arguments-----------------------------------------------------------
  integer,              intent(IN) :: IJmin, IJmax
```

```
        integer,                intent(IN) :: JImin, JImax
        integer,                intent(IN) :: Kmin , Kmax
        real,    dimension(:,:,:), pointer      :: DCoef_3D, FCoef_3D, TiCoef_3D
        real(8), dimension(:,:,:), pointer      :: ECoef_3D
        real,    dimension(:,:,:), pointer      :: ANSWER
        real(8), dimension(:,:  ), pointer      :: VECG, VECW

        !Local-----------------------------------------------------------------
        integer                         :: IJ, JI, II, K,
        integer                         :: CHUNK
        !Begin-----------------------------------------------------------------
        CHUNK = CHUNK_K(Kmin, Kmax)

        !$OMP PARALLEL PRIVATE(IJ, JI, II, K)
        !$OMP DO SCHEDULE(DYNAMIC, CHUNK)
    do4:   do K  = Kmin, Kmax
    do2 :  do IJ = IJmin, IJmax

          VECW(JImin, K) =-FCoef_3D (IJ, 1, K)/ECoef_3D(IJ, 1, K)
          VECG(JImin, K) = TiCoef_3D(IJ, 1, K)/ECoef_3D(IJ, 1, K)

    do3 :     do JI=JImin+1,JImax+1
            VECW(JI, K) = - FCoef_3D(IJ, JI, K) / (ECoef_3D(IJ, JI, K) +              &
                    DCoef_3D(IJ, JI, K) * VECW(JI-1, K))

            VECG(JI, K) =  (TiCoef_3D(IJ, JI, K) - DCoef_3D(IJ, JI, K) * VECG(JI-1, K))/    &
                    (ECoef_3D (IJ, JI, K) + DCoef_3D(IJ, JI, K) * VECW(JI-1, K))
          end do do3

          ANSWER(IJ, (JImax+1), K) = VECG(JImax+1, K)

    do1 :     do II = JImin+1, JImax+1
            ANSWER(IJ, JImax+2-II, K) = VECW(JImax+2-II, K) * ANSWER(IJ, JImax+2-II+1, K) +
VECG(JImax+2-II, K)
          end do do1
        end do do2
        end do do4
        !$OMP END DO NOWAIT
```

```
    !$OMP END PARALLEL
  end subroutine THOMAS_3D_i0_j1_OMP

  !---------------------------------------------------------------------------
```

A1.7. Thomas_3D_i1_j0_OMP

```
  subroutine THOMAS_3D_i1_j0_OMP(IJmin, IJmax,                    &
                    JImin, JImax,                                 &
                    Kmin,  Kmax,                                  &
                    DCoef_3D, ECoef_3D, FCoef_3D, TiCoef_3D,      &
                    ANSWER,                                       &
                    VECG, VECW)

    !Arguments---------------------------------------------------------------
    integer,                intent(IN) :: IJmin, IJmax
    integer,                intent(IN) :: JImin, JImax
    integer,                intent(IN) :: Kmin , Kmax
    real,    dimension(:,:,:), pointer      :: DCoef_3D, FCoef_3D, TiCoef_3D
    real(8), dimension(:,:,:), pointer      :: ECoef_3D
    real,    dimension(:,:,:), pointer      :: ANSWER
    real(8), dimension(:,:  ), pointer      :: VECG, VECW
    !Local--------------------------------------------------------------------
    integer                         :: IJ, JI, II, K
    integer                         :: CHUNK

!Begin--------------------------------------------------------------------
    CHUNK = CHUNK_K(Kmin, Kmax)
    !$OMP PARALLEL PRIVATE(IJ, JI, II, K)
    !$OMP DO SCHEDULE(DYNAMIC, CHUNK)
do4:  do K  = Kmin, Kmax
do2 :  do IJ = IJmin, IJmax

      VECW(JImin, K) =-FCoef_3D (1, IJ, K)/ECoef_3D(1, IJ, K)
      VECG(JImin, K) = TiCoef_3D(1, IJ, K)/ECoef_3D(1, IJ, K)

do3 :    do JI=JImin+1,JImax+1
        VECW(JI, K) = - FCoef_3D(JI, IJ, K) / (ECoef_3D(JI, IJ, K) +    &
                DCoef_3D(JI, IJ, K) * VECW(JI-1, K))
```

```
                VECG(JI, K) = (TiCoef_3D(JI, IJ, K) - DCoef_3D(JI, IJ, K) * VECG(JI-1, K))/     &
                        (ECoef_3D (JI, IJ, K) + DCoef_3D(JI, IJ, K) * VECW(JI-1, K))
              end do do3

              ANSWER(JImax+1, IJ, K) = VECG(JImax+1, K)

        do1 :     do II = JImin+1, JImax+1
                  ANSWER(JImax+2-II, IJ, K) = VECW(JImax+2-II, K) * ANSWER(JImax+2-II+1, IJ, K) +
VECG(JImax+2-II, K)
              end do do1
            end do do2
            end do do4
            !$OMP END DO NOWAIT
            !$OMP END PARALLEL
          end subroutine THOMAS_3D_i1_j0_OMP
```

A1.8. Thomas_Z_NoOpenMP

```
subroutine THOMASZ_NoOpenMP (ILB, IUB,                    &
                JLB, JUB,                                 &
                KLB, KUB,                                 &
                AW, BW, CW, TIW,                          &
                RES,                                      &
                VECG, VECW)

!Arguments------------------------------------------------------------------
integer,                intent(IN) :: ILB, IUB
integer,                intent(IN) :: JLB, JUB
integer,                intent(IN) :: KLB, KUB
real,    dimension(:,:,:), pointer      :: AW, CW, TIW
real(8), dimension(:,:,:), pointer      :: BW
real,    dimension(:,:,:), pointer      :: RES
real(8), dimension(:   ), pointer      :: VECG, VECW

!Local----------------------------------------------------------------------
integer :: I, J, K
integer :: II, MM
!---------------------------------------------------------------------------
```

```
do2 :   DO J = JLB, JUB
do1 :   DO I = ILB, IUB
        VECW(KLB) =-CW (I, J, 1) / BW(I, J, 1)
        VECG(KLB) = TIW(I, J, 1) / BW(I, J, 1)

do3 :     DO K  = KLB+1, KUB+1
          VECW(K) = -CW(I, J, K) / (BW(I, J, K) + AW(I, J, K) * VECW(K-1))

          VECG(K) = (TIW(I, J, K) - AW(I, J, K) * VECG(K-1))          &
                / (BW (I, J, K) + AW(I, J, K) * VECW(K-1))
        END DO do3

        RES(I, J, KUB+1) = VECG(KUB+1)

do4 :     DO II = KLB+1, KUB+1
          MM          = KUB + 2 - II
          RES(I, J, MM) = VECW(MM) * RES(I, J, MM+1) + VECG(MM)
        END DO do4
      END DO do1
      END DO do2
  end subroutine THOMASZ_NoOpenMP
```

A1.9. Thomas_Z_OpenMP

```
      subroutine THOMASZ_OpenMP (ILB, IUB,                    &
                                 JLB, JUB,                    &
                                 KLB, KUB,                    &
                                 AW, BW, CW, TIW,             &
                                 RES,                         &
                                 VECG, VECW)

          !Arguments-------------------------------------------------------------
          integer,                intent(IN) :: ILB, IUB
          integer,                intent(IN) :: JLB, JUB
          integer,                intent(IN) :: KLB, KUB
          real,    dimension(:,:,:), pointer      :: AW, CW, TIW
          real(8), dimension(:,:,:), pointer      :: BW
          real,    dimension(:,:,:), pointer      :: RES
          real(8), dimension(:,:  ), pointer      :: VECG, VECW
```

```
      !Local-------------------------------------------------------------------
      integer                       :: I, J, K
      integer                       :: II, MM
      integer                       :: CHUNK

      !Begin--------------------------------------------------------------------
      CHUNK = CHUNK_J(JLB, JUB)

      !$OMP PARALLEL PRIVATE(I, J, K, II, MM)

      !$OMP DO SCHEDULE(DYNAMIC, CHUNK)
do2 :   DO J = JLB, JUB
do1 :   DO I = ILB, IUB
          VECW(KLB, J) =-CW (I, J, 1) / BW(I, J, 1)
          VECG(KLB, J) = TIW(I, J, 1) / BW(I, J, 1)

do3 :       DO K  = KLB+1, KUB+1
            VECW(K, J) = -CW(I, J, K) / (BW(I, J, K) + AW(I, J, K) * VECW(K-1, J))

            VECG(K, J) = (TIW(I, J, K) - AW(I, J, K) * VECG(K-1, J))          &
                  / (BW (I, J, K) + AW(I, J, K) * VECW(K-1, J))
          END DO do3

          RES(I, J, KUB+1) = VECG(KUB+1, J)

do4 :       DO II = KLB+1, KUB+1
            MM          = KUB + 2 - II
            RES(I, J, MM) = VECW(MM, J) * RES(I, J, MM+1) + VECG(MM, J)
          END DO do4
        END DO do1
        END DO do2
        !$OMP END DO NOWAIT
        !$OMP END PARALLEL
      end subroutine THOMASZ_OpenMP
```

Anexo II Descripción de OpenMP

En este Anexo se presenta una breve síntesis de los conceptos básicos de programación con OpenMP. Mayor información del lenguaje OpenMP se puede consultar en la página http://www.openmp.org.

OpenMP es una interfaz de programación de aplicaciones (API) para la programación multiproceso de memoria compartida en múltiples plataformas. Permite añadir concurrencia a los programas escritos en C, C++ y Fortran sobre la base del modelo de ejecución fork-join. Está disponible en muchas arquitecturas y se compone de un conjunto de directivas de compilador, rutinas de biblioteca, y variables de entorno que influencian el comportamiento en tiempo de ejecución.

OpenMP es un modelo de programación portable y escalable que proporciona a los programadores una interfaz simple y flexible para el desarrollo de aplicaciones paralelas para las plataformas que van desde las computadoras de escritorio hasta las supercomputadoras.

A2.1. Modelo de ejecución

La paralelización con OpenMP se logra mediante la definición de *regiones paralelas* (Parallel regions) y mediante la creación de un conjunto de *hilos* (threads). El conjunto de hilos incluye un *hilo principal* (Master thread) y un conjunto de *hilos trabajadores* (Worker threads). A diferencia del procesamiento en serie, el procesamiento dentro de una región paralela se divide entre los distintos hilos que trabajan en forma simultánea.

Cuando se incluye una directiva OpenMP, implícitamente se incluye al final de la región paralela una barrera de sincronización de los diferentes hilos que recolecta sus resultados y los une en un solo resultado; salvo que explícitamente se indique lo contrario con la directiva nowait.

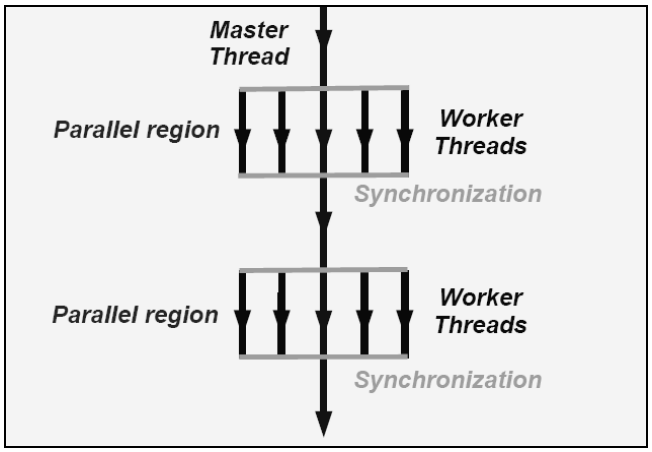

Figura 32: Esquema de ejecución con OpenMP (modelo fork-join) [extraido de OpenMP].

Cada hilo puede tener un conjunto de variables privadas, que sólo pueden ser alteradas por este hilo. La elección de las variables privadas que se definen explícitamente en el código, es una parte central de la programación con OpenMP.

A2.2. Sintaxis básica

Las instrucciones de OpenMP son proporcionados por una serie de directivas. Estas se refieren a diversas acciones tales como la definición de regiones paralelas. Las directivas son específicas del lenguaje de programación subyacente, ya sea C o Fortran.

La sintaxis básica de una directiva de OpenMP en Fortran es:

!$OMP <directiva> [cláusulas]

[bloque de código]

!$OMP END <directiva>

Las cláusulas se utilizan para especificar información adicional a la directiva.

Directivas y cláusulas

En la Tabla 13 se presentan algunas de las directivas más importantes y en la Tabla 14 se presentan algunas de las cláusulas más importantes.

Directiva	Acción
PARALLEL	Indica que la sección puede ser ejecutada por varios hilos.
CRITICAL	Indica que sólo un hilo puede estar en esta sección. No existen barreras al inicio o al final de la sección y todos los hilos pueden procesar esta sección pero sólo uno por vez.
SINGLE	Indica que la sección puede ser ejecutada por un único hilo de todos los lanzados, sin ser obligatoriamente el hilo principal. No existen barreras al inicio o al final de la sección.
MASTER	Indica que la sección puede ser ejecutada únicamente por el hilo principal. No existen barreras al inicio o al final de la sección.
BARRIER	Indica que los hilos deben esperar a que todos los hilos alcancen el punto de la barrera antes de continuar. Una barrera puede ser introducida al final de la directiva SINGLE/MASTER para evitar problemas de sincronización.
DO	Las iteraciones del loop se dividen entre los hilos lanzados. La forma en que el trabajo es dividido entre los distintos hilos se determina con la cláusula SCHEDULE (ver Tabla 14)

Tabla 13: Directivas de OpenMP

Claúsula	Acción
PRIVATE	Lista de variables privadas
NOWAIT	Especifica que los hilos no se deben esperar al final de una directiva.
SCHEDULE (TYPE, CHUNK)	STATIC TYPE: se asigna un número de iteraciones a todos los hilos antes de ejecutar el loop. Por defecto, las iteraciones se dividen entre los hilos en partes iguales. Si se especifica un valor al parámetro "CHUNK" se asignará "CHUNK" número de iteraciones contiguas a cada hilo. La opción estática es aconsejable cuando la cantidad de iteraciones del loop es predecible.Reparte el trabajo de los DO y FOR loops entre los distintos hilos.

Claúsula	Acción
	DYNAMIC TYPE: El trabajo es asignado a los distintos hilos a medida que terminan su asignación anterior. Cada hilo procesa un número fijo de de iteraciones determinado por el parámetro "CHUNK". Después de terminar dicho número de iteraciones cada hilo inicia otra fracción del loop disponible hasta que todas las iteraciones se hayan completado. La opción dinámica es aconsejable cuando el loop involucra una cantidad de iteraciones que no es predecible.

Tabla 14: Cláusulas de OpenMP

Funciones

En la Tabla 15 se presentan algunas de las funciones de OpenMP.

Claúsula	Acción
OMP_SET_NUM_THREADS	Fija el número de hilos simultáneos.
OMP_GET_NUM_THREADS	Devuelve el número de hilos en ejecución.
OMP_GET_MAX_THREADS	Devuelve el número máximo de hilos que lanzará nuestro programa en las zonas paralelas. Es muy útil para reservar memoria para cada hilo.
OMP_GET_THREAD_NUM	Devuelve el número del hilo dentro del equipo (valor entre 0 y OMP_GET_NUM_THREADS()-1)
OMP_GET_NUM_PROCS	Devuelve en número de procesadores de nuestro ordenador o disponibles (en caso de sistemas virtuales).
OMP_SET_DINAMIC	Valor booleano que nos permite especificar si queremos que el número de hilos crezca y decrezca dinámicamente.

Tabla 15: Funciones de OpenMP.

A2.3. Consideraciones importantes

Parallel overhead

A pesar de que la paralelización de un código implica potenciales ganancias en el uso de recursos informáticos, su utilización también implica gastos de recursos que en ocasiones pueden ser significativos.

La creación del conjunto de hilos al principio de cada región paralela es una fuente de gastos. Debido a esto, es preferible en cada programa o rutina crear una sola región paralela y luego usar las directivas de OpenMP para que un solo hilo ejecute las tareas seriales, en lugar de crear varias regiones paralelas. Otra fuente de gastos es la sincronización entre los hilos. Al sobrecosto introducido por la ejecución paralela de un programa se le denomina "parallel overhead".

Estos gastos hacen que en algunos casos la versión paralela de un código no presente un mejor desempeño que su versión serial. Esto ocurre especialmente en los casos que el costo computacional del problema es bajo. En general, a medida

que la escala del problema aumenta los gastos son relativamente menos importantes y la paralelización es más ventajosa.

Condición de carrera o "Race condition"

La condición de carrera se da principalmente cuando varios hilos acceden al mismo tiempo a un recurso compartido, por ejemplo una variable, cambiando su estado y obteniendo de esta forma un valor no esperado de la misma. Esto ocasiona generalmente que la ejecución del programa se suspenda sin ningún tipo de mensaje de error.

El programador debe tener cuidado para evitar esta situación ya que su depuración es muy difícil. Una forma de evitar condiciones de carrera es utilizar la directiva CRITIC o la cláusula REDUCTION.