UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MATEMÁTICA

MODELO HIDRODINÁMICO DE LA MAREA ROJA EN LA BAHÍA QUELLÓN.

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN
CIENCIAS DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MATEMÁTICO

RODRIGO ANDRÉ ZELADA MANCINI

PROFESOR GUÍA:
CARLOS CONCA ROSENDE
PROFESOR COGUÍA :
JORGE SAN MARTÍN HERMOSILLA

MIEMBROS DE LA COMISIÓN:
RAÚL GORMAZ ARANCIBIA

Este trabajo ha sido parcialmente financiado por Proyecto Fondecyt 1151512 y CMM Conicyt PIA AFB170001
Powered@NLHPC: Esta tesis fue parcialmente apoyada por la infraestructura de supercómputo del NLHPC (ECM-02)

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE
INGENIERIO CIVIL MATEMÁTICO Y GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS
POR: RODRIGO ANDRÉ ZELADA MANCINI
FECHA: 24/03/2020
PROF. GUÍA: CARLOS CONCA ROSENDE
PROF. COGUÍA: JORGE SAN MARTÍN HERMOSILLA

MODELO HIDRODINÁMICO DE LA MAREA ROJA EN LA BAHÍA QUELLÓN.

El fenómeno de las mareas rojas ha afectado en las últimas décadas las costas del sur de Chile, la región de los Lagos, la región de Aysén y la de Magallanes. En particular Chiloé se ha visto fuertemente afectado con estos episodios de marea roja, generando pérdidas ecónomicas para la industria pesquera y además peligro en la salud de quienes consumen mariscos, debido al nivel de toxicidad que generan las microoalgas de la marea roja.
El objetivo de este trabajo es desarrollar un modelo matemático, que permita entender la dinámica de estos eventos y además ser capaces de predecirlos. En primera instancia se realiza un estudio teórico de las ecuaciones que rigen la dinámica de fluidos incompresibles (Navier-Stokes), luego se estudian las discretizaciones de las ecuaciones de Navier-Stokes para calcular numericamente la velocidad de la marea y finalmente se calcula la concentración de microalgas en el mar usando la ecuación de transporte de contaminantes de convección-difusión para estudiar la concentración de microoalgas en el mar.

Dedicado a mi madre, padre, hermana, sobrinos y mi polola Amanda.

Agradecimientos

Durante estos años en la Universidad conocí a muchas personas, sobre todo en mi época de mechón. Solía pensar que todos eran mis amigos, sin embargo con el correr de los años me he dado cuenta quiénes han estado en cada momento, que aunque no los veas muy seguido, sabes que puedes contar con ellos. El camino a recorrer no fue fácil y es por ello que es importante agradecer a cada una de las personas que estuvieron en el camino, dando apoyo, conteniendo cuando lo necesité; y también por sacar una sonrisa, distraerse es una parte fundamental para poder trabajar bien.

Quiero agradecer en primer lugar a aquellas amistades que forjé en la Universidad. A Felipe y Selma, compañeros de sección y somos amigos hasta el día de hoy, siempre he contado con ellos pese que desde comenzar la especialidad nos vemos en menor medida.

En este camino participé en un grupo organizado, el CDE (Centro de Desarrollo Estudiantil). Las personas que conocí acá tienen un corazón enorme y pasión por ser un aporte a la sociedad, desde la perspectiva de nuestro rol como futuros ingenieros, siendo un aporte activo. A Nicolás, Joaquín, Rafael, Francisca, Paula, Guillermo, con quienes visitamos el Sename Cread Galvarino en varias ocasiones, regalándole una sonrisa a los niños, quienes con una tarde de juegos eran inmensamente felices. Son bacanes chiquillos, espero no cambien, aprendí mucho y crecí como persona, que muchas veces se deja de lado.

A mis compañeros y amigos del DIM, Daniel una persona tremendamente talentosa que fue un ejemplo a seguir, Reidmen de quién aprendí la importancia del trabajo constante, Pablo con quién trabajamos juntos en varios ramos y además compañero de oficina, cuántas jornadas de estudio en el laboratorio (estamos destinados a estar juntos al parecer, mala mía), Matus a quién aprendí a conocer al final y es alguien tremendamente preocupado por sus compañeros, Diego que es un amor de persona, Vicente que es alguien distinto y muy multifacético.

A mis amigos de la vida Héctor, Yasmin, Claudia, Valeska, Odette, Soledad. Cuando los conocí me sorprendió lo desinhibidos que eran. Cuantas tertulias juntos, también compañeros de viajes, ya han sido años y aún recuerdo como si fuera hoy el día en que nos conocimos. Gracias por aguantarme todo este tiempo, apoyarme y acompañarme en todas, si hay una palabra que los describa yo diría que es motivación.

A Amanda Paz Gallegos Vergara, a quién conozco desde hace cuatro años y hace dos años que somos pareja. En esta última etapa te convertiste en mi pilar, eres una persona tremendamente cariñosa, espontánea, que sabe sacarme una sonrisa a diario y me dabas una palabra de aliento cuando lo necesitaba. Siempre me sacabas de la rutina, viendo películas, saliendo, carreteando, hablando de la vida, imaginando un futuro. Cambiaste mi perspectiva de pareja, una persona muy madura y de la que ha aprendido un montón, gracias por la paciencia. Eres mi amiga, polola, compañera. Como dijiste en alguna oportunidad, todo contigo o nada con

nadie.
A mi profesor guía Carlos Conca, a quién siempre admiré como profesor, me encantaban sus clases, la forma de enseñar y también cuando hablaba de matemática en sí, de grandes matemáticos y demases, hizo despertar en mi la curiosidad, la cual es una característica fundamental en los matemáticos, ser curiosos y siempre querer aprender más, estar en una búsqueda constante de la verdad. Esto sumado al hecho de ser Premio Nacional de Ciencias Exactas, hace que para mi sea un honor haber realizado la tesis con usted. A Jorge San Martín quién fue mi profesor co-guía, tuvimos intensas jornadas de trabajo; aprendí demasiado de usted, del arte de programar (y del gusto por el café también). Gracias por todo profesor, por la dedicación, la ayuda, por el financiamiento. A Raúl Gormaz, quién me acogió en su oficina para hacer mi última práctica, al principio eso me ponía incómodo, pues quería pasar lo más desapercibido posible para no interrumpirlo en su trabajo, pero con el tiempo descubrí que no había nada que temer, es una persona de una calidad humana increíble, siempre dispuesto a ayudar y responder dudas, con el tiempo ese miedo se convirtió en gusto por ir a trabajar, pese a que fuera verano.
Y por último, a mi familia: mi madre, mi padre, mi hermana, mi tía y mis dos ahijados. Sé que quizás ustedes hubieran preferido que no estudiara tanto, pero comprenden que mi pasión es la matemática, que disfruto lo que hago y respetan esa decisión. Gracias por haberme apoyado en esta travesía, estoy tremendamente orgulloso de la familia que tengo, que se esforzaron hasta el final para que yo me pudiese titular, siendo jubilados y aún así trabajando, pues la pensión de ambos no alcanzaba para mantenernos. Si no fuese por ustedes, los principios y educación que me dieron no lo habría conseguido.

Tabla de Contenido

Introducción **1**

1. Preliminares **2**

2. Formulación Variacional de Stokes **8**

3. Marea Roja **17**
3.1. Biología . . . 17
3.2. Modelo Matemático . . . 18
3.3. Campo de velocidades . . . 19
3.4. Bahía Quellón . . . 19
3.5. Diferencia entre oleaje y marea . . . 20

4. Ecuaciones Discretizadas **24**
4.1. Ecuaciones discretizadas de Navier-Stokes . . . 24
4.1.1. Condiciones de borde en Navier-Stokes para la marea . . . 33
4.1.2. Solución del sistema lineal: MINRES . . . 34
4.1.3. Discusión sobre el residuo en MINRES . . . 41
4.2. Ecuación discretizada de convección-difusión . . . 43
4.2.1. Condiciones de borde ecuación de convección-difusión . . . 44
4.2.2. Método de Jacobi . . . 45
4.3. Comando mldivide . . . 46

5. OpenFOAM **49**
5.1. Estructura de OpenFOAM . . . 49
5.2. Pre-procesado: generación de la malla . . . 50
5.3. Solvers . . . 60
5.3.1. Condiciones de Borde . . . 63
5.4. Shallow Water Equations . . . 64
5.4.1. Deducción de las Shallow Water Equations . . . 65
5.4.2. Creación de Solver en OpenFOAM . . . 68
5.4.3. Condiciones de borde . . . 69
5.5. scalarTransportFoam . . . 71

6. Resultados **73**
6.1. MATLAB . . . 73
6.1.1. Resolución de las ecuaciones de Navier-Stokes . . . 73

6.1.2. Resolución de la ecuación de convección-difusión 77
6.2. OpenFOAM 82
6.2.1. MareaRojaPimpleFoam 82
6.2.2. Velocidad con OpenFOAM y concentración con MATLAB 91
6.2.3. hTotalshallowWaterFoamConNroMaximoCourant 94

Conclusión **97**

Bibliografía **99**

Índice de Tablas

4.1. Tabla resumen condiciones de borde en Navier-Stokes implementadas en MATLAB. 34
4.2. Tabla resumen condiciones de borde para la ecuación de convección-difusión de microalgas. 45
4.3. Comparación de resolución sistemas lineales para Stokes estacionario. 46
4.4. Comparación de resolución sistemas lineales para la ecuación de convección-difusión. 48

5.1. Variación en cada coordenada de los puntos de la Bahía Quellón. 52
5.2. Caras internas y prismas vecinos del prisma con numeración 3, antes de aplicar el reordenamiento 55
5.3. Caras internas y prismas vecinos del prisma con numeración 3, luego de aplicar el reordenamiento. 55
5.4. Tabla resumen condiciones de borde Navier-Stokes en OpenFOAM. 64
5.5. Tabla resumen condiciones de borde en Shallow Water. 69

Índice de Ilustraciones

2.1. Dominio octopus 8

3.1. Ciclo de vida de Alexandrium Catenella. 17
3.2. Bahía Quellón vista desde arriba con Google Earth. 20
3.3. Ilustración de la fuerza de mareas. 21
3.4. Tablas de las mareas en Puerto Quellón desde el dia 04 de Febrero del 2020 al 09 de Febrero. 23
3.5. Gráficos en MATLAB a partir de la tabla de mareas leída desde internet. . . 23

4.1. Gráfico Tiempo de ejecución MINRES (usando máximos de iteraciones 1000,2000,3000,4000,5000) vs Residuos. 47
4.2. Gráficos de desempeño del método de Jacobi. 48

5.1. Estructura de OpenFOAM. Fuente: OpenFOAM v7 User Guide, https://cfd.direct/openfoam/user-guide. 49
5.2. Case en OpenFOAM. Fuente: OpenFOAM v7 User Guide, https://cfd.direct/openfoam/user-guide. 50
5.3. Primeras dieciséis filas de la matriz de puntos del dominio. La matriz de puntos tiene dimensiones 54176 filas y 4 columnas. 52
5.4. Primeras dieciocho filas de la matriz de triprismas del dominio. La matriz de puntos tiene dimensiones 76416 filas y 6 columnas. 52
5.5. Ejemplo de los tres niveles de prismas, para las primeras 3 filas de la matriz de prismas. 53
5.6. Primeras dieciocho filas de la matriz de caras del dominio. La matriz de caras completa (sin considerar los volúmenes de dimensiones muy pequeñas) tiene 381705 filas y 4 columnas. Las caras de tres nodos, reciben un cero en la cuarta columna. 54
5.7. Generación de la malla en formato OpenFOAM. 56
5.8. Dominio con sus volúmenes finitos. 57
5.9. Dominio visto con Paraview. 58
5.10. Dominio escalado y malla de volúmenes finitos. 59
5.11. Esquema de la función h distancia del fondo a la superficie y h_0 distancia del sistema de referencia al fondo. 65
5.12. Condición de borde en Mar-Mar: nivel del mar sinusoidal $h_{Total}(x,t) = h_{Total}(x,0) + Asen(\omega t)$, con $A = 2m$, $T = 12hrs$ el período de marea. 69
5.13. Condición de borde en Mar-Mar: nivel del mar $h_{Total}(x,t) = h_{Total}(x,0) + Acos(\omega t)$, con $A = 2m$, $T = 12hrs$ el período de marea. 70

5.14. Batimetría del dominio h_0 y la altura de la columna de agua h. 70

6.1. Patrón de dispersividad matriz de la ecuación de Navier-Stokes discretizada usando volúmenes finitos. 74
6.2. Magnitud de la velocidad de marea, obtenida de resolver las ecuaciones de Navier-Stokes programada en MATLAB. 75
6.3. Gráfico tiempo de ejecución de la ecuación de Navier-Stokes implementada en MATLAB. 75
6.4. Gráfico del residuo de la ecuación de Navier-Stokes discretizada implementada en MATLAB, usando la velocidad calculada con MATLAB para una viscosidad $\nu = 0{,}01$. 76
6.5. Gráfico del Número de Courant obtenido con MATLAB $\nu = 0{,}01$. 76
6.6. Concentración inicial: mancha circular. 77
6.7. Patrón de dispersividad matriz de la ecuación de convección-difusión discretizada usando volúmenes finitos. 78
6.8. Magnitud de la concentración de microalgas, obtenida de la ecuación de convección-difusión programada en MATLAB. 79
6.9. Gráfico tiempo de ejecución de la ecuación de convección-difusión implementada en MATLAB. 80
6.10. Gráfico del residuo de convección-difusión implementada en MATLAB, usando la velocidad calculada con MATLAB para una viscosidad $\nu = 0{,}01$. 80
6.11. Gráfico del máximo y mínimo de concentración en alguna celda del dominio en cada tiempo, obtenido con MATLAB y usando la velocidad calculada con MATLAB para una viscosidad $\nu = 1$. 81
6.12. Tiempo discretizado [horas] vs Número de Peclet. 81
6.13. Magnitud de la velocidad de la marea obtenida por las ecuaciones de Navier-Stokes. 82
6.14. Magnitud de la concentración de la marea obtenida por la ecuación de convección-difusión. 83
6.15. Gráfico tiempo de ejecución del solver *MareaRojaPimpleFoam*. 84
6.16. Gráfico del Número de Courant obtenido con el solver *MareaRojaPimpleFoam* para una viscosidad $\nu = 0{,}01$. 84
6.17. Residuos finales obtenidos por el solver *MareaRojaPimpleFoam*. 85
6.18. Gráfico del máximo y mínimo de concentración en alguna celda del dominio en cada tiempo, obtenido con el solver *MareaRojaPimpleFoam* para una viscosidad $\nu = 0{,}01$. 86
6.19. Magnitud de la velocidad de la marea obtenida por las ecuaciones de Navier-Stokes. 87
6.20. Concentración de la marea obtenida por la ecuación de convección-difusión. . 88
6.21. Gráfico tiempo de ejecución del solver *MareaRojaPimpleFoam*, con viscosidad $\nu = 1$. 89
6.22. Gráfico del Número de Courant obtenido con el solver *MareaRojaPimpleFoam* para una viscosidad $\nu = 1$. 89
6.23. Residuos finales obtenidos por el solver *MareaRojaPimpleFoam* para una viscosidad $\nu = 1$. 90

6.24. Gráfico del máximo y mínimo de concentración en alguna celda del dominio en cada tiempo, obtenido con el solver *MareaRojaPimpleFoam* para una viscosidad $\nu = 1$. 91
6.25. Concentración de la marea obtenida por la ecuación de convección-difusión. . 93
6.26. Gráfico tiempo de ejecución de convección-difusión resuelta en MATLAB y usando la velocidad calculada con *pimpleFoam* para una viscosidad $\nu = 1$. . . 93
6.27. Gráfico del residuo de convección-difusión resuelta en MATLAB y usando la velocidad calculada con *pimpleFoam* para una viscosidad $\nu = 1$. 93
6.28. Gráfico del máximo y mínimo de concentración en alguna celda del dominio en cada tiempo, obtenido con MATLAB y usando la velocidad calculada con *pimpleFoam* para una viscosidad $\nu = 1$. 94
6.29. Nivel de la marea h_{Total} obtenido por las ecuaciones de Shallow Water. . . . 94
6.30. Dominio evolutivo, con la elevación de la superficie obtenido en Shallow Water. 95
6.31. Velocidad horizontal obtenida por las ecuaciones de Shallow Water. 95

Introducción

La marea roja es el incremento masivo de microoalgas en el mar, cuyos efectos pueden llegar a ser perjudiciales tanto para la salud del humano como para la economía. Su nombre se debe a que las Floraciones Algales (*Bloom*) pueden llegar a teñir el agua del mar de diversos colores, en que el más común es rojo. Esta coloración del agua se debe a que estas microalgas poseen pigmentos. Más generalmente (y correctamente) se conoce como *Floraciones Algales Nocivas* (FAN), puesto que estas floraciones masivas se pueden dar sin teñir agua, pero sí siendo nocivas.
Las FAN se pueden clasificar en tóxicas y no tóxicas.

- *FAN no tóxicas*, son aquellas floraciones que dado la gran cantidad de microoalgas, acaparan demasiado oxígeno, generando mortalidad en peces y demases de la vida marina.
- *FAN tóxicas*, son aquellas floraciones en que las algas emiten sustancias tóxicas y como algunos moluscos se alimentan de estas microoalgas, puede ser peligroso consumir mariscos en estas zonas de marea roja para los humanos.

Algunos mariscos que concentran estas toxinas son los choritos, almejas, machas, locos y cholgas, por citar a los más conocidos en la cultura popular. El gran problema de la marea roja, es que no hay forma de determinar si un marisco está contaminado con estas toxinas a simple vista, no se aprecia un cambio en su color o olor, sólo a través de un análisis de laboratorio se puede estudiar el nivel de toxicidad del marisco. Lo anterior motiva este trabajo de tesis; realizar un modelo que pueda predecir los episodios de marea roja, sin la necesidad de estudiar cada marisco, sino que estudiar localmente la concentración de microalgas de la marea roja en cada región del mar.

Esta tesis está organizada como sigue: **Capítulo 1** introduce los preliminares teóricos, necesarios para lo que sigue en esta tesis. **Capítulo 2** estudia la formulación variacional de un problema de Navier-Stokes. **Capítulo 3** explica la parte biológica de la marea roja estudiando la *Alexandrium catenella*, introduce el modelo matemático y el dominio del problema. En el **Capítulo 4** se discuten las discretizaciones e implementaciones computacionales en MATLAB y C++. En el **Capítulo 5** se explica el uso del software OpenFOAM y qué solvers se utilizan. Finalmente en el **Capítulo 6** se realizan las simulaciones y analizan los resultados obtenidos.

Capítulo 1

Preliminares

En este capítulo se estudian los fundamentos matemáticos de las ecuaciones de Stokes y Navier-Stokes, en particular los espacios funcionales asociados a los problemas y teoremas de existencia de solución. Se basa en [6], sin embargo [9] también es una buena referencia.
A lo largo de todo este capítulo se trabaja con la hipótesis $\Omega \subset \mathbb{R}^N$ un abierto, acotado y de frontera Lipschitz.
Primero, es necesario introducir el espacio de las distribuciones (vectoriales) a divergencia nula

Definición 1.1 $\mathcal{V} := \{\phi \in \mathcal{D}(\Omega)^N / \mathrm{div}(\phi) = 0\}$.

Sin embargo el espacio anterior es demasiado regular, lo cual motiva a definir el espacio de las funciones a traza nula y divergencia nula,

Definición 1.2 $V := \{v \in H_0^1(\Omega)^N / \mathrm{div}(v) = 0\}$.

Se dota a $H_0^1(\Omega)$ de la norma $|u|_{1,\Omega} := \|\nabla u\|_{0,\Omega}$. Claramente V es sev cerrado de $H_0^1(\Omega)^N$, entonces por proyección sobre un sev cerrado en un espacio de Hilbert, se tiene que $H_0^1(\Omega)^N = V \oplus V^\perp$ donde $V^\perp = \{u \in H_0^1(\Omega)^N / (\nabla u, \nabla v) = 0, \quad \forall v \in V\}$.

Definición 1.3 $L_0^2(\Omega) := \left\{ p \in L^2(\Omega) / \int_\Omega p \mathrm{d}x = 0 \right\}$.

Proposición 1.4 *$L_0^2(\Omega)$ se identifica isométricamente con $L^2(\Omega)/\mathbb{R}$.*

DEM: Sea $p \in L^2_0(\Omega)$, entonces

$$\begin{aligned}
\|\dot{p}\|^2 &= \inf_{c\in\mathbb{R}} \|p+c\|^2_{0,\Omega} \\
&= \inf_{c\in\mathbb{R}} \|p\|^2_{0,\Omega} + 2c\cancelto{0}{\int_\Omega p \mathrm{d}x} + c^2 |\Omega| \\
&= \|p\|^2_{0,\Omega} + |\Omega| \cancelto{0}{\inf_{c\in\mathbb{R}} c^2} \\
&= \|p\|^2_{0,\Omega}
\end{aligned}$$

□

Definición 1.5 $V^0 = \{y \in H^{-1}(\Omega)^N / \langle y, \phi\rangle = 0, \quad \forall \phi \in V\}$

Obs: NO confundir con el ortogonal de V, que se denota $V^\perp$ el cual es subespacio de $H^1_0(\Omega)^N$, mientras que $V^\perp$ está en el dual de $H^1_0(\Omega)^N$.

El siguiente resultado es crucial y en primera instancia pasa desapercibido, pues es probar cerradura que generalmente se hace por sucesiones y no presenta mayor dificultad, sin embargo, ¿cómo probar que el límite de la sucesión que se toma es el gradiente de una función en $L^2(\Omega)$?. Revisar [6]o [9] para ver una demostración en detalle.

Proposición 1.6 *El operador gradiente definido de $L^2(\Omega)$ en $H^{-1}(\Omega)$ es a imagen cerrada.*

Teorema 1.7 *Si $f \in H^{-1}(\Omega)^N$ tal que $\forall \phi \in V$, $\langle f, \phi\rangle = 0$, entonces existe $p \in L^2(\Omega)$ tal que*

$$f = \nabla p$$

Además, si Ω es conexo, entonces p es único en $L^2(\Omega)/\mathbb{R}$.

DEM: En primer lugar, notar que la hipótesis puede verse de la siguiente forma:

$$\begin{aligned}
\forall v \in V, \langle \varphi, v\rangle = 0 &\iff \varphi \in V^0 \\
&\iff \varphi \in Ker(\mathrm{div})^0 \text{ pues } V = Ker(\mathrm{div})
\end{aligned}$$

y se quiere probar que $\varphi \in Im(\nabla)$. Luego, es suficiente probar que $Ker(\mathrm{div})^0 = Im(\nabla)$. Para ello, la idea es usar el Teorema de la imagen cerrada.
Hay que ver que el adjunto de $\mathrm{div} \in \mathcal{L}(H^1_0(\Omega)^N, L^2(\Omega))$ es el operador $-\nabla \in \mathcal{L}(L^2(\Omega), H^{-1}(\Omega)^N)$. Por representación de Riesz, $L^2(\Omega)$ se identifica con su dual y $H^1_0(\Omega)^N$ con $H^{-1}(\Omega)^N$. Entonces,

$$\begin{aligned}
\langle u, -\nabla p\rangle_{H^1_0(\Omega), H^{-1}(\Omega)} &= -\sum_{\mathrm{i}=1}^N \int_\Omega u_\mathrm{i} \frac{\partial p}{\partial x_\mathrm{i}} \\
&= \sum_{\mathrm{i}=1}^N \int_\Omega p \frac{\partial u_\mathrm{i}}{\partial x_\mathrm{i}} - \cancelto{0}{\int_{\partial\Omega} u_\mathrm{i} p n_\mathrm{i} \mathrm{d}s} \text{ pues } u \in H^1_0(\Omega) \\
&= \int_\Omega p \mathrm{div}(u) \mathrm{d}x \\
&= \langle p, \mathrm{div}(u)\rangle_{L^2(\Omega), L^2(\Omega)}
\end{aligned}$$

Además, por proposición anterior, $Im(\nabla)$ es cerrado en $H^{-1}(\Omega)^N$, y por Teorema de la imagen cerrada, se concluye que $Ker(\mathrm{div})^0 = Im(\nabla)$. □

Recordar que operador gradiente es lineal continua de $L^2(\Omega)$ en $H^{-1}(\Omega^N)$. El siguiente resultado establece a qué subespacios restringirse para obtener que el gradiente es una biyección.

Corolario 1.8 *Sea $\Omega \subset \mathbb{R}^N$. Entonces,*

1. *El operador gradiente es un isomorfismo de $L^2_0(\Omega)$ en V^0*
2. *El operador divergencia es un isomorfimos de $V^\perp$ en $L^2_0(\Omega)$*

Dem:

1. Primero la nyectividad. Sea $p \in L^2_0(\Omega) \cap Ker(\nabla)$. Entonces $\nabla p = 0$ y de la conexidad de Ω, se obtiene que p es constante ctp en Ω y como $p \in L^2_0(\Omega)$, debe tener media nula, luego

$$\begin{aligned} 0 &= \int_\Omega p \mathrm{d}x \\ &= p\,|\Omega| \text{ pues p constante ctp en } \Omega \end{aligned}$$

   Es decir, $Ker(\nabla) = \{0\}$.
   Para probar la sobreyectivad, sea $f \in V^0$ y por Teorema 1.7, $V^0 = Im(\nabla)$, luego existe un único $p \in L^2(\Omega)/\mathbb{R}$ tal que

$$f = \nabla p$$

   Como $L^2_0(\Omega)$ se identifica isométricamente con $L^2(\Omega)/\mathbb{R}$ se concluye.

2. Aquí la idea es razonar usando lo probado en el primer apartado, junto a que $-\nabla$ es el operador adjunto de div. Entonces, div es un isomorfismo de $(V^0)'$ en $(L^2_0(\Omega))' = L^2_0(\Omega)$. Bastará probar que $(V^0)'$ se identifica con $V^\perp$.
   Notar que para cualquier función $f \in V^0 \subset H^{-1}(\Omega)^N$, basta restringirse a $V^\perp$ y así $f|_{V^\perp} \in (V^\perp)'$.
   Sea $g \in (V^\perp)'$, se busca construir $\tilde{g} \in H^{-1}(\Omega)$ tal que $\forall v \in V$,

$$\langle \tilde{g}, v \rangle = 0$$

   Definiendo $\tilde{g}$ de a siguiente forma: $\forall u \in H^1_0(\Omega)$,

$$\langle \tilde{g}, u \rangle = \langle g, P_{V^\perp}(u) \rangle$$

   Si $u \in V$, entonces $P_{V^\perp}(u)$ y por lo tanto,

$$\begin{aligned} \langle \tilde{g}, u \rangle &= \langle g, P_{V^\perp}(u) \rangle \\ &= 0 \end{aligned}$$

   Queda probar que es una isometría, es decir, $\|\tilde{g}\|_{-1,\Omega} = \|g\|_{(V^\perp)'}$.

Se tiene que

$$\begin{aligned}
\|\tilde{g}\|_{-1,\Omega} &= \sup_{u\in H_0^1(\Omega)} \frac{|\langle \tilde{g}, u\rangle|}{|u|_{1,\Omega}} \\
&= \sup_{u\in H_0^1(\Omega)} \frac{|\langle \tilde{g}, P_{V^\perp}(u)\rangle|}{|u|_{1,\Omega}} \\
&\geq \sup_{u\in V^\perp} \frac{|\langle \tilde{g}, P_{V^\perp}(u)\rangle|}{|u|_{1,\Omega}} \\
&= \sup_{u\in V^\perp} \frac{|\langle \tilde{g}, u\rangle|}{|u|_{1,\Omega}} \\
&= \|g\|_{(V^\perp)'}
\end{aligned}$$

La otra desigualdad viene de Cauchy-Schwarz y de que la proyección sobre un sev cerrado es 1-Lipschitz lineal. En efecto,

$$\begin{aligned}
\|\tilde{g}\|_{-1,\Omega} &= \sup_{u\in H_0^1(\Omega)} \frac{|\langle \tilde{g}, P_{V^\perp}(u)\rangle|}{|u|_{1,\Omega}} \\
&\leq \sup_{u\in H_0^1(\Omega)} \frac{\|g\|_{(V^\perp)'}\,|P_{V^\perp}(u)|_{1,\Omega}}{|u|_{1,\Omega}} \\
&\leq \|g\|_{(V^\perp)'}
\end{aligned}$$

□

Proposición 1.9 *Si además que Ω es conexo.*
Para cada $u_0 \in H^{\frac{1}{2}}(\Gamma)$ tal que $\int_\Gamma u_0 \cdot n \mathrm{d}s = 0$, entonces existe una función $g \in H^1(\Omega)$, tal que

$$\begin{cases} \mathrm{div}(g) = 0 & \text{en } \Omega \\ g = u_0 & \text{sobre } \Gamma \end{cases}$$

Dem: Como $u_0 \in H^{\frac{1}{2}}(\Gamma) = \gamma_0(H^1(\Omega))$, entonces existe $w \in H^1(\Omega)$ tal que

$$w = u_0 \text{ sobre } \Gamma$$

Notando que

$$\begin{aligned}
0 &= \int_\Gamma u_0 \cdot n \mathrm{d}s \\
&= \int_\Gamma w \cdot n \mathrm{d}s \\
&= \int_\Omega \mathrm{div}(w) \mathrm{d}x \quad \text{por Teorema de la divergencia}
\end{aligned}$$

Entonces $\mathrm{div}(w) \in L^2_0(\Omega)$. Usando que el operador divergencia es un isomorfismo de $V^\perp$ en $L^2_0(\Omega)$, existe $v \in V^\perp$ tal que $\mathrm{div}(v) = \mathrm{div}(w)$.
Se define $g := w - v$, que por lo anterior es a divergencia nula. Además, $g = w$ sobre Γ pues $v \in H^1_0(\Gamma)$. □

Teorema 1.10 *Si $f \in H^{-1}(\Omega)^N$ tal que $\forall \phi \in \mathcal{V}$, $\langle f, \phi \rangle = 0$, entonces existe $p \in L^2(\Omega)$ tal que*

$$f = \nabla p$$

Además, si Ω es conexo, entonces p es único en $L^2(\Omega)/\mathbb{R}$.

Para probar el Teorema hace falta un lema previo, el cual se omite su demostración debido a la complejidad técnica pero proviene de un Teorema general de espacios de Banach (Peetre-Tartar). La demostración se puede hallar [8].

Lema 1.11 *Sea Ω es conexo. Si $p \in L^2_{loc}(\Omega)$ y $\nabla p \in H^{-1}(\Omega)$, entonces $p \in L^2(\Omega)$*

Ahora sí se puede probar el Teorema enunciado.

DEM: Sin perdida de generalidad se puede suponer que Ω es conexo, pues si no, se puede trabajar sobre cada componente conexa. Por el Lema 1.11, basta probar que existe $p \in L^2_{loc}(\Omega)$ tal que $f = \nabla p \in H^{-1}(\Omega)^N$.
Como Ω es abierto, acotado, conexo y con borde Lipschitz, existe una secuencia creciente $(\Omega_m)_{m\geq 1}$ de abiertos, acotados, conexos y de borde Lipschitz tal que

$$\overline{\Omega}_m \subset \Omega \text{ y } \Omega = \bigcup_{m\geq 1} \Omega_m$$

La idea es usar el Teorema 1.7, sin embargo, como $\mathcal{V}$ está estrictamente contenido V no se puede hacer en Ω, pero si sobre los Ω_m.
Claramente $f|_{\Omega_m} \in H^{-1}(\Omega_m)^N$.
Sea $u_m \in V_m := \{v \in H^1_0(\Omega_m)^N / \mathrm{div}(v) = 0 \text{ en } \Omega_m\}$. Probemos que $\langle f|_{\Omega_m}, u_m \rangle = 0$. Hay que usar la hipótesis, pero no necesariamente se tiene que $u_m \in \mathcal{V}$, así que es necesario regularizar u_m.
Para $\varepsilon > 0$, sea ρ_ε una función en $\mathcal{D}(\mathbb{R}^N)$ el núcleo regularizador, que cumple

- $\rho_\varepsilon \geq 0$
- $\displaystyle\int_{\mathbb{R}^N} \rho_\varepsilon \mathrm{d}x = \int_{B(0,\varepsilon)} \rho_\varepsilon \mathrm{d}x = 1$

El núcleo regularizador cumple que cuando $\varepsilon \to 0$, $\rho_\varepsilon * u_m \to u_m$ en $L^2(\Omega)$. Más aún, como $u_m \in H^1(\Omega)^N$, se tiene que $\rho_\varepsilon * u_m \to u_m$ en $H^1(\Omega)^N$.
Por otro lado, $\rho_\varepsilon * v \in \mathcal{D}(\mathbb{R}^N)$ pues $\rho_\varepsilon \in \mathcal{D}(\mathbb{R}^N)$, entonces por propiedad de la convolución

$$\begin{aligned} \mathrm{div}(\rho_\varepsilon * u_m) &= \rho_\varepsilon * \mathrm{div}(u_m) \\ &= 0 \end{aligned}$$

por lo tanto $\phi = \rho_\varepsilon * u_m \in \mathcal{V}$, y por hipótesis

$$\begin{aligned} \langle f, u_m \rangle &= \lim_{\varepsilon \to 0} \langle f, \rho_\varepsilon * u_m \rangle \\ &= 0 \end{aligned}$$

Del Teorema 1.7, existe $p_m \in L^2(\Omega_m)/\mathbb{R}$ tal que

$$f|_{\Omega_m} = \nabla p_m \tag{1.1}$$

Como los $(\Omega_m)_{m\geq 1}$ son crecientes, es decir, $\Omega_m \subset \Omega_{m+1}$, se tiene

$$f|_{\Omega_m} = f|_{\Omega_{m+1}} \text{ en } \Omega_m$$

y por (1.1), $\nabla p_m = \nabla p_{m+1}$ en Ω_m, es decir,

$$\nabla(p_m - p_{m+1}) = 0 \text{ en } \Omega_m$$

entonces, $p_m - p_{m+1}$ es constante en Ω_m, cuyo valor se denota $C_m := p_m - p_{m+1}$ en Ω_m y puesto que p_m es única salvo constante aditiva, luego tomando $p_m + C_m$ en lugar de p_m se obtiene que $p_m = p_{m+1}$ en Ω_m, es decir,

$$p_m = p_{m+1} \text{ en } \Omega_m \quad \forall m \geq 1$$

Así, basta definir $p := \lim\limits_{m\nearrow\infty} p_m \in L^2(\bigcup\limits_{m\geq 1} \Omega_m)/\mathbb{R} = L^2(\Omega)/\mathbb{R} = L^2_{loc}(\Omega)$ que cumple

$$f = \nabla p \text{ en } \Omega$$

. □

Capítulo 2

Formulación Variacional de Stokes

En este capítulo se estudia la relación entre constantes para definir únicamente la presión cuando el dominio está compuesto por dos abiertos suaves y un tubo que los conecta, basándose en los artículos [2], [3] y [4], donde se usan los resultados sobre espacios funcionales del problema de Stokes.

Sea el siguiente dominio particionado $\Omega_\varepsilon = \Omega_1^\varepsilon \cup \Sigma_1^\varepsilon \cup O_{\varepsilon,L} \cup \Omega_2^\varepsilon \cup \Sigma_2^\varepsilon \subset \mathbb{R}^2$*, que corresponde a un tubo* $O_{\varepsilon,L}$ *de largo* L *y altura* ε*, que conecta dos dominios suaves* Ω_1^ε *y* Ω_2^ε*.*

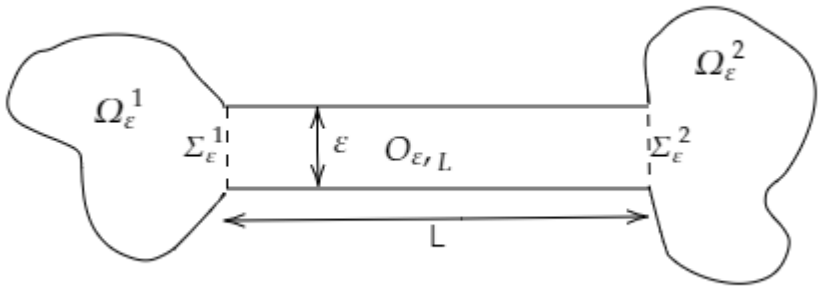


Figura 2.1: Dominio octopus

Se define el espacio $V^\varepsilon := \{v \in H_0^1(\Omega_\varepsilon)^2 / \mathrm{div}(v) = 0 \text{ en } \Omega_\varepsilon\}$. El borde del dominio se denota $\Gamma_\varepsilon = \partial\Omega_\varepsilon$.
El propósito de este capítulo es estudiar el siguiente problema de Stokes con condición de borde no homogénea:

$$(S)\begin{cases} -\nu\Delta u + \nabla p = f & \text{en } \Omega_\varepsilon \\ \mathrm{div}(u) = 0 & \text{en } \Omega_\varepsilon \\ u = u_0 & \text{sobre } \Gamma_\varepsilon \end{cases}$$

con $u_0 \in H^{\frac{1}{2}}(\Gamma_\varepsilon)$ que cumple la condición de compatibilidad, es decir, $\displaystyle\int_{\Gamma_\varepsilon} u_0 \cdot n ds = 0$.

Hay mucha teoría desarollada para el estudio de este problema, usando los resulados exhibidos en el capítulo de preliminares. De acuerdo a la Proposición 1.9, existe $g \in H^1(\Omega_\varepsilon)$ tal que

$$\begin{cases} \mathrm{div}(g) &= 0 \quad \text{en } \Omega_\varepsilon \\ g &= u_0 \quad \text{sobre } \Gamma_\varepsilon \end{cases}$$

Haciendo dicho relevo, el problema se puede estudiar variacionalmente como

$$(FVS) \begin{cases} \text{Encontrar } v \in H^1(\Omega_\varepsilon)^2 \text{ tal que} \\ (v-g) \in V^\varepsilon \\ \displaystyle\int_{\Omega_\varepsilon} \nabla u : \nabla v \mathrm{d}x = \int_{\Omega_\varepsilon} f \cdot v \mathrm{d}x, \quad \forall v \in V^\varepsilon \end{cases}$$

El inconveniente de este problema es que si bien la teoría es ampliamente conocida, resolverlo numéricamente es muy costoso, debido al largo L de $O_{\varepsilon,L}$. Esto motiva a definir el siguiente conjunto:

Definición 2.1 $W^\varepsilon := \left\{ w \in V^\varepsilon / w(x_1,x_2) = \begin{pmatrix} w_1(x_2) \\ 0 \end{pmatrix} \ \textit{con } (x_1,x_2) \in O_{\varepsilon,L} \right\}$

Proposición 2.2 *W^ε es sev cerrado en $H_0^1(\Omega_\varepsilon)^2$. Por lo tanto, W^ε es un espacio de Hilbert dotado del producto interno usual de $H_0^1(\Omega_\varepsilon)^2$.*

Dem: Sea $(w_n)_n \subset W^\varepsilon$ tal que $w_n \to w$ en $H_0^1(\Omega_\varepsilon)^2$. En particular,
$\mathrm{div}(w_n) \to \mathrm{div}(w)$ en $L^2(\Omega_\varepsilon)$ Como $w_n \in W^\varepsilon$, $\mathrm{div}(w_n) = 0$ y por lo tanto $\mathrm{div}(w) = 0$.
Queda ver que w no depende de x_1 en $0_{\varepsilon,L}$. Del hecho que $w_n \to w$ en $L^2(\Omega_\varepsilon)^2$, se obtiene que existe una subsucesión (w_{n_k}) tal que $w_{n_k} \to w$ ctp en Ω_ε. En particular, $w_{n_k} \to w$ ctp en $O_{\varepsilon,L}$ y puesto que $w_{n_k}(x_1,x_2) = \begin{pmatrix} w_{n_k,1}(x_2) \\ 0 \end{pmatrix}$ en $O_{\varepsilon,L}$, se obtiene que

$$w(x_1,x_2) = \begin{pmatrix} w_1(x_2) \\ 0 \end{pmatrix} \text{ ctp en } O_{\varepsilon,L}. \quad \square$$

La idea de introducir el subespacio W^ε es simplificar el problema (FVS). Así, se plantea el siguiente problema

$$(FVW) \begin{cases} \text{Encontrar } u \in H^1(\Omega_\varepsilon)^2 \text{ tal que} \\ (u-g) \in W^\varepsilon \\ \displaystyle\int_{\Omega_\varepsilon} \nabla u : \nabla w \mathrm{d}x = \int_{\Omega_\varepsilon} f \cdot w \mathrm{d}x, \quad \forall w \in W^\varepsilon \end{cases}$$

Teorema 2.3 *El problema variacional (FVW) posee solución y esta es única.*

Dem: Definimos $L : W^\varepsilon \to \mathbb{R}$ dado por $L(w) := \displaystyle\int_{\Omega_\varepsilon} f \cdot w - \nabla g : \nabla w \mathrm{d}x$ y
$a : W^\varepsilon \times W^\varepsilon \to \mathbb{R}$ dado por $a(z,w) = \displaystyle\int_{\Omega_\varepsilon} \nabla z : \nabla w \mathrm{d}x$. Claramente L es lineal y $a(\cdot,\cdot)$ es una forma bilineal pues la integral y derivada son lineales.

- L es continua: Sea $w \in W^{\varepsilon}$,

$$\begin{aligned}
|L(w)| &\leq \int_{\Omega_\varepsilon} |f \cdot w - \nabla g : \nabla w| \, \mathrm{d}x \\
&\leq \int_{\Omega_\varepsilon} |f \cdot w| + |\nabla g : \nabla w| \, \mathrm{d}x \\
&\leq \int_{\Omega_\varepsilon} |f| \, |w| + \sum_{j,k=1}^{2} \left| \frac{\partial g_k}{\partial x_j} \right| \left| \frac{\partial w_k}{\partial x_j} \right| \mathrm{d}x \\
&\leq \|f\|_{0,\Omega_\varepsilon} \|w\|_{0,\Omega_\varepsilon} + \sum_{k,j=1}^{2} \left\| \frac{\partial g_k}{\partial x_j} \right\|_{0,\Omega_\varepsilon} \left\| \frac{\partial w_k}{\partial x_j} \right\|_{0,\Omega_\varepsilon} \\
&\leq \|f\|_{0,\Omega_\varepsilon} \|w\|_{1,\Omega_\varepsilon} + \|\nabla g\|_{0,\Omega_\varepsilon} \|w\|_{1,\Omega_\varepsilon} \\
&\leq (\|f\|_{0,\Omega_\varepsilon} + \|\nabla g\|_{0,\Omega_\varepsilon}) \|w\|_{1,\Omega_\varepsilon}
\end{aligned}$$

- $a(\cdot, \cdot)$ es continua: Sea $z, w \in W^{\varepsilon}$,

$$\begin{aligned}
|a(z, w)| &\leq \int_{\Omega_\varepsilon} |\nabla z : \nabla w| \, \mathrm{d}x \\
&= \int_{\Omega_\varepsilon} \sum_{j,k=1}^{2} \left| \frac{\partial z_k}{\partial x_j} \right| \left| \frac{\partial w_k}{\partial x_j} \right| \mathrm{d}x \\
&\leq \sum_{j,k=1}^{2} \left\| \frac{\partial z_k}{\partial x_j} \right\|_{0,\Omega_\varepsilon} \left\| \frac{\partial w_k}{\partial x_j} \right\|_{0,\Omega_\varepsilon} \\
&\leq \|\nabla z\|_{0,\Omega_\varepsilon} \|w\|_{1,\Omega_\varepsilon} \\
&\leq \|z\|_{1,\Omega_\varepsilon} \|w\|_{1,\Omega_\varepsilon}
\end{aligned}$$

- $a(\cdot, \cdot)$ coerciva: Sea $w \in W^{\varepsilon}$,

$$\begin{aligned}
|a(w, w)| &= \int_{\Omega_\varepsilon} |\nabla w|^2 \, \mathrm{d}x \\
&\geq C \|w\|^2_{1,\Omega_\varepsilon}
\end{aligned}$$

  donde la última desigualdad es por la equivalencia de normas en $H^1_0(\Omega_\varepsilon)$ debido a la desigualdad de Poincarè.

Entonces por el Teorema de Lax-Milgram, existe un único $z \in W^{\varepsilon}$, tal que

$$a(z, w) = L(w)$$

entonces,

$$\int_{\Omega_\varepsilon} \nabla(z + g) : \nabla w \mathrm{d}x = \int_{\Omega_\varepsilon} f \cdot w \mathrm{d}x$$

Luego, basta definir $w := z + g$, pues así, $z = w - g \in W^{\varepsilon}$, tal que

$$\int_{\Omega_\varepsilon} \nabla w : \nabla w \mathrm{d}x = \int_{\Omega_\varepsilon} f \cdot w \mathrm{d}x \qquad \square$$

Surge la pregunta natural: ¿Qué problema diferencial resuelve la solución de (FVW)?

Para responder a dicha pregunta, es necesario usar un resultado tipo Teorema 1.10 (De Rham), adaptado al espacio particular W^ε.

Definición 2.4 *Se define el siguiente subespacio de* $H^1(\Omega_\varepsilon)^2$
$$Y^\varepsilon = \left\{ w \in H^1(\Omega_\varepsilon)^2 / w \cdot n = 0 \text{ sobre } \partial\Omega_\varepsilon, \mathrm{div}(w) = 0 \text{ en } \Omega_\varepsilon, w|_{O_{\varepsilon,L}} = \begin{pmatrix} w_1(x_2) \\ 0 \end{pmatrix} \right\}$$

Observación: Es claro que $W^\varepsilon \subset Y^\varepsilon$.

El siguiente lema permite caracterizar el ortogonal en $L^2(\Omega_\varepsilon)^2$ de Y^ε.

Lema 2.5

$$Y^{\varepsilon\perp} = \left\{ \varphi \in L^2(\Omega_\varepsilon)^2 \,\middle/\, \varphi = \nabla p_k \text{ en } L^2(\Omega_k^\varepsilon)^2, \int_0^L \varphi_1 \mathrm{d}x_1 = [p] \text{ en } L^2((0,\varepsilon)), p_k \in H^1(\Omega_k^\varepsilon), k = 1,2 \right\}$$

Dem: Por simplificación en la notación, se llama

$$X := \left\{ \varphi \in L^2(\Omega_\varepsilon)^2 \,\middle/\, \varphi = \nabla p_k \text{ en } L^2(\Omega_k^\varepsilon)^2, \int_0^L \varphi_1 \mathrm{d}x_1 = [p] \text{ en } \mathcal{D}'((0,\varepsilon)), p_k \in H^1(\Omega_k^\varepsilon), k = 1,2 \right\}$$

Primero se prueba que X es un sev cerrado en $L^2(\Omega_\varepsilon)^2$. Sea $\varphi^n = (\varphi_1^n, \varphi_2^n) \subset X$, tal que $\varphi^n \to \varphi$ en $L^2(\Omega_\varepsilon)^2$. Hay que ver que $\varphi \in X$.
Se tiene que $\varphi^n = \nabla p_k^n$ en $L^2(\Omega_k^\varepsilon)$, y como $\varphi^n \to \varphi$ en $L^2(\Omega_\varepsilon)^2$, entonces $\nabla p_k^n \to \varphi|_{\Omega_k^\varepsilon}$ en $L^2(\Omega_k^\varepsilon)^2$ y como el operador $\nabla \in \mathcal{L}(H^1(\Omega_k^\varepsilon), L^2(\Omega_k^\varepsilon))$ es a imagen cerrada en $L^2(\Omega_k^\varepsilon)$, entonces existe una subsucesión n_j tal que

$$p_k = \lim_{j\to\infty} p_k^{n_j} \text{ en } H^1(\Omega_k^\varepsilon), \varphi = \nabla p_k \text{ en } L^2(\Omega_k^\varepsilon)^2$$

y de la continuidad de la traza de las funciones $H^1(\Omega_\varepsilon)$, se sigue que

$$p_k^{n_j}|_{\Sigma_k^\varepsilon} \to p_k|_{\Sigma_k^\varepsilon} \text{ en } L^2((0,\varepsilon))$$

Además, como $\varphi_1^{n_j} \to \varphi_1$ en $L^2(O_{\varepsilon,L})$, se puede extraer una subsucesión la cual se seguirá denotando n_j para no sobrecargar la notación, tal que

$$\int_0^L \varphi_1^{n_j} \mathrm{d}x_1 \to \int_0^L \varphi_1 \mathrm{d}x_1 \text{ en } L^2((0,\varepsilon))$$

Pasando al límite, se obtiene que

$$\int_0^L \varphi_1 \mathrm{d}x_1 = p|_{\Sigma_2^\varepsilon} - p|_{\Sigma_2^\varepsilon} \text{ en } L^2((0,\varepsilon))$$

Por lo tanto, $\varphi \in X$. A continuación se prueba por doble inclusión que $X^\perp = Y^\varepsilon$.

- $Y^\varepsilon \subset X^\perp$: Sea $w \in Y^\varepsilon$. Sea $\varphi \in X$,

$$\begin{aligned}
(w,\varphi)_{0,\Omega_\varepsilon} &= \int_{\Omega_\varepsilon} w\cdot\varphi \mathrm{d}x \\
&= \int_{\Omega_1^\varepsilon} \nabla p_1\cdot w\mathrm{d}x + \int_{\Omega_2^\varepsilon}\nabla p_2\cdot w\mathrm{d}x + \int_{O_{\varepsilon,L}} \begin{pmatrix} w_1(x_2)\\ 0\end{pmatrix}\begin{pmatrix}\varphi_1\\ \varphi_2\end{pmatrix}\mathrm{d}x \\
&= \int_{\partial\Omega_1^\varepsilon} p_1 w\cdot n\mathrm{d}s - \int_{\Omega_1^\varepsilon} p_1 \underbrace{\mathrm{div}(w)}_{0} + \int_{\partial\Omega_2^\varepsilon} p_2 w\cdot n\mathrm{d}s - \int_{\Omega_2^\varepsilon} p_2\underbrace{\mathrm{div}(w)}_{0} + \int_{O_{\varepsilon,L}}\varphi_1 w_1 \\
&= \int_{\partial\Omega_1^\varepsilon\cap\partial\Omega_\varepsilon} p_1\underbrace{w\cdot n}_{0} + \int_{\Sigma_1^\varepsilon} p_1 w\cdot n + \int_{\partial\Omega_2^\varepsilon\cap\partial\Omega_\varepsilon} p_2\underbrace{w\cdot n}_{0} + \int_{\Sigma_2^\varepsilon} p_2 w\cdot n + \int_{O_{\varepsilon,L}}\varphi_1 w_1 \\
&= \int_{\Sigma_1^\varepsilon} p|_{\Sigma_1^\varepsilon} w_1\mathrm{d}x_2 - \int_{\Sigma_2^\varepsilon} p|_{\Sigma_2^\varepsilon} w_1\mathrm{d}x_2 + \int_0^\varepsilon w_1(x_2)\int_0^L \varphi_1(x_1,x_2)\mathrm{d}x_1\mathrm{d}x_2 \\
&= \int_0^\varepsilon w_1(x_2)\int_0^L \varphi_1(x_1,x_2)\mathrm{d}x_1\mathrm{d}x_2 - \int_0^\varepsilon w_1(x_2)[p](x_2)\mathrm{d}x_2 \\
&= 0 \quad \text{pues } \varphi\in X
\end{aligned}$$

  Notar que se pudo usar la Identidad de Green: $\forall q\in H^1(\Omega), v\in H(\mathrm{div},\Omega)$,

$$(\nabla q, v)_{0,\Omega} + (\mathrm{div}(v), q)_{0,\Omega} = \langle v\cdot n, q\rangle_{0,\partial\Omega}$$

  pues $p_k \in H^1(\Omega_k^\varepsilon)$ y $w\in H^1(\Omega_k^\varepsilon)^2$.
- $X^\perp \subset Y^\varepsilon$: Sea $w\in X^\perp$, es decir, $\forall\varphi\in X$, $(w,\varphi)_{0,\Omega_\varepsilon}=0$.
  Tomando $\varphi=\nabla p$, con $p\in H^1(\Omega_\varepsilon)$ se tiene que $\varphi\in L^2(\Omega_\varepsilon)$. Además,

$$\begin{aligned}
\int_0^L \varphi_1\mathrm{d}x_1 &= \int_0^L \frac{\partial p}{\partial x_1}\mathrm{d}x_1 \\
&= p(L,x_2) - p(0,x_2) \\
&= [p]
\end{aligned}$$

  Esto prueba que $\varphi=\nabla p\in X$ para cada $p\in H^1(\Omega_\varepsilon)$, por lo cual,

$$\begin{aligned}
0 &= (w,\nabla p)_{0,\Omega_\varepsilon} \\
&= \int_{\partial\Omega_\varepsilon} pw\cdot n\mathrm{d}s - \int_{\Omega_\varepsilon} p\mathrm{div}(w)\mathrm{d}x
\end{aligned}$$

  Luego,

$$\int_{\partial\Omega_\varepsilon} pw\cdot n\mathrm{d}s = \int_{\Omega_\varepsilon} p\mathrm{div}(w)\mathrm{d}x \tag{2.1}$$

  En particular, $\forall p\in\mathcal{D}(\Omega_\varepsilon)$, $(\mathrm{div}(w),p)_{0,\Omega_\varepsilon}=0$, esto es,

$$\mathrm{div}(w) = 0 \text{ en } \mathcal{D}'(\Omega_\varepsilon)$$

  y a fortiori esta igualdad es en $L^2(\Omega_\varepsilon)$, pues $w\in H^1(\Omega_\varepsilon)^2$. Usando esto en (2.1), se obtiene que $\forall p\in H^1(\Omega_\varepsilon), \langle w\cdot n, p\rangle_{0,\partial\Omega_\varepsilon}=0$, por lo tanto $w\cdot n=0$ en el sentido de $H^{-\frac{1}{2}}(\partial\Omega_\varepsilon)$.

Finalmente, queda probar que $w|_{O_{\varepsilon,L}}(x_1,x_2) = \begin{pmatrix} w_1(x_2) \\ 0 \end{pmatrix}$. Sea $\varphi \in X$,

$$\begin{aligned}
0 &= (w,\varphi)_{0,\Omega_\varepsilon} \\
&= \int_{\Omega_1^\varepsilon} w\cdot\nabla p_1 \mathrm{d}x + \int_{\Omega_2^\varepsilon} w\cdot\nabla p_2 \mathrm{d}x + \int_{O_{\varepsilon,L}} w\cdot\varphi \mathrm{d}x \\
&= \int_{\partial\Omega_1^\varepsilon} p_1 w\cdot n \mathrm{d}s - \int_{\Omega_1^\varepsilon} p_1 \mathrm{div}(w)\mathrm{d}x + \int_{\partial\Omega_2^\varepsilon} p_2 w\cdot n\mathrm{d}s - \int_{\Omega_2^\varepsilon} p_2 \mathrm{div}(w)\mathrm{d}x + \int_{O_{\varepsilon,L}} w\cdot\varphi\mathrm{d}x \\
&= \int_{\partial\Omega_1^\varepsilon\cap\partial\Omega_\varepsilon} p_1 w\cdot n\mathrm{d}s + \int_{\Sigma_1^\varepsilon} p_1 w\cdot n\mathrm{d}s + \int_{\partial\Omega_2^\varepsilon\cap\partial\Omega_\varepsilon} p_2 w\cdot n\mathrm{d}s + \int_{\Sigma_2^\varepsilon} p_2 w\cdot n\mathrm{d}s + \int_{O_{\varepsilon,L}} w\cdot\varphi\mathrm{d}x \\
&= \int_0^\varepsilon p|_{\Sigma_1^\varepsilon} w_1 \mathrm{d}x_2 - \int_0^\varepsilon p|_{\Sigma_2^\varepsilon} w_1 \mathrm{d}x_2 + \int_{O_{\varepsilon,L}} w\cdot\varphi\mathrm{d}x
\end{aligned}$$

Entonces, para todo $\varphi \in X$

$$\int_{O_{\varepsilon,L}} w\cdot\varphi\mathrm{d}x = \int_0^\varepsilon w_1[p]\mathrm{d}x_2 \tag{2.2}$$

$$\int_{O_{\varepsilon,L}} w\cdot\varphi\mathrm{d}x = \int_{O_{\varepsilon,L}} \begin{pmatrix} w_1(x_2) \\ 0 \end{pmatrix}\cdot\varphi\mathrm{d}x \tag{2.3}$$

Como no hay restricciones sobre φ_2 en $O_{\varepsilon,L}$ se tiene que $\varphi = \begin{pmatrix} \frac{\partial p}{\partial x_1} \\ \frac{\partial p}{\partial x_2}\mathbb{1}_{\Omega_\varepsilon\setminus O_{\varepsilon,L}} \end{pmatrix}$, con $p \in \mathcal{D}(O_{\varepsilon,L})$. Usando (2.2),

$$\begin{aligned}
0 &= \int_{O_{\varepsilon,L}} w_1 \frac{\partial p}{\partial x_1}\mathrm{d}x \\
&= -\left\langle \frac{\partial w_1}{\partial x_1}, p \right\rangle
\end{aligned}$$

Entonces,

$$\frac{\partial w_1}{\partial x_1} = 0 \text{ en } \mathcal{D}'(O_{\varepsilon,L})$$

De lo cual se deduce que $w_1(x_1,x_2) = w_1(x_2)$ en $\mathcal{D}'(O_{\varepsilon,L})$, y como $w_1(\cdot,\cdot) \in L^2(\Omega_\varepsilon)$,

$$w_1(x_1,x_2) = w_1(x_2) \text{ en } L^2(O_{\varepsilon,L}) \tag{2.4}$$

Para ver que $w_2|_{O_{\varepsilon,L}}(x_1,x_2) = 0$, se usa que (2.3) y (2.4), llegando a que

$$\int_{O_{\varepsilon,L}} w_2(x_1,x_2)\varphi_2\mathrm{d}x = 0, \quad \forall \varphi_2 \in L^2(O_{\varepsilon,L})$$

Así, $w \in Y^\varepsilon$.

En consecuencia, $X^\perp = Y^\varepsilon$ y como Y^ε es sev cerrado en $L^2(\Omega_\varepsilon)^2$, $(X^\perp)^\perp = X$.
Se concluye que $Y^{\varepsilon\perp} = X$. □

El lema anterior será particularmente útil para interpretar la solución u de (FVW). Con el mismo propósito, pero con aplicación más directa aún es el siguiente Teorema:

Teorema 2.6 *Sea $\varphi \in L^2(\Omega_\varepsilon)^2$, tal que $\forall w \in H_0^1(\Omega_\varepsilon)^2$, $\mathrm{div}(w) = 0$ en Ω_k^ε y $w|_{O_{\varepsilon,L}}(x_1, x_2) = \begin{pmatrix} w_1(x_2) \\ 0 \end{pmatrix}$, $\langle \varphi, w \rangle = 0$. Entonces, existen $p_1 \in L^2(\Omega_1^\varepsilon)$ y $p_2 \in L^2(\Omega_2^\varepsilon)$ únicas salvo constante tales que*

$$\varphi = \nabla p_1 \quad \text{en } \mathcal{D}'(\Omega_1^\varepsilon) \tag{2.5}$$

$$\varphi = \nabla p_2 \quad \text{en } \mathcal{D}'(\Omega_2^\varepsilon) \tag{2.6}$$

$$\int_0^L \varphi_1(x_1, \cdot)\mathrm{d}x_1 = [p] \quad \text{en } \mathcal{D}'((0,\varepsilon)) \tag{2.7}$$

Dem: En particular, se tiene que para cualquier $w \in \mathcal{D}(\Omega_k^\varepsilon)^2$, tal que $\mathrm{div}(w) = 0$ en Ω_k^ε con $k \in \{1, 2\}$

$$\langle \varphi|_{\Omega_k^\varepsilon}, w \rangle = 0$$

y luego en virtud del Teorema 1.7, existe $p_k \in L^2(\Omega_k^\varepsilon)$ única salvo constante tal que

$$\varphi = \nabla p_k \text{ en } \mathcal{D}'(\Omega_k^\varepsilon)$$

Más aún, como $\varphi \in L^2(\Omega_\varepsilon)^2$, se tiene a fortiori que $\nabla p_k \in L^2(\Omega_\varepsilon)^2$, es decir, $p_k \in H^1(\Omega_\varepsilon)$. Queda probar (2.7).

Sea $w \in H_0^1(\Omega_\varepsilon)^2$, tal que $\mathrm{div}(w) = 0$ en Ω_k^ε y $w|_{O_{\varepsilon,L}}(x_1, x_2) = \begin{pmatrix} w_1(x_2) \\ 0 \end{pmatrix}$,

$$\begin{aligned}
0 &= \langle \varphi, w \rangle \\
&= \int_{\Omega_1^\varepsilon} \nabla p_1 \cdot w \mathrm{d}x + \int_{\Omega_2^\varepsilon} \nabla p_2 \cdot w \mathrm{d}x + \int_0^\varepsilon \int_0^L \varphi_1(x_1, x_2) w_1(x_2) \mathrm{d}x_1 \mathrm{d}x_2 \\
&= \int_{\partial\Omega_1^\varepsilon} p_1 w \cdot n \mathrm{d}s - \int_{\Omega_1^\varepsilon} p_1 \underbrace{\mathrm{div}(w)}_{0} \mathrm{d}x + \int_{\partial\Omega_2^\varepsilon} p_2 w \cdot n \mathrm{d}s - \int_{\Omega_2^\varepsilon} p_2 \underbrace{\mathrm{div}(w)}_{0} \mathrm{d}x + \int_0^\varepsilon \int_0^L \varphi_1 w_1 \mathrm{d}x \\
&= \int_{\partial\Omega_1^\varepsilon \cap \partial\Omega_\varepsilon} p_1 \underbrace{w \cdot n}_{0} \mathrm{d}s + \int_{\Sigma_1^\varepsilon} p_1 w \cdot n \mathrm{d}s + \int_{\partial\Omega_2^\varepsilon \cap \partial\Omega_\varepsilon} p_2 \underbrace{w \cdot n}_{0} \mathrm{d}s + \int_{\Sigma_2^\varepsilon} p_2 w \cdot n \mathrm{d}s + \int_0^\varepsilon \int_0^L \varphi_1 w_1 \mathrm{d}x \\
&= \int_{\Sigma_1^\varepsilon} p|_{\Sigma_1^\varepsilon} w_1 \mathrm{d}x_2 - \int_{\Sigma_2^\varepsilon} p|_{\Sigma_2^\varepsilon} w_1 \mathrm{d}x_2 + \int_0^\varepsilon w_1(x_2) \int_0^L \varphi_1(x_1, x_2) \mathrm{d}x \\
&= \int_0^\varepsilon w_1(x_2) \int_0^L \varphi_1(x_1, x_2) \mathrm{d}x_1 \mathrm{d}x_2 - \int_0^\varepsilon w_1(x_2)[p](x_2) \mathrm{d}x_2
\end{aligned}$$

Luego,

$$\int_0^\varepsilon w_1(x_2) \int_0^L \varphi_1(x_1, x_2) \mathrm{d}x_1 \mathrm{d}x_2 = \int_0^\varepsilon w_1(x_2)[p](x_2) \mathrm{d}x_2 \tag{2.8}$$

Tomando $w_1 \in \mathcal{D}((0,\varepsilon))$ y extendiéndola a $O_{\varepsilon,L}$ como $w|_{O_{\varepsilon,L}}(x_1, x_2) = \begin{pmatrix} w_1(x_2) \\ 0 \end{pmatrix}$. Se define

la función $w := w|_{O_{\varepsilon,L}} \mathbb{1}_{\overline{O}_{\varepsilon,L}}$, la cual pertenece a $H_0^1(\Omega_\varepsilon)^2$, pues

$$\begin{aligned}
\|\nabla w\|^2_{0,\Omega_\varepsilon} &= \left\|\frac{\partial w_1}{\partial x_2}\right\|^2_{0,O_{\varepsilon,L}} \\
&= \int_0^\varepsilon \int_0^L \left|\frac{\partial w_1(x_2)}{\partial x_2}\right|^2 \mathrm{d}x_1\mathrm{d}x_2 \\
&= L\int_0^\varepsilon \left|\frac{\partial w_1(x_2)}{\partial x_2}\right|^2 \mathrm{d}x_2 \\
&< \infty \quad \text{pues } w_1 \in H_0^1((0,\varepsilon))
\end{aligned}$$

y por construcción $w = 0$ sobre $\partial\Omega_\varepsilon$.
Además, claramente $\mathrm{div}(w) = 0$ en Ω_ε, pues como sólo depende de x_2, $\mathrm{div}(w) = 0$ en $O_{\varepsilon,L}$ y es nula fuera de $\overline{O}_{\varepsilon,L}$. Por lo tanto se tiene (2.8) para cada $w_1 \in \mathcal{D}((0,\varepsilon))$, es decir,

$$\int_0^L \varphi_1(x_1,\cdot)\mathrm{d}x_1 = [p] \quad \text{en } \mathcal{D}'((0,\varepsilon)) \quad \square$$

Teorema 2.7 *Si u es solución de (FVW), entonces existe $p_k \in L^2(\Omega_k^\varepsilon)$ única salvo constante, con $k \in \{1,2\}$ tal que*

$$\begin{cases}
-\Delta u + \nabla p_1 &= f \text{ en } \mathcal{D}'(\Omega_1^\varepsilon) \\
\mathrm{div}(u) &= 0 \text{ en } \Omega_1^\varepsilon \\
u &= u_0 \text{ en el sentido de la traza sobre } \partial\Omega_\varepsilon \cap \partial\Omega_1^\varepsilon \\
u &= \begin{pmatrix} u_1 \\ g_2 \end{pmatrix} \text{ en el sentido de la traza sobre } \Sigma_1^\varepsilon
\end{cases} \tag{2.9}$$

$$\begin{cases}
-\Delta u + \nabla p_2 &= f \text{ en } \mathcal{D}'(\Omega_2^\varepsilon) \\
\mathrm{div}(u) &= 0 \text{ en } \Omega_2^\varepsilon \\
u &= u_0 \text{ en el sentido de la traza sobre } \partial\Omega_\varepsilon \cap \partial\Omega_2^\varepsilon \\
u &= \begin{pmatrix} u_1 \\ g_2 \end{pmatrix} \text{ en el sentido de la traza sobre } \Sigma_2^\varepsilon
\end{cases} \tag{2.10}$$

$$\begin{cases}
-\displaystyle\int_0^L \frac{\partial^2 u_1}{\partial x_2^2}\mathrm{d}x_1 + [p] &= \displaystyle\int_0^L f\mathrm{d}x_1 \text{ en } \mathcal{D}'((0,\varepsilon)) \\
u_1(0) &= u_0(0,0), \text{ en el sentido de la traza sobre } \partial\Omega_\varepsilon \cap (0,\varepsilon) \\
u_1(\varepsilon) &= u_0(0,\varepsilon), \text{ en el sentido de la traza sobre } \partial\Omega_\varepsilon \cap (0,\varepsilon) \\
\displaystyle\int_0^\varepsilon u_1\mathrm{d}x_2 &= -\displaystyle\int_{\partial\Omega_\varepsilon \cap \partial\Omega_1^\varepsilon} u_0 \cdot n\mathrm{d}s
\end{cases} \tag{2.11}$$

NOTACIÓN: $[p] := p|_{\Sigma_2^\varepsilon} - p|_{\Sigma_1^\varepsilon} = p_2(L,x_2) - p_1(0,x_2)$

DEM: Como $(u-g) \in V^\varepsilon$, entonces

$$\begin{aligned}
0 &= \mathrm{div}(u-g) \\
&= \mathrm{div}(u) - \cancelto{0}{\mathrm{div}(g)} \text{ pues el relevo es a la divergencia nula}
\end{aligned}$$

Luego se cumple $\mathrm{div}(u) = 0$ en $\mathcal{D}'(\Omega_k^\varepsilon)$. Usando el Teorema de la divergencia,

$$\begin{aligned} 0 &= \int_{\Omega_k^\varepsilon} \mathrm{div}(u)\mathrm{d}x \\ &= \int_{\partial\Omega_k^\varepsilon} u \cdot n\mathrm{d}s \\ &= \int_{\partial\Omega_k^\varepsilon \cap \partial\Omega^\varepsilon} u \cdot n\mathrm{d}s + \int_{\Sigma_k^\varepsilon} u \cdot n\mathrm{d}s \\ &= \int_{\partial\Omega_k^\varepsilon \cap \partial\Omega^\varepsilon} u_0 \cdot n\mathrm{d}s + \int_0^\varepsilon u_1\mathrm{d}s \end{aligned}$$

De donde se obtiene que, $\int_0^\varepsilon u_1\mathrm{d}s = -\int_{\partial\Omega_k^\varepsilon \cap \partial\Omega^\varepsilon} u_0 \cdot n\mathrm{d}s$.
Del hecho que $(u-g) \in H_0^1(\Omega_\varepsilon)^2$ y el relevo $g = u_0$ sobre Γ_ε, se sigue que
$u = u_0$ sobre Γ_ε, y luego, $u = u_0$ en el sentido de la traza sobre $\partial\Omega_\varepsilon \cap \partial\Omega_k^\varepsilon$. Además, $(u-g)|_{O_{\varepsilon,L}} = \begin{pmatrix} (u_1 - g_1)(x_2) \\ 0 \end{pmatrix}$, entonces

$$\begin{aligned} u(0,x_2) &= \begin{pmatrix} u_1(0,x_2) \\ g_2(0,x_2) \end{pmatrix} \text{ sobre } \Sigma_1^\varepsilon \\ u(L,x_2) &= \begin{pmatrix} u_1(L,x_2) \\ g_2(L,x_2) \end{pmatrix} \text{ sobre } \Sigma_2^\varepsilon \end{aligned}$$

Por otro lado, u es solución de, $\int_{\Omega_\varepsilon} \nabla u : \nabla w\mathrm{d}x = \int_{\Omega_\varepsilon} f \cdot w\mathrm{d}x$, $\forall w \in W^\varepsilon$, en particular, se tiene que
$\forall w \in \mathcal{D}(\Omega_\varepsilon)^2$, tal que $\mathrm{div}(w) = 0$ en Ω_ε y $w|_{O_{\varepsilon,L}} = \begin{pmatrix} w_1(x_2) \\ 0 \end{pmatrix}$,

$$\langle \Delta u + f, w \rangle = 0$$

Usando el Teorema 2.6 con $\varphi = \Delta u + f$, se obtiene que existen existen $p_1 \in L^2(\Omega_1^\varepsilon)$ y $p_2 \in L^2(\Omega_2^\varepsilon)$ únicas salvo constante tales que

$$\begin{aligned} \Delta u + f &= \nabla p_1 \quad \text{en } \mathcal{D}'(\Omega_1^\varepsilon) \\ \Delta u + f &= \nabla p_2 \quad \text{en } \mathcal{D}'(\Omega_2^\varepsilon) \\ \int_0^L \frac{\partial^2 u_1}{\partial x_2^2} + f_1(x_1,\cdot)\mathrm{d}x_1 &= [p] \quad \text{en } \mathcal{D}'((0,\varepsilon)) \end{aligned}$$

Otra opción es notar que $\varphi = \Delta u + f \in Y^{\varepsilon\perp}$ y usando el Lema 2.5, se obtiene que $\Delta u + f \in X$, dando un resultado análogo al de usar Teorema 2.6. Lo cual concluye la prueba. □

Observación: Para determinar constantes de manera única de la presión, se puede por ejemplo integrar la ecuación (2.11), respecto a x_2, obteniendo

$$\int_0^\varepsilon \int_0^L \frac{\partial^2 u_1}{\partial x_2^2}\mathrm{d}x_1\mathrm{d}x_2 = \int_0^\varepsilon [p]\mathrm{d}x_2 - \int_0^\varepsilon \int_0^L f\mathrm{d}x_1\mathrm{d}x_2$$

e imponer, $\int_0^\varepsilon \int_0^L \frac{\partial^2 u_1}{\partial x_2^2}\mathrm{d}x_1\mathrm{d}x_2 = 0$, lo cual implica que $\int_0^\varepsilon [p]\mathrm{d}x_2 = \int_0^\varepsilon \int_0^L f\mathrm{d}x_1\mathrm{d}x_2$. Esto es, la presión es tal que integrar verticalmente el salto de presión coincide con integrar la componente tangencial de la fuerza en todo el tubo $O_{\varepsilon,L}$.

Capítulo 3

Marea Roja

3.1. Biología

Desde 1972 se ha señalado a la *Alexandrium catenella* como la gran responsable del Veneno Paralizante de los Mariscos (VPM), encontrada en la región de Magallanes. La *Alexandrium catenella* es un Fitoplancton, organismo con capacidad de realizar fotosíntesis. Desde un punto de vista toxicológico, es la microalga de mayor importancia, por lo cual es fundamental estudiar su ciclo de vida para comprender el fenómeno de marea roja.
El ciclo de la *Alexandrium catenella* consta de cinco etapas.

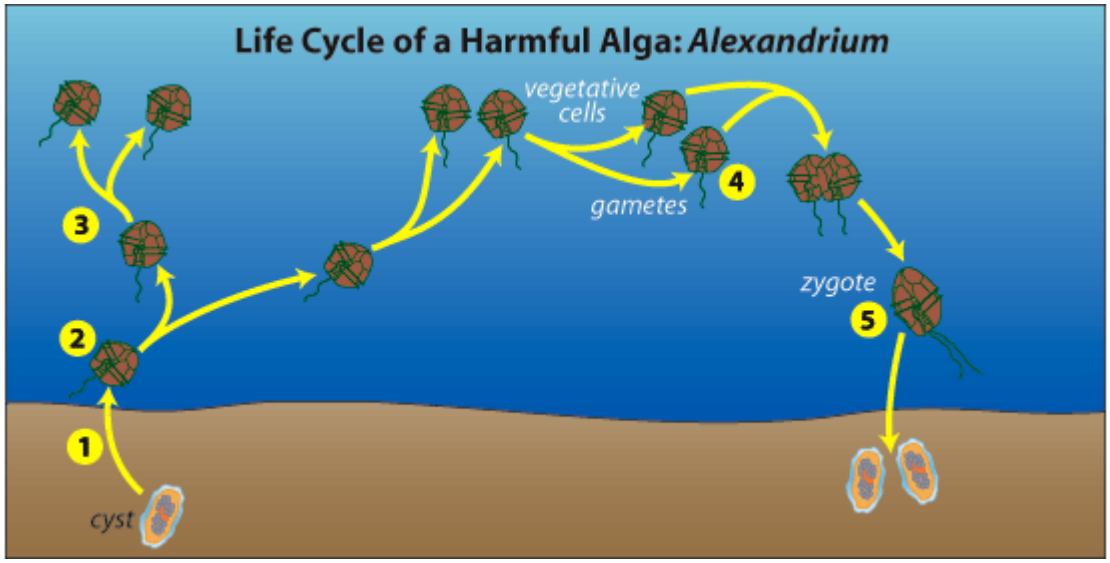


Figura 3.1: Ciclo de vida de Alexandrium Catenella.

1. **Quiste**: Residen en el fondo del mar, cubierto por sedimentos. Esta etapa puede durar años.
2. **Germinación**: Ocurre bajo condiciones especificas del medio ambiente, tales como la temperatura, luz y oxígeno. El Quiste se rompe y emerge una celula, la cual se puede

reproducir (dividiéndose) a los pocos días.

3. **Crecimiento vegetativo**: La célula nada hacia la superficie, comenzando su etapa de crecimiento gracias a la luz y la disponibilidad de nutrientes.
4. **Gameto**: Cuando se acaban los nutrientes, el crecimiento vegetativo se termina y se forman dos gametos.
5. **Cigoto**: Los dos gametos anteriores se unen, formando una célula la cual se desarrolla hasta convertirse en cigoto y luego en un quiste, el cual cae al fondo del mar y comienza el ciclo nuevamente.

3.2. Modelo Matemático

Como se explica en la introducción, la marea roja es un fenómeno tremendamente complejo, pues depende de muchos factores tanto físicos como biólogicos, entre ellos la temperatura, intensidad de la luz, concentración de nutrientes, velocidad del mar, entre muchos otros. Por lo cual para modelarlo se deben hacer simplificaciones.
El objeto de estudio es c la concentración de alga en cierta porción del mar. Dicha concentración evoluciona con el tiempo y no es homógenea en espacio. Para describir la concentración de microalgas se utiliza la ecuación de convección-difusión:

$$\frac{\partial c(x,t)}{\partial t} + \mathrm{div}(cu) - \mathrm{div}(D\nabla c) = F. \tag{3.1}$$

con u la velocidad de la marea, D la constante de difusión del alga en el mar y F el término de fuente. Esta ecuación es una combinación de dos procesos físicos:

- Transporte convectivo (advectivo): es la capacidad del medio de transportar la concentración debido a la velocidad del medio.
- Difusión: las partículas se mueven de las zonas de mayor concentración a zonas de menor concentración.

Esta ecuación en el fondo se refiere a que si la concentración de algas comienza en un lugar, por ejemplo el centro, el medio, es decir el mar, debido a su dinámica transporta el alga, por lo cual la concentración en ese sitio inicial va cambiar.
El término fuente F se determina como la tasa de nacimiento menos la tasa de muerte de microalgas. Es en éste término que aparece la componente biológica (temperatura del ambiente, intensidad de la luz, salinidad, etc). Sin embargo, por la complejidad de este término al no contar con una expresión analítica, en esta tesis se trabaja con la hipótesis $F = 0$, es decir, no se producen nuevas microalgas ni mueren dentro del dominio de interés. La idea es contar en el futuro con datos para poder modelar esta función. De momento, basta con suponer $F = 0$ y así estudiar el transporte de microoalgas en el mar.
El principal problema de esta formulación es que depende de la velocidad del medio que en este caso es el Mar: la concentración es la incógnita y la velocidad debe ser dato, pero claramanente la velocidad del mar no es un dato, o al menos hasta el momento no existe una forma de medir la velocidad del mar en cada punto y tiempo. Hallar la velocidad del mar es más difícil que resolver (3.1), pues en la convección-difusión hay una sola incógnita: la concentración.

3.3. Campo de velocidades

Antes de resolver la EDP asociada a la concentración de algas, se debe calcular el campo de velocidad de la marea. Está se calcula a través de las ecuaciones de Navier-Stokes:

$$\rho\left(\frac{\partial u}{\partial t} + (u\cdot\nabla)u - \mu\Delta u\right) = -\nabla p + f \text{ en } \Omega \tag{3.2}$$

$$\nabla\cdot u = 0 \text{ en } \Omega \tag{3.3}$$

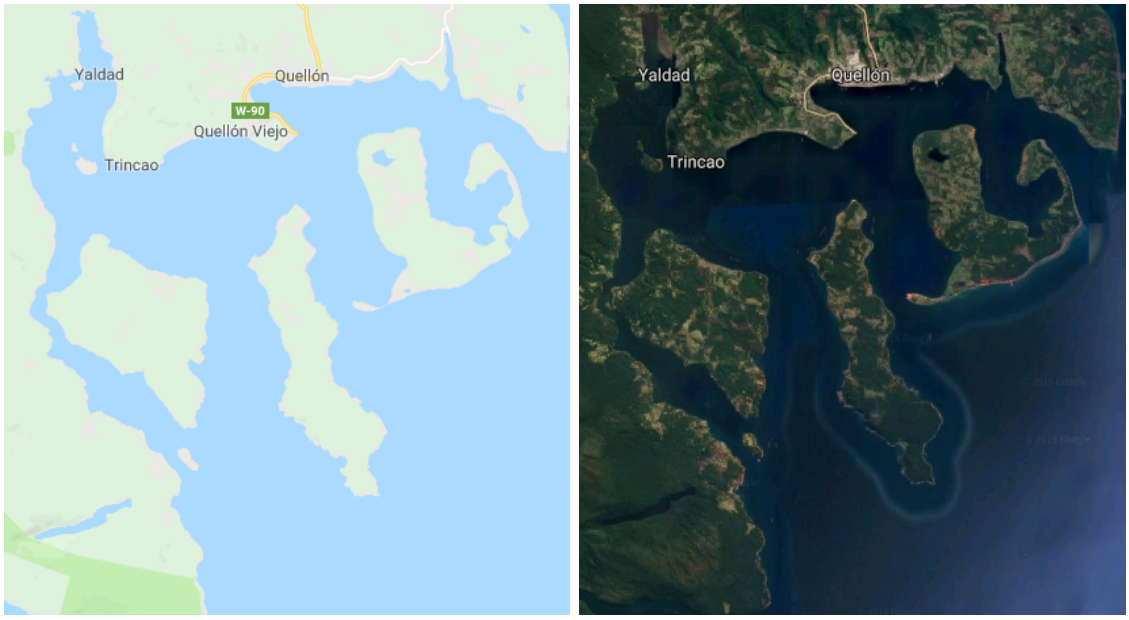
donde las incógnitas son u la velocidad y p la presión. Notar que $u : \mathbb{R}\times\mathbb{R}^3 \to \mathbb{R}^3$, y la presión $p : \mathbb{R}\times\mathbb{R}^3 \to \mathbb{R}^3$. Luego, $u(t,x,y,z) = (u_1(t,x,y,z), u_2(t,x,y,z), u_3(t,x,y,z))$ sus componentes en el eje x, y, z respectivamente.
Y los siguientes parámetros:

- f la fuerza por unidad de volumen.
- μ la viscocidad dinámica de un fluido.
- ρ la densidad del fluido.

Está probado (ver [9]) que existe solución única si la viscocidad es lo suficientemente grande. Este régimen se conoce como laminar (el movimiento del fluido es ordenado), mientras que para viscocidades pequeñas el movimiento del fluido es desordenado, regimen conocido como turbulento.
Se define $\nu = \dfrac{\mu}{\rho}$ la viscocidad cinemática.
Para el caso del agua la viscocidad cinemática es del orden de 10^{-6} a 10^{-5} $\frac{m^2}{s}$, la cual es bastante pequeña y se entra en regímenes turbulentos (lo cual tiene sentido pues el movimiento del mar es desordenado).
Como primera aproximación a resolver el problema, en esta tesis se desarolla el caso laminar. El caso turbulento requiere de tener bien estudiado el caso laminar (que es resolver Navier-Stokes estable), pues aparecen nuevas componentes en las ecuaciones.
La viscosidad corresponde con el concepto informal de *espesor*. Es una propiedad física característica de todos los fluidos, la cual emerge de las colisiones entre las partículas del fluido que se mueven a diferentes velocidades, provocando una resistencia a su movimiento. En pocas palabras es una resistencia a fluir. Es una medida de la resistencia de la deformación del fluido.

3.4. Bahía Quellón

El dominio de interés es la Bahía Quellón, ubicado en Chiloé, Región de los Lagos. La siguiente figura muestra la Bahía Quellón visto con Google Earth.



(a) Sin relieves. (b) Con relieves.

Figura 3.2: Bahía Quellón vista desde arriba con Google Earth.

3.5. Diferencia entre oleaje y marea

Es recurrente confundir oleaje con marea, por lo cuál es necesario entender la diferencia entre éstos conceptos, ya que el objeto de interés en este problema es la marea y no el oleaje, para así imponer las condiciones de borde adecuadas.
Las olas son movimientos ondulatorios, oscilaciones periódicas de la superficie del mar, formadas por crestas y depresiones que se desplazan horizontalmente. Las olas las crea el rozamiento del viento sobre la superficie del agua y la realimentación: cuanto más altas, más viento recogen y más crecen. La marea es el cambio periódico del nivel del mar producido principalmente por las fuerzas de atracción gravitatoria que ejercen el Sol y la Luna sobre la Tierra.
Marea alta es el momento en que el agua del mar alcanza su máxima altura en el ciclo de la marea, mientras que la *Marea baja* es el momento en que el agua del mar alcanza su altura mínima en el ciclo de la marea.
Un modelo simple que explica por qué se producen las mareas proviene del equilibrio de fuerzas usando la segunda Ley de Newton. En esta explicación se considera la Tierra como un cuerpo rígido de forma esférica de radio R_T, que está cubierta por una capa de agua de espesor uniforme. La Tierra se ve sometida a la atracción gravitatoria tanto del Sol como la Luna, ambos influyen en la marea. Por simplicidad simplemente se considera el efecto de la Luna.
La Tierra y la Luna rotan en torno al centro de masas común del sistema Tierra-Luna. La distancia entre el centro de masa de la Tierra y el centro de masa de la Luna se denota d_{TL}.
La fuerza de marea F_t en un punto de la superficie de la Tierra, es igual a la diferencia entre la fuerza de atracción que la Luna ejerce sobre un objeto situado en dicha posición, y la

fuerza de atracción que ejercería sobre tal objeto si estuviese en el centro de la Tierra. Hay cuatro puntos interesantes A, A' y B, B', que representan las dos zonas de marea alta y las zonas de marea baja, respectivamente. En la siguiente figura se esquematiza esta situación

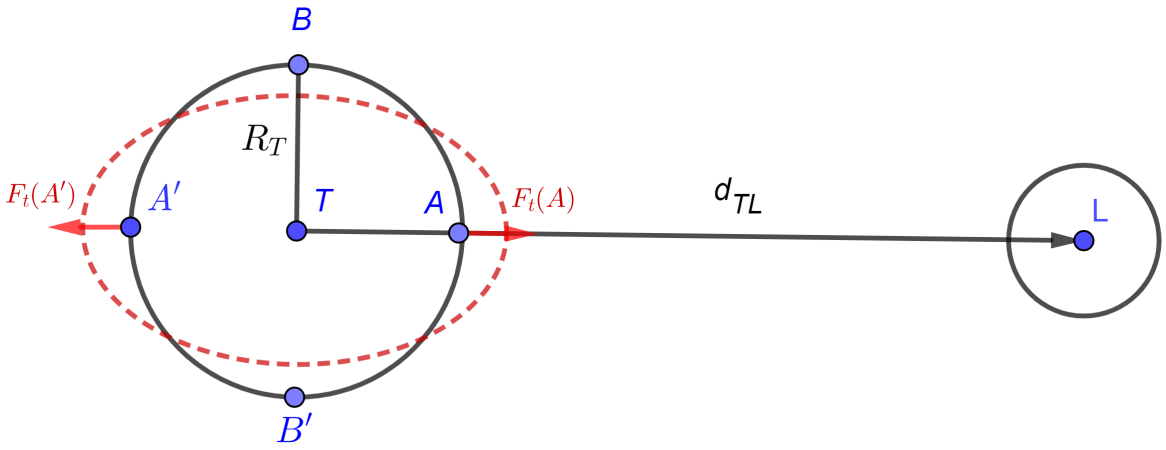


Figura 3.3: Ilustración de la fuerza de mareas.

- En el centro de masa de la Tierra C, hay equilibrio de fuerzas entre la atracción gravitacional ejercida por la Luna sobre la Tierra y la fuerza centrífuga. Es decir,

$$F_g(T) = F_c(T) \tag{3.4}$$

- En el punto A la atracción de gravedad ejercida por la Luna es mayor (menor distancia) a la fuerza fuerza centrífuga. Luego la fuerza de marea es la diferencia

$$F_t(A) = F_g(A) - F_g(T)$$

La distancia entre el centro de masa de la Luna y el punto A de la Tierra es $\mathrm{d}_{TL} - R_T$, por lo tanto $F_g(A) = -G\frac{M_T M_L}{(\mathrm{d}_{TL}-R_T)^2}\hat{\mathrm{i}}$. Entonces,

$$\begin{aligned} F_t(A) &= -GM_T M_L \left(\frac{1}{(\mathrm{d}_{TL}-R_T)^2} - \frac{1}{\mathrm{d}_{TL}^2}\right)\hat{\mathrm{i}} \\ &= -GM_T M_L \frac{2\mathrm{d}_{TL}R_T - R_T^2}{\mathrm{d}_{TL}^2(\mathrm{d}_{TL}-R_T)^2}\hat{\mathrm{i}} \end{aligned}$$

Como $R_T << \mathrm{d}_{TL}$, se puede hacer la siguiente aproximación

$$F_t(A) \approx \frac{-2GM_T M_L R_T}{\mathrm{d}_{TL}^3}\hat{\mathrm{i}}$$

- En el punto A' la atracción de gravedad ejercida por la Luna es menor (mayor distancia) que la fuerza centrífuga. La fuerza de marea es la diferencia $F_t(A') = F_g(A') - F_g(T)$. Analogamente a los cálculos anteriores, como la distancia entre el centro de masa de la Luna y el punto A' de la Tierra es $\mathrm{d}_{TL} + R_T$, se obtiene que

$$F_t(A) \approx \frac{2GM_T M_L R_T}{\mathrm{d}_{TL}^3}\hat{\mathrm{i}}$$

- En el punto B la fuerza de atracción gravitacional es radial $\hat{r}$ y como $F_g(T)$ apunta en $\hat{\imath}$, es necesario hacer la descomposición de fuerzas $\hat{r} = cos(\theta)\hat{\imath} + seno(\theta)\hat{j}$, como el ángulo θ es muy pequeño, se tiene que $cos(\theta) \approx 1$ y $sen(\theta) \approx \frac{R_T}{d_{TL}}$. Entonces

$$\begin{aligned} F_t(B) &= \frac{-GM_TM_LR_T}{d_{TL}(d_{TL}^2 + R_T^2)}\hat{j} - \frac{GM_TM_L}{d_{TL}^2 + R_T^2}\hat{\imath} + \frac{GM_TM_L}{d_{TL}^2}\hat{\imath} \\ &\approx -\frac{GM_TM_LR_T}{d_{TL}^3}\hat{j} \end{aligned}$$

  Analogamente para B', se obtiene que $F_t(B') = \frac{GM_TM_LR_T}{d_{TL}^3}\hat{j}$.
  Notar que, en módulo $F_t(B)$ y $F_t(B')$ es la mitad que $F_t(A)$ y $F_t(A')$.

Los cálculos muestran que las mareas se producen por la diferencia de fuerza de atracción gravitatoria con la Luna (con el Sol es análogo) entre el punto más cercano y más lejano de la Tierra a la Luna (que están conectados por el diámetro en la dirección radial a la Luna). Esta diferencia provoca que se estire la Tierra desde estos extremos, formando un elipsoide como el que muestra la figura 3.3. Además, como el período de rotación de la Tierra es de aproximadamente 24 horas, se demora 6 horas en pasar del punto A al punto B, igualmente en pasar de B a A', de A' a B' y de B' a A. Esto explica porque el tiempo aproximado entre marea alta y marea baja es de 6 horas y el período de mareas (entre una marea alta y la marea alta siguiente o entre una marea baja y la marea baja siguiente) es de 12 horas aproximadamente.
Para trabajar con datos reales de mareas, se extraen datos a partir tablas de mareas de la Bahía Quellón de internet [1]. Con este fin, se usa la función *getTableFromWeb* de MATLAB [2], que permite leer tablas con datos y retorna los strings. Por lo tanto, el trabajo fue convertir esos strings (considerar que no todos los días tienen 4 mareas) en vector y posteriormente realizar una interpolación trigonométrica. Todo sto se hace en el script *GenerarDatoDeTablasDeMarea* en MATLAB.
La tabla 3.4 muestra las cuatro mareas al día que se producen en el día, durante una semana completa: desde el 04 de Febrero hasta el 09 de Febrero.

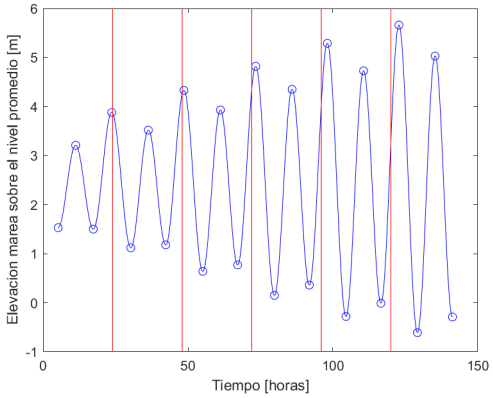
El gráfico 3.5a muestra la altura que alcanza la marea en cada hora del día, en la Bahía Quellón la semana desde el 04 de Febrero al 09 de Febrero. Esto es útil a la hora de establecer las condiciones de borde del problema, para posteriormente simularlo. Las rectas verticales, son líneas de separación entre cada día. La figura 3.5b muestra la altura de las mareas del día 04 de Febrero. Los puntos azules muestran las cuatro mareas (bajas y altas) de ese día.

[1] La página utilizada es https://es.tideschart.com/Chile/Los-Lagos/Provincia-de-Chiloe/Puerto-Quellon. También se consideró la opción de usar datos del SHOA https://www.shoa.cl/php/mareas.php, el problema es que no se encuentran datos de la Bahía Quellón en este caso. De todas formas no es tan relevante, el propósito es simplemente considerar como condición de borde algo más realista que $sen(\omega t)$ y $cos(\omega t)$, sino que combinaciones de estás con distintas amplitudes en cada marea.

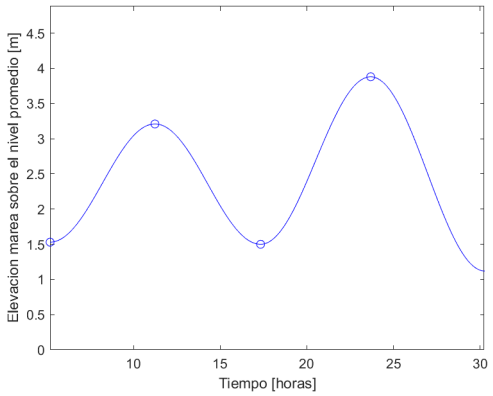
[2] https://www.mathworks.com/matlabcentral/fileexchange/29642-get-html-table-data-into-matlab-via-urlread-and-without-builtin-browser Cortesía de Sven Koerner por compartir su función.

DIA	MAREAS PARA PUERTO QUELLóN					
	1ra MAREA	2da MAREA	3ra MAREA	4ta MAREA		
18 Fri	04:50 h 4.63 m	11:10 h 0.56 m	17:13 h 4.02 m	23:13 h 0.92 m	06:59 h	20:21 h
19 Sat	05:27 h 4.38 m	11:53 h 0.85 m	17:58 h 3.66 m	23:56 h 1.26 m	06:57 h	20:23 h
20 Sun	06:17 h 4.08 m	12:54 h 1.15 m	19:03 h 3.33 m		06:55 h	20:24 h
21 Mon	01:03 h 1.59 m	07:29 h 3.81 m	14:24 h 1.3 m	20:41 h 3.2 m	06:54 h	20:25 h
22 Tue	02:46 h 1.71 m	09:09 h 3.78 m	16:03 h 1.11 m	22:23 h 3.47 m	06:52 h	20:26 h
23 Wed	04:27 h 1.42 m	10:40 h 4.1 m	17:17 h 0.66 m	23:33 h 4.01 m	06:51 h	20:28 h
24 Thu	05:37 h 0.87 m	11:47 h 4.6 m	18:12 h 0.14 m		06:49 h	20:29 h

Figura 3.4: Tablas de las mareas en Puerto Quellón desde el dia 04 de Febrero del 2020 al 09 de Febrero.



(a) Gráfico de las mareas (altura vs horas) en Puerto Quellón el dia 18 de Octubre del 2019.



(b) Gráfico de las mareas (altura vs horas) en Puerto Quellón el dia 18 de Octubre del 2019.

Figura 3.5: Gráficos en MATLAB a partir de la tabla de mareas leída desde internet.

Capítulo 4

Ecuaciones Discretizadas

En este capítulo se estudian las ecuaciones discretizadas de Navier-Stokes y convección-difusión usando el Método de Volúmenes Finitos. Se implementan los algoritmos tanto en MATLAB como en $C++$.

4.1. Ecuaciones discretizadas de Navier-Stokes

Como al discretizar, cada componente de velocidad tendrá dimensión igual al número de volúmenes finitos considerados, se vuelve muy pesada la notación $u = (u_1, u_2, u_3)$ para la velocidad, lo cual conlleva a adoptar la notación $u = (U, V, W)$ para la velocidad. Además, considerando la densidad ρ constante, se define $P := \dfrac{p}{\rho}$, que por simplicidad se sigue llamando presión. Por último, la fuerza f simplemente se expresa como $f = (f \cdot \mathrm{e}_1, f \cdot \mathrm{e}_2, f \cdot \mathrm{e}_3)$ en que $\mathrm{e}_1, \mathrm{e}_2, \mathrm{e}_3$ es la base canónica en $\mathbb{R}^3$.
Con la notación especificada arriba la ecuación de momentum en el eje x queda:

$$\frac{\partial U}{\partial t} + u \cdot \nabla U - \nu \Delta U + \frac{\partial P}{\partial x} = f \cdot \mathrm{e}_1$$

Sean $\omega_\mathrm{i} \subset \Omega$ volumenes finitos, con $\mathrm{i} \in \{1, \ldots, N\}$ y N el número de volúmenes finitos, es decir $\Omega = \bigcup_{\mathrm{i}=1}^{N} \omega_\mathrm{i}$. Sea ω_i un volumen finito, cuyo borde es $\partial\omega_\mathrm{i} = \bigcup_{j:\Gamma_j \text{ es cara de } \omega_\mathrm{i}} \Gamma_j$.
Integrando sobre un volumen finito ω_i cualquiera,

$$\int_{\omega_\mathrm{i}} \frac{\partial U}{\partial t} \mathrm{d}\omega + \int_{\omega_\mathrm{i}} u \cdot \nabla U \mathrm{d}\omega - \nu \int_{\omega_\mathrm{i}} \Delta U \mathrm{d}\omega + \int_{\omega_\mathrm{i}} \frac{\partial P}{\partial x} \mathrm{d}\omega = \int_{\omega_\mathrm{i}} f \cdot \mathrm{e}_1 \mathrm{d}\omega$$

Por hipótesis de volúmenes finitos las funciones son constantes en el interior de cada volumen finito (en los bordes depende de la interacción entre volúmenes contiguos).

$\forall x \in \omega_\mathrm{i}$,

$$\begin{array}{ccccccc} U(x,t) & = & U(x_\mathrm{i},t) & =: & U_\mathrm{i}(t) & =: & U_\mathrm{i}^k \\ P(x,t) & = & P(x_\mathrm{i},t) & =: & P_\mathrm{i}(t) & =: & P_\mathrm{i}^k \\ f\cdot \mathrm{e}_1(x,t) & = & f\cdot \mathrm{e}_1(x_\mathrm{i},t) & =: & (f\cdot \mathrm{e}_1)_\mathrm{i}(t) & =: & (f\cdot \mathrm{e}_1)_\mathrm{i}^k \end{array}$$

en que i será usado como subíndice y representa el valor de la función en el volumen ω_i, mientras que $k \in \mathbb{N}$ el superíndice es la etapa temporal actual.

A continuación se discretiza cada término:

- Término de evolución:

$$\begin{aligned} \int_{\omega_\mathrm{i}} \frac{\partial U}{\partial t}\mathrm{d}\omega &= |\omega_\mathrm{i}|\,\frac{\partial U}{\partial t} \\ &\approx \frac{|\omega_\mathrm{i}|}{\Delta t}(U_\mathrm{i}^k - U_\mathrm{i}^{k-1}) \end{aligned}$$

- Término de convección:

$$\begin{aligned} \int_{\omega_\mathrm{i}} u\cdot\nabla U\mathrm{d}\omega &= \int_{\omega_\mathrm{i}} \mathrm{div}(Uu)\mathrm{d}\omega && \text{usando que } \mathrm{div}(u)=0 \\ &= \int_{\partial\omega_\mathrm{i}} U(u\cdot n)\mathrm{d}s \\ &\approx \sum_{j:\Gamma_j \text{ cara de } \omega_\mathrm{i}} U_\mathrm{i}^k(\Gamma_j)(u^k\cdot n)(\Gamma_j)\,|\Gamma_j| \end{aligned}$$

  Queda definir bien el término de velocidad en el borde, pues ésta toma valores distintos en los volúmenes finitos vecinos. Si Γ_j es cara externa, es decir $\Gamma_j \in \partial\Omega$, simplemente $U^k(\Gamma_j) = U_\mathrm{i}^k$. Si la cara Γ_j es interna, es decir, $\partial\omega_\mathrm{i} \cap \partial\omega_j = \Gamma_j$, entonces se utiliza la condición conocida como *upwind*

$$U^k(\Gamma_j) = \begin{cases} U_\mathrm{i}^k & \text{si } (u^k\cdot n)(\Gamma_j) \geq 0 \\ U_j^k & \text{si } (u^k\cdot n)(\Gamma_j) < 0 \end{cases}$$

  Esto quiere decir, que si el flujo sale del volumen finito ω_i (entra al volumen finito vecino ω_j) la velocidad en el borde es la del volumen finito con que entra el flujo U_i^k, y si el flujo entra al volumen finito ω_i (sale del volumen finito vecino ω_j la velocidad en el borde es la del volumen finito con que entra U_j^k).

$$\begin{aligned} \int_{\omega_\mathrm{i}} u\cdot\nabla U\mathrm{d}\omega \approx &\sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ es cara interna} \\ (u^k\cdot n)(\Gamma_j)<0}} U_j^k(u^k\cdot n)(\Gamma_j)\,|\Gamma_j| + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ es cara interna} \\ (u^k\cdot n)(\Gamma_j)\geq 0}} U_\mathrm{i}^k(u^k\cdot n)(\Gamma_j)\,|\Gamma_j| \\ &+ \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ es cara externa}}} U_\mathrm{i}^k(u^k\cdot n)(\Gamma_j)\,|\Gamma_j| \quad (4.1) \end{aligned}$$

  Además, del hecho de que $\mathrm{div}(u) = 0$ en ω_i y usando el Teorema de la Divergencia

$$
\begin{aligned}
0 &= \int_{\omega_\mathrm{i}} \mathrm{div}(u)\mathrm{d}\omega \\
&= \int_{\partial\omega_\mathrm{i}} u \cdot n \mathrm{d}s \\
&\approx \sum_{j:\Gamma_j \text{ cara de } \omega_\mathrm{i}} u^k \cdot n(\Gamma_j)\,|\Gamma_j| \\
&= \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara interna} \\ (u^k\cdot n)(\Gamma_j)<0}} u^k \cdot n(\Gamma_j)\,|\Gamma_j| + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara interna} \\ (u^k\cdot n)(\Gamma_j)\geq 0}} u^k \cdot n(\Gamma_j)\,|\Gamma_j| + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara externa}}} u^k \cdot n(\Gamma_j)\,|\Gamma_j|
\end{aligned}
$$

De donde se obtiene que,

$$
\begin{aligned}
\sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara interna} \\ (u^k\cdot n)(\Gamma_j)\geq 0}} u^k \cdot n(\Gamma_j)\,|\Gamma_j| + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara externa}}} u^k \cdot n(\Gamma_j)\,|\Gamma_j| \\
= -\sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara interna} \\ (u^k\cdot n)(\Gamma_j)<0}} u^k \cdot n(\Gamma_j)\,|\Gamma_j| \quad (4.2)
\end{aligned}
$$

Multiplicando por U_i^k,

$$
\begin{aligned}
\sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara interna} \\ (u^k\cdot n)(\Gamma_j)\geq 0}} U_\mathrm{i}^k (u^k \cdot n)(\Gamma_j)\,|\Gamma_j| + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara externa}}} U_\mathrm{i}^k (u^k \cdot n)(\Gamma_j)\,|\Gamma_j| \\
= -\sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara interna} \\ (u^k\cdot n)(\Gamma_j)<0}} U_\mathrm{i}^k (u^k \cdot n)(\Gamma_j)\,|\Gamma_j| \quad (4.3)
\end{aligned}
$$

Finalmente, reemplazando esta última expresión en (4.1) se obtiene la discretización del término convectivo

$$
\int_{\omega_\mathrm{i}} u \cdot \nabla U \mathrm{d}\omega \approx \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara interna} \\ (u^k\cdot n)(\Gamma_j)<0}} (U_j^k - U_\mathrm{i}^k)(u^k \cdot n)(\Gamma_j)\,|\Gamma_j| \quad (4.4)
$$

- Término de difusión: Laplaciano

$$
\begin{aligned}
\int_{\omega_\mathrm{i}} \Delta U \mathrm{d}\omega &= \int_{\omega_\mathrm{i}} \mathrm{div}(\nabla U)\mathrm{d}\omega \\
&= \int_{\partial\omega_\mathrm{i}} \nabla U \cdot n \mathrm{d}s && \text{por Teorema de la Divergencia} \\
&= \int_{\partial\omega_\mathrm{i}} \frac{\partial U}{\partial n}\mathrm{d}s \\
&\approx \sum_{j:\Gamma_j \text{ cara de } \omega_\mathrm{i}} \frac{(U_j^k - U_\mathrm{i}^k)}{\mathrm{d}_c(\omega_\mathrm{i}, \omega_j)}\,|\Gamma_j|
\end{aligned}
$$

donde $\mathrm{d}_c(\omega_\mathrm{i}, \omega_j)$ denota la distancia desde el centro del volumen finito ω_i al centro del volumen finito ω_j. Esta aproximación de la derivada normal tiene sentido si el vector distancia entre centros es perpendicular a la cara Γ_j. Sino, se puede considerar como $\mathrm{d}_c(\omega_\mathrm{i}, \omega_j) = \mathrm{d}(\omega_\mathrm{i}, \omega_j) \cdot n(\Gamma_j)$ la distancia entre centros producto punto la normal a Γ_j.

- Término de Difusión: Gradiente de presión

$$\begin{aligned} \int_{\omega_\mathrm{i}} \frac{\partial P}{\partial x} \mathrm{d}\omega &= \int_{\partial\omega_\mathrm{i}} P n_x \mathrm{d}s \qquad \text{integrando por partes} \\ &\approx \sum_{j:\Gamma_j \text{ cara de } \omega_\mathrm{i}} P_j n_x(\Gamma_j) \left|\Gamma_j\right| \end{aligned}$$

en que $n_x(\Gamma_j)$ es la componente x de la normal exterior a Γ_j.

Observación: Surge la pregunta natural de si definir la variable presión en las caras o en los volúmenes finitos. La respuesta es que se consideran presiones en cada volumen finito y también las presiones en las caras externas en las cuales no hay condición de borde sobre la velocidad. Esto se explica más adelante.

- Término fuente:

$$\int_{\omega_\mathrm{i}} f \cdot \mathrm{e}_1 \mathrm{d}\omega \approx (f \cdot \mathrm{e}_1)_\mathrm{i}^k \left|\omega_\mathrm{i}\right|$$

Luego la ecuación de momentum en x discretizada queda como sigue a continuación:

$$\begin{aligned} \frac{|\omega_\mathrm{i}|}{\Delta t} U_\mathrm{i}^k + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara interna} \\ (u^k \cdot n)(\Gamma_j) < 0}} (U_j^k - U_\mathrm{i}^k)(u^k \cdot n)(\Gamma_j) \left|\Gamma_j\right| - \sum_{j:\Gamma_j \text{ cara de } \omega_\mathrm{i}} (U_j^k - U_\mathrm{i}^k) \frac{\nu \left|\Gamma_j\right|}{\mathrm{d}_c(\omega_\mathrm{i}, \omega_j)} \\ + \sum_{j:\Gamma_j \text{ cara de } \omega_\mathrm{i}} P_j^k n_x(\Gamma_j) \left|\Gamma_j\right| = \left|\omega_\mathrm{i}\right| (f \cdot \mathrm{e}_1)_\mathrm{i}^k + \frac{|\omega_\mathrm{i}|}{\Delta t} U_\mathrm{i}^{k-1} \end{aligned} \tag{4.5}$$

Como el término convectivo es no lineal, no se pueden resolver las ecuaciones discretizadas como un sistema lineal del tipo

$$A[U; V; W; P] = b$$

por esta razón es necesario linearizar la discretización. Esto se hace a través de iteraciones tipo punto fijo:

Sea $U_\mathrm{i}^{k,0} := U_\mathrm{i}^k$, $P_\mathrm{i}^{k,0} := P_\mathrm{i}^k$ y para cada $\ell \geq 1$ se definen $U_\mathrm{i}^{k,\ell}$, $P_\mathrm{i}^{k,\ell}$ como la solución de

$$\begin{aligned} \frac{|\omega_\mathrm{i}|}{\Delta t} U_\mathrm{i}^{k,\ell} + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara interna} \\ (u^{k,\ell-1} \cdot n)(\Gamma_j) < 0}} (U_j^{k,\ell-1} - U_\mathrm{i}^{k,\ell})(u^{k,l-1} \cdot n)(\Gamma_j) \left|\Gamma_j\right| + \sum_{j:\Gamma_j \text{ cara de } \omega_\mathrm{i}} (U_\mathrm{i}^{k,\ell} - U_j^{k,\ell}) \frac{\nu \left|\Gamma_j\right|}{\mathrm{d}_c(\omega_\mathrm{i}, \omega_j)} \\ + \sum_{j:\Gamma_j \text{ cara de } \omega_\mathrm{i}} P_j^{k,\ell} n_x(\Gamma_j) \left|\Gamma_j\right| = \left|\omega_\mathrm{i}\right| (f \cdot \mathrm{e}_1)_\mathrm{i}^k + \frac{|\omega_\mathrm{i}|}{\Delta t} U_\mathrm{i}^{k-1} \end{aligned} \tag{4.6}$$

donde se espera que $\lim_{l\to\infty} U_{\mathrm{i}}^{k,\ell} = U_{\mathrm{i}}^{k}$ y $\lim_{\ell\to\infty} P_{\mathrm{i}}^{k,\ell} = P_{\mathrm{i}}^{k}$.

Analógamente, se obtienen las ecuaciones discretizadas de momentum en y, z

$$\frac{|\omega_{\mathrm{i}}|}{\Delta t}V_{\mathrm{i}}^{k,\ell} + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}} \\ \Gamma_j \text{ cara interna} \\ (u^{k,\ell-1}\cdot n)(\Gamma_j)<0}} V_{\mathrm{i}}^{k,\ell}(-u^{k,\ell-1}\cdot n)(\Gamma_j)\,|\Gamma_j| + \sum_{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}}} (V_{\mathrm{i}}^{k,\ell} - V_j^{k,\ell})\frac{\nu\,|\Gamma_j|}{\mathrm{d}_c(\omega_{\mathrm{i}},\omega_j)}$$

$$+ \sum_{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}}} P_j^{k,\ell} n_y(\Gamma_j)\,|\Gamma_j| = |\omega_{\mathrm{i}}|\,(f\cdot \mathrm{e}_2)_{\mathrm{i}}^{k} + \frac{|\omega_{\mathrm{i}}|}{\Delta t}V_{\mathrm{i}}^{k-1} - \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}} \\ \Gamma_j \text{ cara interna} \\ (u^{k,\ell-1}\cdot n)(\Gamma_j)<0}} V_j^{k,\ell-1}(u^{k,\ell-1}\cdot n)(\Gamma_j)\,|\Gamma_j|$$

$$\frac{|\omega_{\mathrm{i}}|}{\Delta t}W_{\mathrm{i}}^{k,\ell} + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}} \\ \Gamma_j \text{ cara interna} \\ (u^{k,\ell-1}\cdot n)(\Gamma_j)<0}} |\Gamma_j|\,W_{\mathrm{i}}^{k,\ell}(-u^{k,\ell-1}\cdot n)(\Gamma_j) + \sum_{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}}} (W_{\mathrm{i}}^{k,\ell} - W_j^{k,\ell})\frac{\nu\,|\Gamma_j|}{\mathrm{d}_c(\omega_{\mathrm{i}},\omega_j)}$$

$$+ \sum_{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}}} P_j^{k,\ell} n_z(\Gamma_j)\,|\Gamma_j| = |\omega_{\mathrm{i}}|\,(f\cdot \mathrm{e}_3)_{\mathrm{i}}^{k} + \frac{|\omega_{\mathrm{i}}|}{\Delta t}W_{\mathrm{i}}^{k-1} - \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}} \\ \Gamma_j \text{ cara interna} \\ (u^{k,\ell-1}\cdot n)(\Gamma_j)<0}} W_j^{k,\ell-1}(u^{k,\ell-1}\cdot n)(\Gamma_j)\,|\Gamma_j|$$

Éstas igualdades son ciertas para todo i $\in \{1,\ldots,N\}$, por cada componente de la velocidad hay N ecuaciones y como la velocidad es vectorial $u = (U,V,W)$ se tiene en total $3N$ ecuaciones, mientras que hay $3N+M$ incógnitas, $3N$ para la velocidad y M para la presión. Por lo tanto faltan ecuaciones.
Primero es necesario aclarar cómo definir la presión y por consiguiente el valor de M, para lo cual hay dos opciones:

1. Definir las presiones sobre las caras, en cuyo caso M corresponde al número de caras.
2. Definir las presiones sobre volúmenes finitos, más las caras externas en que la presión no es condición de borde (y la velocidad sí). Así, M es el número de volúmenes finitos más las caras externas en que la presión no es condición de borde.

Las ecuaciones restantes vienen de la condición de fluido incompresible $\nabla\cdot u = 0$. La idea es traducir esta condición en el número de ecuaciones de cada opción.

Opción 1: Para obtener tantas ecuaciones como caras se impone que el flujo que entra en cada borde es el mismo que lo que debe salir (pensando en que en dicho borde Γ conecta dos volúmenes finitos vecinos ω_{i} y ω_j el flujo entra desde ω_{i} a ω_j):

$$\begin{aligned} u_{\mathrm{i}}^{k}\cdot n_{\mathrm{i}}(\Gamma_j) &= -u_j^{k}\cdot n_j(\Gamma_j) \\ &= u_j^{k}\cdot n_{\mathrm{i}}(\Gamma_j) \end{aligned}$$

pues $n_{\mathrm{i}} = -n_j$, la normal exterior a ω_{i} en la cara Γ_j tiene signo opuesto a la normal exterior a ω_j en la cara Γ_j.
A modo de justificación de dicha hipótesis, se enuncia la siguiente proposición:

Proposición 4.1 *Sea $u \in H^1(\Omega)$ tal que $\mathrm{div}(u) = 0$ en $\Omega = \bigcup_{\mathrm{i}=1}^{N} \omega_{\mathrm{i}}$, con u constante en espacio en cada volumen finito ω_{i}, esto es $u(x,t) = u_{\mathrm{i}}(t)\ \forall x \in \omega_{\mathrm{i}}$.*
Entonces, $\forall k, l \in \{1, \ldots, N\}$,

$$u_l \cdot n(\partial\omega_k \cap \partial\omega_l) = u_k \cdot n(\partial\omega_k \cap \partial\omega_l)$$

con $n := n_l = -n_k$.
Dem:

$$\begin{aligned}
0 &= \langle \mathrm{div}(u), \varphi \rangle \\
&= -\langle u, \nabla\varphi \rangle \\
&= -\int_{\Omega} u \cdot \nabla\varphi \mathrm{d}\omega \\
&= -\sum_{\mathrm{i}=1}^{N} \int_{\omega_{\mathrm{i}}} u \cdot \nabla\varphi \mathrm{d}\omega \\
&= -\sum_{\mathrm{i}=1}^{N} \int_{\omega_{\mathrm{i}}} U\frac{\partial\varphi}{\partial x} + V\frac{\partial\varphi}{\partial y} + W\frac{\partial\varphi}{\partial z} \mathrm{d}\omega \\
&= -\sum_{\mathrm{i}=1}^{N} U_{\mathrm{i}} \int_{\omega_{\mathrm{i}}} \frac{\partial\varphi}{\partial x}\mathrm{d}\omega + V_{\mathrm{i}} \int_{\omega_{\mathrm{i}}} \frac{\partial\varphi}{\partial y}\mathrm{d}\omega + W_{\mathrm{i}} \int_{\omega_{\mathrm{i}}} \frac{\partial\varphi}{\partial z}\mathrm{d}\omega \\
&= -\sum_{\mathrm{i}=1}^{N} u_{\mathrm{i}} \cdot \int_{\omega_{\mathrm{i}}} \nabla\varphi \mathrm{d}\omega \\
&= -\sum_{\mathrm{i}=1}^{N} u_{\mathrm{i}} \cdot \int_{\partial\omega_{\mathrm{i}}} \varphi n \mathrm{d}\omega \\
&= -\sum_{\mathrm{i}=1}^{N} \sum_{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}}} u_{\mathrm{i}} \cdot \int_{\Gamma_j} \varphi n \mathrm{d}s
\end{aligned}$$

En particular, tomando $\varphi \in \mathcal{D}(\omega_k \cup \omega_l)$ tal que $sop(\varphi) \supseteq \partial\omega_k \cap \partial\omega_l$, se tiene que

$$\begin{aligned}
0 &= -u_l \cdot \int_{\Gamma_{k_l}} \varphi n_l \mathrm{d}s - u_k \cdot \int_{\Gamma_{l_k}} \varphi n_k \mathrm{d}s \\
&= -u_l \cdot \int_{\partial\omega_k \cap \partial\omega_l} \varphi n \mathrm{d}s + u_k \cdot \int_{\partial\omega_k \cap \partial\omega_l} \varphi n \mathrm{d}s
\end{aligned}$$

Como las caras son polígonos planos, la normal a una cara es constante sobre la cara, entonces

$$\begin{aligned}
0 &= -(u_l \cdot n) \int_{\partial\omega_k \cap \partial\omega_l} \varphi \mathrm{d}s + (u_k \cdot n) \int_{\partial\omega_k \cap \partial\omega_l} \varphi \mathrm{d}s \\
&= (u_k - u_l) \cdot n \int_{\partial\omega_k \cap \partial\omega_l} \varphi \mathrm{d}s
\end{aligned}$$

En que basta tomar φ tal que $\int_{\partial\omega_k \cap \partial\omega_l} \varphi \mathrm{d}s > 0$. Se concluye que $u_k \cdot n = u_l \cdot n$. □

A pesar de que esta es la verdadera condición, pues la condición de divergencia nula se traduce en que el flujo que entra por una cara, debe ser el mismo flujo que sale, tiene el problema de que por cada presión hay una restricción sobre la velocidad en esa cara, lo

cual conlleva a tener demasiadas restricciones, y para dominios con muchos volúmenes finitos, el número de caras es mayor al número de velocidades, por lo cual hay más restricciones sobre la velocidad que incógnitas.

Opción 2: Es una relajación del caso anterior. Descomponiendo el término en que aparece la presión en las ecuaciones discretizadas,

$$\sum_{j:\Gamma_j \text{ cara de } \omega_\text{i}} P_j^k n_x(\Gamma_j)\,|\Gamma_j| = \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\text{i} \\ \Gamma_j \text{ es cara interna}}} P_j^k n_x(\Gamma_j)\,|\Gamma_j| + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\text{i} \\ \Gamma_j \text{ es cara de borde}}} P_j^k n_x(\Gamma_j)\,|\Gamma_j|$$

Luego la presión se define siguiendo la Opción 2, pues la Opción 1 considera más restricciones que incógnitas, lo cual vuelve el sistema matricial infactible. Entonces la presión en los bordes se calcula como

$$P^k(\Gamma_j) = \begin{cases} \dfrac{P_\text{i}^k + P_j^k}{2} & \text{si } \Gamma_j = \partial\omega_\text{i} \cap \partial\omega_j \text{ es cara interna} \\ P_{\Gamma_j} & \text{si } \Gamma_j \text{ es cara externa (de borde)} \end{cases}$$

Hay tantas presiones M como volúmenes finitos más el número de caras externas, y la restricción asociada a cada presión es,

$$\sum_{j:\Gamma_j \text{ es cara de } \omega_\text{i}} u^k \cdot n(\Gamma_j)\,|\Gamma_j| = 0 \tag{4.7}$$

Así se consiguen las $3N+M$ ecuaciones y se puede plantear un sistema matricial para resolver el problema discretizado.
Para una cierta etapa temporal $k \in \mathbb{N}$ fija y para cada $\ell \in \mathbb{N}$, el sistema de ecuaciones se escribe en forma matricial a continuación

$$\begin{bmatrix} A_{UU} & 0_{N\times N} & 0_{N\times N} & A_{UP} \\ 0_{N\times N} & A_{UU} & 0_{N\times N} & A_{VP} \\ 0_{N\times N} & 0_{N\times N} & A_{UU} & A_{WP} \\ A_{UP}^T & A_{VP}^T & A_{WP}^T & 0_{M\times N} \end{bmatrix} \begin{pmatrix} U^{k,\ell} \\ V^{k,\ell} \\ W^{k,\ell} \\ P^{k,\ell} \end{pmatrix} = \begin{pmatrix} b_U \\ b_V \\ b_W \\ b_P \end{pmatrix} \tag{4.8}$$

donde

$$\begin{aligned} U^{k,\ell} &= (U_1^{k,\ell}, \ldots, U_N^{k,\ell}) \\ V^{k,\ell} &= (V_1^{k,\ell}, \ldots, V_N^{k,\ell}) \\ W^{k,\ell} &= (W_1^{k,\ell}, \ldots, W_N^{k,\ell}) \\ P^{k,\ell} &= (P_1^{k,\ell}, \ldots, P_N^{k,\ell}, P_{N+1}^{k,\ell} \ldots P_M^{k,\ell}) \end{aligned}$$

y,

$$
\begin{aligned}
(A_{UU})_{\mathrm{ii}} &= \frac{|\omega_{\mathrm{i}}|}{\Delta t} + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}} \\ \Gamma_j \text{ cara interna} \\ (u_{\mathrm{i}}^{k,\ell-1}\cdot n)(\Gamma_j)<0}} (-u^{k,\ell-1}\cdot n)(\Gamma_j)\,|\Gamma_j| + \sum_{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}}} \frac{\nu\,|\Gamma_j|}{\mathrm{d}_c(\omega_{\mathrm{i}},\omega_j)} \\
(A_{UU})_{\mathrm{i}j} &= \frac{-\nu\,|\Gamma_j|}{\mathrm{d}_c(\omega_{\mathrm{i}},\omega_j)} \qquad \text{si } \Gamma_j \text{ es cara de } \omega_{\mathrm{i}} \\
(A_{UP})_{\mathrm{ii}} &= \frac{1}{2} \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}} \\ \Gamma_j \text{ cara interna}}} n_x(\Gamma_j)\,|\Gamma_j| \\
(A_{UP})_{\mathrm{i}j} &= \frac{n_x(\Gamma_j)\,|\Gamma_j|}{2} \qquad \text{si } \Gamma_j \text{ es cara de } \omega_{\mathrm{i}} \text{ y } \Gamma_j \text{ cara interna} \\
(A_{UP})_{\mathrm{i}b} &= n_x(\Gamma_b)\,|\Gamma_b| \qquad \text{si } \Gamma_b \text{ es cara de } \omega_{\mathrm{i}} \text{ y } \Gamma_b \text{ cara del borde} \\
(A_{VP})_{\mathrm{ii}} &= \frac{1}{2} \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}} \\ \Gamma_j \text{ cara interna}}} n_y(\Gamma_j)\,|\Gamma_j| \\
(A_{VP})_{\mathrm{i}j} &= \frac{n_y(\Gamma_j)\,|\Gamma_j|}{2} \qquad \text{si } \Gamma_j \text{ es cara de } \omega_{\mathrm{i}} \text{ y } \Gamma_j \text{ cara interna} \\
(A_{VP})_{\mathrm{i}b} &= n_y(\Gamma_b)\,|\Gamma_b| \qquad \text{si } \Gamma_b \text{ es cara de } \omega_{\mathrm{i}} \text{ y } \Gamma_b \text{ cara del borde} \\
(A_{WP})_{\mathrm{ii}} &= \frac{1}{2} \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}} \\ \Gamma_j \text{ cara interna}}} n_z(\Gamma_j)\,|\Gamma_j| \\
(A_{WP})_{\mathrm{i}j} &= \frac{n_z(\Gamma_j)\,|\Gamma_j|}{2} \qquad \text{si } \Gamma_j \text{ es cara de } \omega_{\mathrm{i}} \text{ y } \Gamma_j \text{ cara interna} \\
(A_{WP})_{\mathrm{i}b} &= n_z(\Gamma_b)\,|\Gamma_b| \qquad \text{si } \Gamma_b \text{ es cara de } \omega_{\mathrm{i}} \text{ y } \Gamma_b \text{ cara del borde}
\end{aligned}
$$

los términos restantes son 0. Por lo cual la matriz A es *sparse* y además es simétrica. Notar que

$$
(A_{UP})_{\mathrm{ii}} P_{\mathrm{i}} + (A_{VP})_{\mathrm{ii}} P_{\mathrm{i}} + (A_{WP})_{\mathrm{ii}} P_{\mathrm{i}} = \frac{P_{\mathrm{i}}}{2} \left(\sum_{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}}} n(\Gamma_j)\,|\Gamma_j| - \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}} \\ \Gamma_j \text{ cara externa}}} n(\Gamma_j)\,|\Gamma_j| \right)
$$

y por Teorema de la Divergencia,

$$\begin{aligned} 0 &= \int_{\omega} \mathrm{div} v(1) \\ &= \int_{\partial\omega} n \mathrm{d}s \\ &\approx \sum_{j:\Gamma_j \text{ cara de } \omega_\mathrm{i}} n(\Gamma_j) \left|\Gamma_j\right| \end{aligned}$$

Se deduce que

$$(A_{UP})_{\mathrm{ii}} P_\mathrm{i} + (A_{VP})_{\mathrm{ii}} P_\mathrm{i} + (A_{WP})_{\mathrm{ii}} P_\mathrm{i} = -\frac{P_\mathrm{i}}{2} \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara externa}}} n(\Gamma_j) \left|\Gamma_j\right|$$

¿Qué ecuación es la que se obtiene del último bloque de filas de la matriz A?

$$A_{UP}^T U + A_{VP}^T V + A_{WP}^T W = 0$$

Hay 2 casos, el caso en que la fila corresponda a un volumen finito ω_i o bien a una cara externa Γ_b.

$$\begin{aligned} (A_{UP}^T)_{\mathrm{i}\bullet} U &= \sum_{j=1}^{N} (A_{UP})_{j\mathrm{i}} U_j \\ &= \sum_{j:\Gamma_\mathrm{i} = \partial\omega_\mathrm{i} \cap \partial\omega_j \neq \emptyset} \frac{n_x(\Gamma_\mathrm{i}) \left|\Gamma_\mathrm{i}\right|}{2} U_j \\ (A_{UP}^T)_{b\bullet} U &= \sum_{j=1}^{N} (A_{UP})_{jb} U_j \\ &= \sum_{\substack{j:\Gamma_b \text{ cara de } \omega_j \\ \Gamma_b \text{ cara externa}}} n_x(\Gamma_b) \left|\Gamma_b\right| U_j \\ &= n_x(\Gamma_b) \left|\Gamma_b\right| U_{m_b} \end{aligned}$$

con m_b el volumen finito al cual pertenece la cara de borde Γ_b. Analogamente para los demás productos,

$$\begin{aligned} (A_{VP}^T)_{\mathrm{i}\bullet} V &= \sum_{j:\Gamma_\mathrm{i} = \partial\omega_\mathrm{i} \cap \partial\omega_j \neq \emptyset} \frac{n_y(\Gamma_\mathrm{i}) \left|\Gamma_\mathrm{i}\right|}{2} V_j \\ (A_{WP}^T)_{\mathrm{i}\bullet} W &= \sum_{j:\Gamma_\mathrm{i} = \partial\omega_\mathrm{i} \cap \partial\omega_j \neq \emptyset} \frac{n_z(\Gamma_\mathrm{i}) \left|\Gamma_\mathrm{i}\right|}{2} W_j \end{aligned}$$

Luego,

$$\begin{aligned} (A_{UP}^T)_{\mathrm{i}\bullet} U + (A_{VP}^T)_{\mathrm{i}\bullet} V + (A_{WP}^T)_{\mathrm{i}\bullet} W &= \frac{1}{2} \sum_{j:\Gamma_\mathrm{i} = \partial\omega_\mathrm{i} \cap \partial\omega_j \neq \emptyset} u \cdot n(\Gamma_\mathrm{i}) \left|\Gamma_\mathrm{i}\right| = 0 \\ (A_{UP}^T)_{b\bullet} U + (A_{VP}^T)_{b\bullet} V + (A_{WP}^T)_{b\bullet} W &= u_{m_b} \cdot n(\Gamma_b) \left|\Gamma_b\right| \end{aligned}$$

La primera igualdad se interpreta como la condición de flujo promedio nulo sobre las caras del volumen finito, mientras que la segunda es imponer una condición sobre las velocidades

normales, dado que la presión es incógnita en dichas caras.

El lado derecho queda como

$$
\begin{aligned}
(b_U)_{\mathrm{i}} &= \frac{|\omega_{\mathrm{i}}|}{\Delta t} U_{\mathrm{i}}^{k-1} + |\omega_{\mathrm{i}}| \, (f \cdot \mathrm{e}_1)_{\mathrm{i}}^{k} + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}} \\ \Gamma_j \text{ cara interna} \\ (u_{\mathrm{i}}^{k,\ell-1} \cdot n)(\Gamma_j) < 0}} |\Gamma_j| \, (-u_{\mathrm{i}}^{k,\ell-1} \cdot n)(\Gamma_j) U_j^{k,\ell-1} \\
(b_V)_{\mathrm{i}} &= \frac{|\omega_{\mathrm{i}}|}{\Delta t} V_{\mathrm{i}}^{k-1} + |\omega_{\mathrm{i}}| \, (f \cdot \mathrm{e}_2)_{\mathrm{i}}^{k} + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}} \\ \Gamma_j \text{ cara interna} \\ (u_{\mathrm{i}}^{k,\ell-1} \cdot n)(\Gamma_j) < 0}} |\Gamma_j| \, (-u_{\mathrm{i}}^{k,\ell-1} \cdot n)(\Gamma_j) V_j^{k,\ell-1} \\
(b_U)_{\mathrm{i}} &= \frac{|\omega_{\mathrm{i}}|}{\Delta t} W_{\mathrm{i}}^{k-1} + |\omega_{\mathrm{i}}| \, (f \cdot \mathrm{e}_3)_{\mathrm{i}}^{k} + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}} \\ \Gamma_j \text{ cara interna} \\ (u_{\mathrm{i}}^{k,\ell-1} \cdot n)(\Gamma_j) < 0}} |\Gamma_j| \, (-u_{\mathrm{i}}^{k,\ell-1} \cdot n)(\Gamma_j) W_j^{k,\ell-1} \\
(b_P)_{\mathrm{i}} &= 0 \\
(b_P)_b &= \text{ dato sobre las velocidad normal en la cara } \Gamma_b
\end{aligned}
$$

Con lo cual, el sistema matricial queda completamente determinado.

Observación 1: Con la descripción de las matrices dadas, las condiciones de borde vienen escondidas en la matriz A, las cuales son datos y deben pasar al lado derecho. Esto depende de cada región del borde (Fondo, Superficie, Mar-Tierra y Mar-Mar) y se hace al programar las matrices.

Observación 2: Sin pérdida de generalidad se puede considerar $f = 0$, pues basta considerar el cambio de variable P en lugar de $P - gz$.

4.1.1. Condiciones de borde en Navier-Stokes para la marea

Aquí es necesario tener en consideración que si se impone condición de borde sobre la velocidad en una región, entonces la presión en dicha cara es incógnita, así mismo si hay condición de borde sobre la presión en una región, la velocidad en los volúmenes finitos que tienen una de sus caras en dicha región es incógnita (la *dualidad* entre la velocidad y la presión). A continuación se discute la elección de la condición de borde sobre cada región.

- Fondo: Aquí el fluido debe derrapar, así que se usa $u \cdot n = 0$, y la velocidad tangencial se deja libre.
- Superficie: Esta es la condición de borde que modela la marea. Como el nivel del mar varía sinusoidalmente y se parte con velocidad nula, la elevación del mar en la superficie tiene perfil $z(t) = Acos(\omega t)$ con $\omega = \frac{2\pi}{T}$ el período angular y A la amplitud. Luego la velocidad viene dada por $u_z(t) = W = -A\omega sen(\omega t)$.

- Mar-Tierra: Es la condición de borde más simple, $u = 0$.
- Mar-Mar: Aquí se usa $P = 0$. Se puede pensar en usar presión hidrostática $P = gz$, pero recordar el cambio de variable $P \leftarrow P - gz$.

Regiones del borde	Condición sobre u	Condición sobre P
Fondo	$u \cdot n = 0$	no hay
Superficie	$u_z = W = -A\omega sen(\omega t)$	no hay
Mar-Tierra	$u = 0$	no hay
Mar-Mar	no hay	$P = 0$

Tabla 4.1: Tabla resumen condiciones de borde en Navier-Stokes implementadas en MATLAB.

4.1.2. Solución del sistema lineal: MINRES

Para resolver Navier-Stokes el problema se reduce a resolver el sistema lineal $Ax = b$ con una matriz cuadrada de $(3N+M)\times(3N+M)$ y b un vector de dimensión $3N+M$. Conviene estudiar el problema en general: Sea A una matriz simétrica invertible de d × d y $b \in \mathbb{R}^{\mathrm{d}}$.

En el caso de Navier-Stokes como la matriz A es simétrica y no es definida positiva (por ende no se puede usar Gradiente Conjugado) se opta por utilizar el Método MINRES, el cual se detalla a continuación a partir del artículo original [7].
MINRES es un método iterativo para la solución numérica de un sistema simétrico de ecuaciones lineales.
La idea del método es aproximar la solución por un vector en un subespacio adecuado, de forma de minimizar el residuo $r(x) := \|Ax - b\|$.

Definición 4.2 *Para cada $n \geq 1$ se define el n-ésimo subespacio de Krylov*

$$\mathcal{K}_n = \langle b, Ab, \dots A^{n-1}b\rangle$$

El siguiente resultado es muy importante, pues dice que bajo ciertas condiciones la solución $x = A^{-1}b$ pertenece a un subespacio de Krylov, y así el algoritimo MINRES cobra sentido.

Proposición 4.3 *Si existe $k \in \mathbb{N}$ tal que,* $\dim(\mathcal{K}_1) = 1$, $\dim(\mathcal{K}_2) = 2$,..., $\dim(\mathcal{K}_k) = k = \dim(\mathcal{K}_{k+1})$, *entonces:*

i) $\forall \ell \geq k, \dim(\mathcal{K}_\ell) = k$
ii) $A^{-1}b \in \mathcal{K}_k$.

Dem:

i) Basta probarlo para $\ell = k+2$. Se tiene que $\mathcal{K}_{k+2} = \langle b, Ab, \dots, A^{k+1}b\rangle$, por lo tanto hay que demostrar que $A^{k+1}b$ se escribe como combinación lineal de $b, Ab, \dots, A^{k-1}b$ Dado que $\dim(\mathcal{K}_k) = \dim(\mathcal{K}_{k+1})$, se tiene que $A^k b \in \mathcal{K}_k$, luego existen $\lambda_0, \lambda_1, \dots, \lambda_{k-1} \in \mathbb{R}$, tal que

$$A^k b = \lambda_0 b + \lambda_1 Ab + \dots + \lambda_{k-1}A^{k-1}b \tag{4.9}$$

Multiplicando por A, se obtiene

$$A^{k+1}b = \lambda_0 Ab + \lambda_1 A^2 b + \ldots + \lambda_{k-1} A^k b \tag{4.10}$$

Reemplazando (4.9) en (4.10), se obtiene

$$A^{k+1}b = \bar{\lambda}_0 b + \bar{\lambda}_1 Ab + \ldots + \bar{\lambda}_{k-1} A^{k-1} b$$

donde $\bar{\lambda}_0 := \lambda_0\lambda_{k-1}, \bar{\lambda}_1 := \lambda_0 + \lambda_1\lambda_{k-1}, \ldots, \bar{\lambda}_{k-1} = \lambda_{k-2} + \lambda_{k-1}^2$.

ii) Multiplicando (4.9) por A^{-1}, se obtiene

$$A^{k-1}b = \lambda_0 A^{-1} b + \lambda_1 b + \ldots + \lambda_{k-1} A^{k-2} b$$

Despejando, se deduce que

$$A^{-1}b = \tilde{\lambda}_0 b + \tilde{\lambda}_1 Ab + \ldots + \tilde{\lambda}_{k-1} A^{k-1} b \in \mathcal{K}_k$$

con $\tilde{\lambda}_0 = \dfrac{-\lambda_1}{\lambda_0}, \tilde{\lambda}_1 = -\dfrac{\lambda_2}{\lambda_0}, \ldots, \tilde{\lambda}_{k-1} = \dfrac{1}{\lambda_0}$. □

En ocasiones los vectores b, Ab, $A^2b, \ldots, A^{k-1}b$ pueden estar muy cerca de ser linealmente dependientes, por lo cual conviene ortonormalizar la base del subespacio de Krylov. Para ello se aplica el método de Gram-Schmidt:

$$\begin{cases}
v_1 := b \in \mathcal{K}_1, & \\
w_1 := v_1, & \beta_1 := \|w_1\|, \\
\hat{w}_1 := \dfrac{w_1}{\beta_1} \in \mathcal{K}_1, & \\
v_2 := A\hat{w}_1 \in \mathcal{K}_2, & \\
w_2 := v_2 - \langle v_2, \hat{w}_1\rangle \hat{w}_1, & \beta_2 := \|w_2\|, \\
\hat{w}_2 := \dfrac{w_2}{\beta_2} \in \mathcal{K}_2, & \\
v_3 := A\hat{w}_2 \in \mathcal{K}_3, & \\
w_3 := v_3 - \langle v_3, \hat{w}_1\rangle \hat{w}_1 - \langle v_3, \hat{w}_2\rangle \hat{w}_2, & \beta_3 := \|w_3\|, \\
\hat{w}_3 := \dfrac{w_3}{\beta_3} \in \mathcal{K}_3, & \\
\vdots & \\
v_{k+1} := A\hat{w}_k \in \mathcal{K}_k, & \\
w_{k+1} := v_{k+1} - \langle v_{k+1}, \hat{w}_{k-1}\rangle \hat{w}_{k-1}, -\langle v_{k+1}, \hat{w}_k\rangle \hat{w}_k - \displaystyle\sum_{j=1}^{k-2} \langle v_k, \hat{w}_j\rangle, & \beta_{k+1} := \|w_{k+1}\|, \\
\hat{w}_{k+1} := \dfrac{w_{k+1}}{\beta_{k+1}} \in \mathcal{K}_k &
\end{cases}$$

Proposición 4.4 *Se tienen las siguientes propiedades para la bases ortonormal del subespacio de Krylov*

i) Para todo $j \leq k-2$ se cumple que $\langle A\hat{w}_k, \hat{w}_j\rangle = 0$

ii) $w_{k+1} := A\hat{w}_k - \langle A\hat{w}_k, \hat{w}_{k-1}\rangle\hat{w}_{k-1} - \langle A\hat{w}_k, \hat{w}_k\rangle\hat{w}_k$

iii) Para todo $j \in \mathbb{N}$ se cumple que $\langle A\hat{w}_{j+1}, \hat{w}_j\rangle = \langle \hat{w}_{j+1}, A\hat{w}_j\rangle = \beta_{j+1}$

DEM:

i)

$$\begin{aligned}\langle A\hat{w}_k, \hat{w}_j\rangle &= \langle \hat{w}_k, A\hat{w}_j\rangle \text{ por simetría de A} \\ &= \|w_{j+1}\|\langle \hat{w}_k, \hat{w}_k\rangle + \sum_{\mathrm{i}=1}^{j}\langle \hat{w}_k, \hat{w}_\mathrm{i}\rangle\langle A\hat{w}_j, \hat{w}_\mathrm{i}\rangle \\ &= 0\end{aligned}$$

donde la última igualdad es por la ortogonalidad de los vectores $\hat{w}_k$, $\hat{w}_\mathrm{i}$ para todo $\mathrm{i} \neq k$.

ii) Directo de i).

iii) Usando ii),

$$\begin{aligned}\langle \hat{w}_{j+1}, A\hat{w}_j\rangle &= \langle \hat{w}_{j+1}, \beta_{j+1}\hat{w}_{j+1} + \langle A\hat{w}_j, A\hat{w}_j\rangle\hat{w}_j + \langle A\hat{w}_j, \hat{w}_{j-1}\rangle\hat{w}_{j-1}\rangle \\ &= \beta_{j+1}\|\hat{w}_{j+1}\| + \langle A\hat{w}_j, \hat{w}_j\rangle\langle \hat{w}_{j+1}, \hat{w}_j\rangle + \langle A\hat{w}_j, \hat{w}_{j-1}\rangle\langle \hat{w}_{j+1}, \hat{w}_{j-1}\rangle \\ &= \beta_{j+1}\end{aligned}$$

donde la última igualdad proviene de la ortnormalidad de los vectores de la base, $\langle \hat{w}_n, \hat{w}_m\rangle = \delta_{nm}$. □

NOTACIÓN: $\alpha_j := \langle A\hat{w}_j, \hat{w}_j\rangle$

Definición 4.5 *La base orthonormal de $\mathcal{K}_k$ queda dada por*

$$\begin{cases} \hat{w}_1 = \dfrac{b}{\beta_1} \\ \beta_2\hat{w}_2 = A\hat{w}_1 - \alpha_1\hat{w}_1 \\ \beta_3\hat{w}_3 = A\hat{w}_2 - \beta_1\hat{w}_1 - \alpha_2\hat{w}_2 \\ \vdots \\ \beta_{k+1}\hat{w}_{k+1} = A\hat{w}_k - \alpha_k\hat{w}_k - \beta_{k-1}\hat{w}_{k-1} \end{cases}$$

Definición 4.6 $V_k = (\hat{w}_1|\hat{w}_2|\cdots|\hat{w}_k) \in \mathcal{M}_{\mathrm{d},k}$,

Proposición 4.7 *El producto matricial AV_k se puede escribir como*

$$AV_k = V_kT_k + \beta_{k+1}\hat{w}_{k+1}\mathrm{e}_k^T \tag{4.11}$$

donde

$$T_k = \begin{pmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 & 0 \\ \beta_2 & \alpha_2 & \beta_3 & \cdots & 0 & 0 \\ 0 & \beta_3 & \ddots & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & & \ddots & \alpha_{k-1} & \beta_k \\ 0 & 0 & \cdots & \cdots & \beta_k & \alpha_k \end{pmatrix}$$

DEM:

$$\begin{aligned} AV_k &= [\ A\hat{w}_1 \mid A\hat{w}_2 \mid \cdots \mid A\hat{w}_k\ ] \\ &= [\ \hat{w}_1\alpha_1 + \hat{w}_2\beta_2 \mid \hat{w}_1\beta_2 + \hat{w}_2\alpha_2 + \hat{w}_3\beta_3 \mid \cdots \mid \hat{w}_{k-1}\beta_k + \hat{w}_k\alpha_k + \hat{w}_{k+1}\beta_{k+1}\ ] \\ &= [\ \hat{w}_1 \mid \hat{w}_2 \mid \cdots \mid \hat{w}_k \mid \hat{w}_{k+1}\ ]_{\mathrm{d},k+1} \begin{pmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 & 0 \\ \beta_2 & \alpha_2 & \beta_3 & \cdots & 0 & 0 \\ 0 & \beta_3 & \ddots & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & & \ddots & \alpha_{k-1} & \beta_k \\ 0 & 0 & \cdots & \cdots & \beta_k & \alpha_k \\ 0 & 0 & \cdots & \cdots & 0 & \beta_{k+1} \end{pmatrix}_{k+1,k} \\ &= V_kT_k + \beta_{k+1}\hat{w}_{k+1}\mathrm{e}_k^T \qquad \square. \end{aligned}$$

OBSERVACIÓN: La matriz T_k es simétrica.

El algoritmo MINRES resuelve el sistema lineal mediante una solución aproximada que minimiza el residuo:

$$\begin{cases} \text{mín}\, r_{k+1} := \|Ax_{k+1} - b\| \\ \text{s.a. } x_{k+1} \in \mathcal{K}_{k+1} \end{cases}$$

Como se busca la solución aproximada en un subespacio de Krylov, $x_k \in \mathcal{K}_k$, esta se puede escribir de la siguiente manera

$$x_k = V_k y$$

donde $y \in \mathbb{R}^k$, por lo cual el problema se traduce en resolver

$$\begin{cases} \text{mín}\, \|AV_{k+1}y_{k+1} - b\|^2 \\ \text{s.a. } y \in \mathbb{R}^{k+1} \end{cases}$$

Al ser un problema de minimización, se utiliza la regla de Fermat para llegar a alguna condición necesaria. Por comodidad y para reducir notación, se define $B := AV_k$, que por (4.11) se tiene $B = V_kT_k + \beta_{k+1}\hat{w}_{k+1}\mathrm{e}_k^T$

$$\begin{aligned} \|AV_ky - b\|^2 &= \|By - b\|^2 \\ &= \langle By, By\rangle - 2\langle By, b\rangle + \|b\|^2 \end{aligned}$$

Considerando la función $f : \mathbb{R}^k \to \mathbb{R}$, $f(y) = \langle By, By\rangle - 2\langle By, b\rangle$, e imponiendo $\nabla f(y) = 0$, se obtiene

$$B^T By = B^T b \tag{4.12}$$

La condición de mínimo es:

$$\begin{aligned} B^T By = B^T b \quad &\iff \quad V_k^T AAV_k y = V_k^T Ab \\ &\iff \quad (T_k V_k^T + \beta_{k+1} \mathrm{e}_k \hat{w}_{k+1}^T)(V_k T_k + \beta_{k+1} \hat{w}_{k+1} \mathrm{e}_k^T) y = (T_k V_k^T + \beta_{k+1} \mathrm{e}_k \hat{w}_{k+1}^T)\beta_1 \hat{w}_1 \\ &\iff \quad (T_k T_k + \beta_{k+1}^2 \mathrm{e}_k \mathrm{e}_k^T) y = \beta_1 T_k \mathrm{e}_1 \end{aligned} \tag{4.13}$$

El problema de la última expresión es que T_k es simétrica pero no necesariamente invertible, por lo cual la idea es hacer una descomposición LQ.

Definición 4.8 *Sean $Q_1, \ldots, Q_k$ matrices ortogonales elementales tales que el producto $T_k Q_1 Q_2 \cdots Q_{k-1}$ sea una matriz triangular inferior.*

Las matrices ortogonales no pueden ser cualquiera, sino que los coeficientes deben ser tales de ir formando un producto triangular superior.

$$Q_1 = \begin{pmatrix} c_1 & s_1 & 0 & \cdots & 0 & 0 \\ s_1 & -c_1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & & \ddots & 1 & 0 \\ 0 & 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}$$

con $c_1 = cos(\theta)$, $s_1 = sen(\theta)$. Entonces

$$T_k Q_1 = \begin{pmatrix} c_1\alpha_1 + s_1\beta_2 & \boxed{s_1\alpha_1 - c_1\beta_2} & 0 & \cdots & 0 & 0 \\ c_1\beta_2 + s_1\alpha_2 & s_1\beta_2 - c_1\alpha_2 & \beta_3 & \cdots & 0 & 0 \\ s_1\beta_3 & -c_1\beta_3 & \ddots & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & & \ddots & \alpha_{k-1} & \beta_k \\ 0 & 0 & \cdots & \cdots & \beta_k & \alpha_k \end{pmatrix}$$

s_1 y c_1 se escogen de modo que $s_1\alpha_1 - c_1\beta_2 = 0$, luego $\tan(\theta) = \dfrac{\beta_2}{\alpha_1}$, y por lo tanto

$$\begin{aligned} c_1 &= \frac{\alpha_1}{\sqrt{\alpha_1^2 + \beta_2^2}} \\ s_1 &= \frac{\beta_2}{\sqrt{\alpha_1^2 + \beta_2^2}} \end{aligned}$$

Así,

$$T_kQ_1 = \begin{pmatrix} \gamma_1 := \sqrt{\alpha_1^2+\beta_2^2} & 0 & 0 & 0 & \cdots & 0 & 0 \\ \delta_2 := c_1\beta_2 + s_1\alpha_2 & \overline{\gamma_2} := s_1\beta_2 - c_1\alpha_2 & \beta_3 & 0 & \cdots & 0 & 0 \\ \varepsilon_3 := s_1\beta_3 & \overline{\delta_3} := -c_1\beta_3 & \alpha_3 & \beta_4 & \ddots & & \\ 0 & 0 & \beta_4 & \alpha_4 & & 0 & 0 \\ \vdots & \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & & & \ddots & \alpha_{k-1} & \beta_k \\ 0 & 0 & \cdots & & \cdots & \beta_k & \alpha_k \end{pmatrix}$$

Del mismo modo

$$T_kQ_1Q_2 = \begin{pmatrix} \gamma_1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \delta_2 & \gamma_2 := \sqrt{\overline{\gamma_2}^2+\beta_3^2} & 0 & 0 & \cdots & 0 & 0 \\ \varepsilon_3 & \delta_3 := c_2\overline{\delta_3} + s_2\alpha_3 & \overline{\gamma_3} := s_2\overline{\delta_3} - c_2\alpha_3 & \beta_4 & \ddots & & \\ 0 & \varepsilon_4 := s_2\beta_4 & \overline{\delta_4} := -c_2\beta_4 & \alpha_4 & & 0 & 0 \\ \vdots & \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & & & \ddots & \alpha_{k-1} & \beta_k \\ 0 & 0 & \cdots & & \cdots & \beta_k & \alpha_k \end{pmatrix}$$

con

$$c_2 = \frac{\overline{\gamma_2}}{\sqrt{\overline{\gamma_2}^2+\beta_3^2}}, s_2 = \frac{\beta_3}{\sqrt{\overline{\gamma_2}^2+\beta_3^2}}$$

Proposición 4.9 (**Descomposición LQ**)

$$\overline{L_k} := T_kQ_1Q_2\cdots Q_{k-1} = \begin{pmatrix} \gamma_1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \delta_2 & \gamma_2 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \varepsilon_3 & \delta_3 & \gamma_3 & 0 & \ddots & & & \\ 0 & \varepsilon_4 & \delta_4 & \gamma_4 & & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & & & \\ \vdots & \vdots & & \ddots & \ddots & \ddots & \ddots & \\ 0 & 0 & & & \ddots & \delta_{k-1} & \gamma_{k-1} & 0 \\ 0 & 0 & \cdots & & \cdots & \varepsilon_k & \delta_k & \overline{\gamma_k} \end{pmatrix} \tag{4.14}$$

con

$$\gamma_k = \sqrt{\overline{\gamma_k}^2+\beta_{k+1}^2}, \quad c_k = \frac{\overline{\gamma_k}}{\gamma_k}, \quad s_k = \frac{\beta_{k+1}}{\gamma_k}$$

Además, se cumple que $T_k = \overline{L_k}Q^T$, *con la notación* $Q := Q_1Q_2\cdots Q_{k-1}$.

Definición 4.10 *Se define la matriz L_k como*

$$L_k := \begin{pmatrix} \gamma_1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \delta_2 & \gamma_2 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \varepsilon_3 & \delta_3 & \gamma_3 & 0 & \ddots & & & \\ 0 & \varepsilon_4 & \delta_4 & \gamma_4 & & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & & & \\ \vdots & \vdots & & \ddots & \ddots & \ddots & \ddots & \\ 0 & 0 & & & \ddots & \delta_{k-1} & \gamma_{k-1} & 0 \\ 0 & 0 & \cdots & & \cdots & \varepsilon_k & \delta_k & \gamma_k \end{pmatrix}$$

Las matrices L_k, $\overline{L_k}$ defieren sólo en γ_k.

Proposición 4.11 *L_k y $\overline{L_k}$ cumplen las siguientes relaciones*

1. $\overline{L_k} = L_k D_k$, *con* $D_k = \mathrm{diag}\left(1, 1, \ldots, 1, \frac{\overline{\gamma_k}}{\gamma_k}\right)$
2. $\overline{L_k}\,\overline{L_k}^T = L_k L_k^T + \beta_{k+1}^2 \mathrm{e}_k \mathrm{e}_k^T$

Usando la descomposición LQ,

$$\begin{aligned} T_k T_k &= T_k T_k^T \text{ por simetría de } T_k \\ &= (\overline{L_k} Q^T)(\overline{L_k} Q^T)^T \\ &= \overline{L_k} Q^T Q \overline{L_k}^T \\ &= \overline{L_k}\,\overline{L_k}^T \quad Q^T,\ Q = I \text{ pues son matrices ortogonales} \end{aligned}$$

Reemplazando en (4.13), y usando la Proposición 4.11 se llega a

$$L_k\, L_k^T y = \beta_1 \overline{L_k} Q \mathrm{e}_1$$

Nuevamente gracias a la proposición anterior se tiene que

$$L_k\, L_k^T y = \beta_1 L_k D_k Q \mathrm{e}_1$$

Como L_k es invertible (es triangular inferior y los términos de su diagonal son no nulos),

$$L_k^T y = \beta_1 D_k Q \mathrm{e}_1$$

Finalmente,

$$y = L_k^{-T} \cdot t, \quad \text{donde } t = \beta_1 D_k Q \mathrm{e}_1 \tag{4.15}$$

Recordando que la solución viene dada por $x_k = V_k y$, reemplazando (4.15)

$$x_k = V_k L_k^{-T} t \tag{4.16}$$

Definiendo $M := V_k L_k^{-T} = [\ \overrightarrow{m}_1 \mid \overrightarrow{m}_2 \mid \cdots \mid \overrightarrow{m}_k\ ]$, basta calcular el producto entre M y t para determinar la solución aproximada. Sin embargo, conviene aprovechar la estructura

incremental de x_k, pues es más eficiente computacionalmente.
Suponiendo que se conoce $M_k^{(k)}$, que es solucion de $M_k^{(k)} L_k^T = V_k$. Entonces,

$$\begin{bmatrix} M_k^{(k+1)} & \overrightarrow{m}_{k+1} \end{bmatrix} \begin{pmatrix} & L_k^T & & \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \varepsilon_{k+1} \\ \delta_{k+1} \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix} & & & \gamma_{k+1} \end{pmatrix} = [V_k \mid \hat{w}_{k+1}]$$

$$\iff \quad M_k^{(k+1)} L_k^T + \overrightarrow{m}_{k+1} \cdot 0 = V_k, \quad M_k \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \varepsilon_{k+1} \\ \delta_{k+1} \end{pmatrix} + \overrightarrow{m}_{k+1} \gamma_{k+1} = \hat{w}_{k+1}$$

Se obtiene

$$\overrightarrow{m}_{k+1} = \frac{1}{\gamma_{k+1}} \Big(\hat{w}_{k+1} - \overrightarrow{m}_{k-1} \varepsilon_{k+1} - \overrightarrow{m}_k \delta_{k+1} \Big)$$

Por lo tanto, la solución aproximada del problema viene dada por

$$x_{k+1} = x_k + t_{k+1} \overrightarrow{m}_{k+1} \tag{4.17}$$

Algoritmo 1 Algoritmo MINRES.

1. Calcular el vector $\hat{w}_{k+1}$ de la base ortonormal del subespacio de Krylov
2. Buscar x_{k+1} solución aproximada en el subespacio de Krylov que minimice el residuo:

$$\begin{cases} \text{mín } r_{k+1} := \|Ax_{k+1} - b\| \\ \text{s.a. } x_{k+1} \in \mathcal{K}_{k+1} \end{cases}$$

2.1. Calcular $t_{k+1} = \beta_1 s_1 \cdots s_k c_{k+1}$, con $s_{\text{i}} = \frac{\beta_{\text{i}+1}}{\gamma_{\text{i}}}$y $c_{\text{i}} = \frac{\overline{\gamma_{\text{i}}}}{\gamma_{\text{i}}}$.

2.2. Calcular $\vec{m}_{k+1} = \frac{1}{\gamma_{k+1}}(\hat{w}_{k+1} - \vec{m}_{k-1}\varepsilon_{k+1} - \vec{m}_k \delta_{k+1})$.

2.3. Calcular $x_{k+1} = x_k + t_{k+1}\vec{m}_{k+1}$.

3. Si el residuo r_{k+1} es menor a la tolerancia, entonces $x = x_{k+1}$ es la solución aproximada al sistema lineal.
   Si no, hacer $k \leftarrow k+1$ y volver a 1.

4.1.3. Discusión sobre el residuo en MINRES

Recordar que MINRES minimiza el residuo $r_n = \|Ax_n - b\|$ sobre el subespacio de Krylov $\mathcal{K}_n$, los cuales cumplen $\mathcal{K}_1 \subset \mathcal{K}_2 \ldots$, luego los residuos son no crecientes $r_1 > r_2 > \cdots$.

Transcurridas d iteraciones con d el tamaño de la matriz A, MINRES encuentra la solución exacta, pues ahí $\mathcal{K}_{\mathrm{d}} = \mathbb{R}^{\mathrm{d}}$, entonces $r_{\mathrm{d}} = 0$. Sin embargo, la idea es no realizar tantas iteraciones (en la geometría de Quellón $\mathrm{d} = 3N + M \geq 50000$). El objetivo es hallar una solución aproximada con residuo pequeño.
¿Cómo calcular los residuos? Una primera opción es reemplazar el x_n obtenido en la $n-$ésima iteración en $r_n = \|Ax_n - b\|^2$, no obstante, multiplicar la matriz A es muy costoso, así que es mejor explicitar aún más el residuo usando la teoría.
Como $x_n \in \mathcal{K}_n$, $x_n = V_n y$ con $y \in R^n$ el vector de ponderadores reales. Luego,

$$\begin{aligned}
\|Ax_n - b\| &= \|AV_n - \beta_1 \hat{w}_1\| \\
&= \|(V_n T_n + \beta_{n+1} \mathrm{e}_n^T \hat{w}_{n+1}) y - \beta_1 \hat{w}_1\| \\
&= \left\| (V_n \mid \hat{w}_{n+1}) \begin{pmatrix} T_n y \\ \beta_{n+1} \mathrm{e}_n^T \end{pmatrix} - \beta_1 \hat{w}_1 \right\| \\
&= \left\| V_{n+1} \begin{pmatrix} T_n \\ \beta_{n+1} \mathrm{e}_n^T \end{pmatrix} y - \beta_1 V_{n+1} \mathrm{e}_1^{(n+1)} \right\| \\
&= \left\| V_{n+1} \left(\begin{pmatrix} T_n \\ \beta_{n+1} \mathrm{e}_n^T \end{pmatrix} y - \beta_1 \mathrm{e}_1^{(n+1)} \right) \right\| \\
&= \left\langle V_{n+1} \left(\begin{pmatrix} T_n \\ \beta_{n+1} \mathrm{e}_n^T \end{pmatrix} y - \beta_1 \mathrm{e}_1^{(n+1)} \right), V_{n+1} \left(\begin{pmatrix} T_n \\ \beta_{n+1} \mathrm{e}_n^T \end{pmatrix} y - \beta_1 \mathrm{e}_1^{(n+1)} \right) \right\rangle^{\frac{1}{2}} \\
&= \left\langle \left(\begin{pmatrix} T_n \\ \beta_{n+1} \mathrm{e}_n^T \end{pmatrix} y - \beta_1 \mathrm{e}_1^{(n+1)} \right), V_{n+1}^T V_{n+1} \left(\begin{pmatrix} T_n \\ \beta_{n+1} \mathrm{e}_n^T \end{pmatrix} y - \beta_1 \mathrm{e}_1^{(n+1)} \right) \right\rangle^{\frac{1}{2}} \\
&= \left\| \begin{pmatrix} T_n \\ \beta_{n+1} \mathrm{e}_n^T \end{pmatrix} y - \beta_1 \mathrm{e}_1^{(n+1)} \right\|^2, \quad V_{n+1}^T V_{n+1} = I \text{ por ser matrices ortonormales} \\
&= \|QR_{n+1,n} y - \beta_1 Q Q^T \mathrm{e}_1^{(n+1)}\|, \quad \text{haciendo descomposición QR y usando } QQ^T = I
\end{aligned}$$

Luego el mínimo se puede calcular sobre $\|R_{n+1,n} y - \beta_1 Q^T \mathrm{e}_1^{(n+1)}\|$, y escribiendo $R_{n+1,n} = \begin{pmatrix} R \\ 0 \end{pmatrix}$ con R una matriz cuadrada de $n \times n$, se obtiene que

$$r_n^{\text{MINRES}} = \left\| \begin{pmatrix} R \\ 0 \end{pmatrix} y - \begin{pmatrix} Q_{1,1}\beta_1 \\ \vdots \\ Q_{1,n+1}\beta_1 \end{pmatrix} \right\|$$

al no haber restricción sobre y en la última fila, se sigue que

$$r_n^{\text{MINRES}} = \beta_1 Q_{1,n+1} = \beta_1 s_1 \cdots s_n$$

Que es mucho más eficiente de calcular, pues no requiere hacer la multiplicación matricial Ax_n, además de ser incremental (se guarda el residuo, y en la etapa siguiente simplemente se multiplica por s_{n+1}).

4.2. Ecuación discretizada de convección-difusión

Integrando la EDP de convección-difusión (3.1) sobre un volumen $\omega_\mathrm{i} \subset \Omega$, con $\mathrm{i} \in \{1, \ldots, N\}$ se obtiene que:

$$\int_{\omega_\mathrm{i}} \frac{\partial c}{\partial t} \mathrm{d}\omega + \int_{\omega_\mathrm{i}} \mathrm{div}(cu) \mathrm{d}\omega - \int_{\omega_\mathrm{i}} \mathrm{div}(D\nabla c) \mathrm{d}\omega = \int_{\omega_\mathrm{i}} F \mathrm{d}\omega.$$

Por hipótesis del Método de los Volúmenes Finitos, la concentración c y el término fuente son constantes en espacio para cada volumen finito, es decir, $\forall t > 0, \forall x \in \omega_\mathrm{i}$,

$$\begin{aligned} c(x,t) &= c_\mathrm{i}(x,t) =: c_\mathrm{i}^k \\ F(x,t) &= F_\mathrm{i}(t) =: F_\mathrm{i}^k \end{aligned}$$

en que el subíndice i denota el valor de la función en el volumen finito ω_i y el superíndice k denota la etapa temporal $k \in \mathbb{N}$. A continuación se discretiza cada término:

- Término de evolución:

$$\begin{aligned} \int_{\omega_\mathrm{i}} \frac{\partial c}{\partial t} \mathrm{d}\omega &= \frac{\partial c_\mathrm{i}(t)}{\partial t} |\omega_\mathrm{i}| \\ &\approx \frac{c_\mathrm{i}^k - c_\mathrm{i}^{k-1}}{\Delta t} |\omega_\mathrm{i}| \end{aligned}$$

- Término convectivo:

$$\begin{aligned} \int_{\omega_\mathrm{i}} \mathrm{div}(cu) \mathrm{d}\omega &= \int_{\partial\omega_\mathrm{i}} c(u \cdot n) \, \mathrm{d}s \\ &\approx \sum_{j:\Gamma_j \text{ cara de } \omega_\mathrm{i}} c^k(\Gamma_j)(u^k \cdot n)(\Gamma_j) \, |\Gamma_j| \end{aligned}$$

  Para definir bien el término de la concentración en las caras internas $\Gamma_j = \partial\omega_\mathrm{i} \cap \partial\omega_j$, nuevamente se utiliza la condición de *upwind*:

$$c^k(\Gamma_j) = \begin{cases} c_\mathrm{i}^k & \text{si } (u^k \cdot n)(\Gamma_j) \geq 0 \\ c_j^k & \text{si } (u^k \cdot n)(\Gamma_j) < 0 \end{cases}$$

  y siguiendo un procedimiento análogo a (4.1), (4.2) y (4.3) se obtiene

$$\int_{\omega_\mathrm{i}} \mathrm{div}(cu) \mathrm{d}\omega \approx \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara interna} \\ (u^k \cdot n)(\Gamma_j) < 0}} (c_j^k - c_\mathrm{i}^k)(u^k \cdot n)(\Gamma_j) \, |\Gamma_j|$$

- Término difusivo:

$$\begin{aligned} \int_{\omega_\mathrm{i}} \mathrm{div}(D\nabla c) \mathrm{d}\omega &= \int_{\partial\omega_\mathrm{i}} D\nabla c \cdot n \mathrm{d}s \\ &\approx \sum_{j:\Gamma_j \text{ cara de } \omega_\mathrm{i}} D \frac{(c^k(\Gamma_j) - c_\mathrm{i}^k)}{\mathrm{d}_c(\omega_\mathrm{i}, \omega_j)} \, |\Gamma_j| \end{aligned}$$

- Término fuente:

$$\int_{\omega_\mathrm{i}} F \mathrm{d}\omega \approx F_\mathrm{i}^k \left|\omega_\mathrm{i}\right|$$

Sumando cada término, se obtiene la ecuación de convección-difusión discretizada

$$\left(\frac{\left|\omega_\mathrm{i}\right|}{\Delta t} + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara interna} \\ (u^k \cdot n)(\Gamma_j)<0}} (-u^k \cdot n)(\Gamma_j) \left|\Gamma_j\right| + \sum_{j:\Gamma_j \text{ cara de } \omega_\mathrm{i}} \frac{D \left|\Gamma_j\right|}{\mathrm{d}_c(\omega_\mathrm{i}, \omega_j)} \right) c_\mathrm{i}^k$$
$$- \left(\sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara interna} \\ (u^k \cdot n)(\Gamma_j)<0}} (-u^k \cdot n)(\Gamma_j) \left|\Gamma_j\right| c_j^k + \sum_{j:\Gamma_j \text{ cara de } \omega_\mathrm{i}} \frac{D \left|\Gamma_j\right|}{\mathrm{d}_c(\omega_\mathrm{i}, \omega_j)} c^k(\Gamma_j) \right) = \frac{\left|\omega_\mathrm{i}\right|}{\Delta t} c_\mathrm{i}^{k-1} + F_\mathrm{i}^k \left|\omega_\mathrm{i}\right| \tag{4.18}$$

Para las caras internas es recomendable usar $u^k(\Gamma_j) = \dfrac{u_\mathrm{i}^k + u_j^k}{2}$. Al igual que en la sección anterior, para resolver las ecuaciones discretizadas se puede plantear el sistema matricial, pero no es necesario también pues se puede usar el Método de Jacobi resolviendo el sistema de ecuaciones con un método iterativo. Las matrices asociadas a la ecuación de convección-difusión están dadas por

$$\begin{aligned}
A_{\mathrm{ii}} &= \frac{\left|\omega_\mathrm{i}\right|}{\Delta t} + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_\mathrm{i} \\ \Gamma_j \text{ cara interna} \\ (u^k \cdot n)(\Gamma_j)<0}} -u^k \cdot n(\Gamma_j) \left|\Gamma_j\right| + \sum_{j:\Gamma_j \text{ cara de } \omega_\mathrm{i}} \frac{D \left|\Gamma_j\right|}{\mathrm{d}_c(\omega_\mathrm{i}, \omega_j)} \\
A_{\mathrm{i}j} &= u^k \cdot n(\Gamma_j) \left|\Gamma_j\right| - \frac{D \left|\Gamma_j\right|}{\mathrm{d}_c(\omega_\mathrm{i}, \omega_j)} \text{ si } \omega_j \text{ es vecino de } \omega_\mathrm{i} \text{ y } u^k \cdot n(\Gamma_j) < 0 \\
A_{\mathrm{i}j} &= -\frac{D \left|\Gamma_j\right|}{\mathrm{d}_c(\omega_\mathrm{i}, \omega_j)} \text{ si } \omega_j \text{ es vecino de } \omega_\mathrm{i} \text{ y } u^k \cdot n(\Gamma_j) \geq 0 \\
b_\mathrm{i} &= \frac{\left|\omega_\mathrm{i}\right|}{\Delta t} c_\mathrm{i}^{k-1} + F_\mathrm{i}^k \left|\omega_\mathrm{i}\right|
\end{aligned}$$

y el resto de términos son nulos. Donde para el lado derecho está implicíta la condición de derivada normal nula, $\dfrac{c^k(\Gamma_j) - c_\mathrm{i}^k}{\mathrm{d}_c(\omega_\mathrm{i}, \omega_j)} = 0$.

4.2.1. Condiciones de borde ecuación de convección-difusión

Dada la hipótesis de no-transferencia de microalgas en los bordes (pues la cantidad de concentración es la misma en el dominio), la condición de borde es tipo Neumann. Si se considera la creación/destrucción de microalgas o transferencia de microalgas en Mar-Mar cambia la condición de borde.

Regiones del borde	Condición sobre c
Fondo	$\frac{\partial c}{\partial n} = 0$
Superficie	$\frac{\partial c}{\partial n} = 0$
Mar-Tierra	$\frac{\partial c}{\partial n} = 0$
Mar-Mar	$\frac{\partial c}{\partial n} = 0$

Tabla 4.2: Tabla resumen condiciones de borde para la ecuación de convección-difusión de microalgas.

4.2.2. Método de Jacobi

El Método de Jacobi es un algoritmo iterativo para determinar la solución de un sistema de ecuaciones lineales estrictamente diagonal dominante.
Sea $A \in \mathcal{M}_{\mathrm{d}\times\mathrm{d}}$, $b \in \mathbb{R}^{\mathrm{d}}$. Se quiere resolver $Ax = b$.
Para ello, se descompone la matriz A en su diagonal y la matriz restante

$$A = D + R$$

donde

$$D = \begin{pmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & a_{\mathrm{dd}} \end{pmatrix}, \quad R = \begin{pmatrix} 0 & a_{12} & \dots & a_{1\mathrm{d}} \\ a_{21} & 0 & \dots & a_{2\mathrm{d}} \\ \vdots & & \ddots & \vdots \\ a_{\mathrm{d}1} & \dots & a_{\mathrm{d}(\mathrm{d}-1)} & 0 \end{pmatrix}$$

La solución aproximada se obtiene iterativamente como

$$x^{(n+1)} = D^{-1}(b - Rx^{(n)}) \tag{4.19}$$

donde $x^{(n)}$ es la $n-$ésima aproximación de la solución x.
Por componentes, la solución viene dada por

$$x_{\mathrm{i}}^{(n+1)} = \frac{1}{a_{\mathrm{ii}}} \left(b_{\mathrm{i}} - \sum_{j \neq \mathrm{i}} a_{\mathrm{i}j} x_j^{(n)} \right), \quad \forall \mathrm{i} = 1, 2, \dots, \mathrm{d} \tag{4.20}$$

El siguiente Teorema establece un criterio de convergencia para el Método de Jacobi.

Teorema 4.12 *Si A es estrictamente diagonal dominante, entonces la solución aproximada $x^{(n)}$ converge a la solución exacta $x = A^{-1}b$ cuando $n \to \infty$.*

La demostración se puede encontrar en [1].
Al usar el Método de Jacobi para resolver (4.18), hay dos etapas: la temporal que se denota k y también es necesario introducir otro superíndice, para las iteraciones de Jacobi, ℓ.

$$c_{\mathrm{i}}^{k,(\ell+1)} = \frac{\frac{|\omega_{\mathrm{i}}|}{\Delta t} c_{\mathrm{i}}^{k-1} + F_{\mathrm{i}}^{k} \, |\omega_{\mathrm{i}}| + \left(\sum_{\substack{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}} \\ \Gamma_j \text{ cara interna} \\ (u^k \cdot n)(\Gamma_j) < 0}} (-u^k \cdot n)(\Gamma_j) \, |\Gamma_j| \, c_j^{k,(\ell)} + \sum_{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}}} \frac{D}{\mathrm{d}_c(\omega_{\mathrm{i}}, \omega_j)} c_j^{k,(\ell)} \right)}{\left(\frac{|\omega_{\mathrm{i}}|}{\Delta t} + \sum_{\substack{j:\Gamma_j \text{ cara de } \omega_{\mathrm{i}} \\ \Gamma_j \text{ cara interna} \\ (u^k \cdot n)(\Gamma_j) < 0}} (-u^k \cdot n)(\Gamma_j) \, |\Gamma_j| + \sum_{j:\Gamma_j \text{ es cara de } \omega_{\mathrm{i}}} \frac{D}{\mathrm{d}_c(\omega_{\mathrm{i}}, \omega_j)} \right)}$$

El criterio para terminar las iteraciones de Jacobi (en cada tiempo) es fijando una tolerancia: $\|c^{k,(\ell+1)} - c^{k,(\ell)}\| < \varepsilon$ y con un número máximo de iteraciones, de modo que la que se cumpla primero termine el loop del Método de Jacobi.
Observación: En las ecuaciones discretizadas de Navier-Stokes no se puede usar el Método de Jacobi pues los términos de la diagonal de la matriz A asociados a las presiones son nulos.

4.3. Comando mldivide

En un principio estos métodos se implementaron en C++, sin embargo, el método MINRES nunca convergió, el residuo no es lo suficientemente pequeño, y como se deben realizar varias iteraciones temporales, además de que en cada iteración temporal están las iteraciones no lineales esto vuelve inviable el uso del método MINRES. Esto motivo a buscar una opción alternativa para resolver el sistema lineal: el comando *mldivide* en MATLAB, o también conocido como \, el cual está basado en UMFPACK de C++, con algoritmos que resuelven sistemas lineales con desempeño optimizado.
Para comparar los métodos implementados, se usa simplemente el operador de Stokes (sin considerar el término convectivo), cuya resolución numérica es más simple. Usando parámetros $\nu = 0{,}01$ la viscosidad, $A = 2$ la amplitud y $t = 0{,}5$ s el tiempo a evaluar en la velocidad vertical sinusoidal en la condición de borde de marea, se obtiene la siguiente tabla

	mldivide	minres
Residuo	1.15189e-15	1.28184 e-2
Tiempo [s]	37	381

Tabla 4.3: Comparación de resolución sistemas lineales para Stokes estacionario.

donde se fijó un máximo de 5000 iteraciones en minres. También se obtiene el siguiente gráfico que describe la relación entre el tiempo de ejecución de Minres (que aumenta según aumenta el máximo de iteraciones) con el residuo

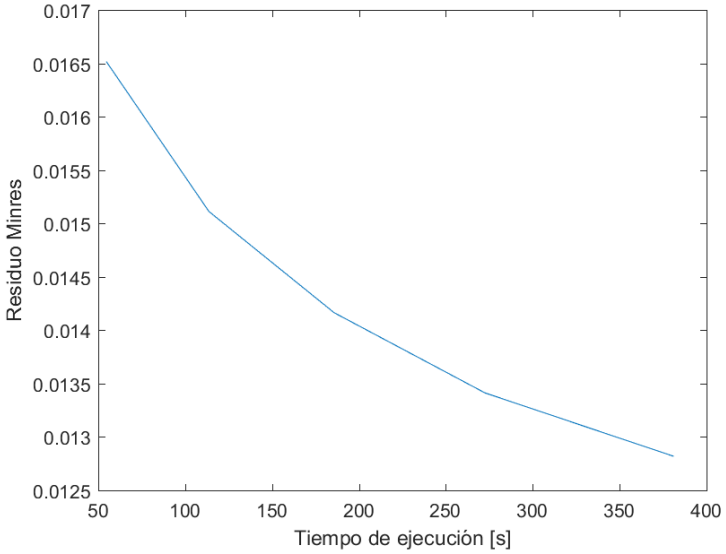
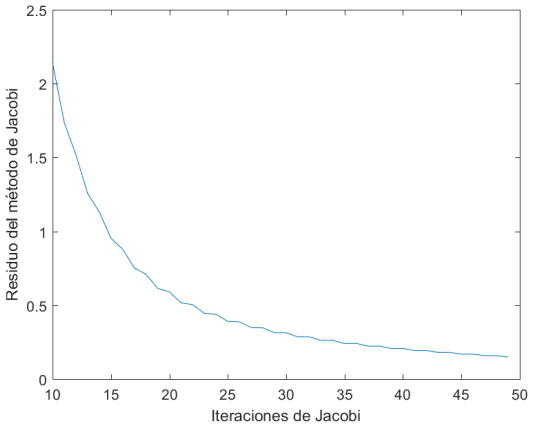
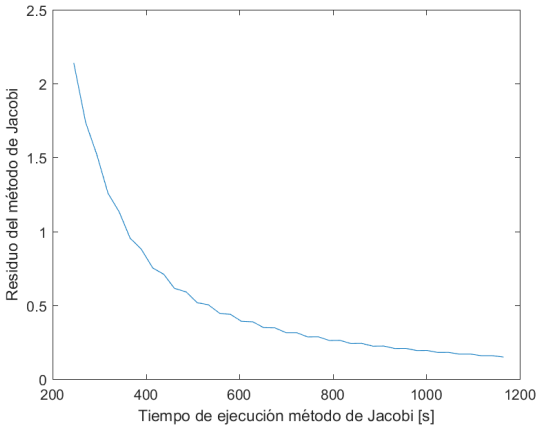


Figura 4.1: Gráfico Tiempo de ejecución MINRES (usando máximos de iteraciones 1000,2000,3000,4000,5000) vs Residuos.

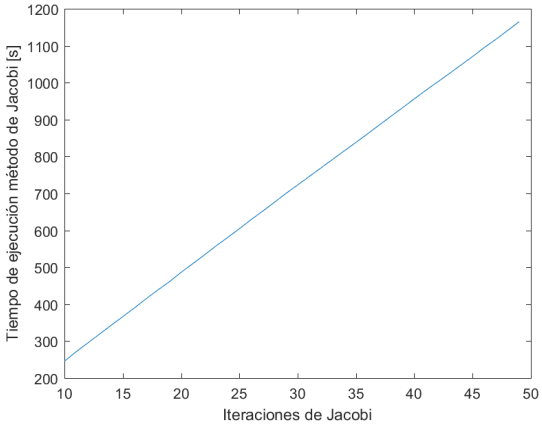
Para el caso de la ecuación de convección-difusión, usando las velocidades obtenidas en Navier-Stokes y coeficiente de difusión $D = 0{,}03$, $\Delta t = 30$ minutos



(a) Iteraciones de Jacobi vs Residuo.



(b) Tiempo de ejecución [s] vs Residuo.



(c) Iteraciones de Jacobi vs Tiempo de ejecución [s].

Figura 4.2: Gráficos de desempeño del método de Jacobi.

	mldivide	Jacobi
Residuo	2.7242e-12	1.48 e-1
Tiempo [s]	3	1165

Tabla 4.4: Comparación de resolución sistemas lineales para la ecuación de convección-difusión.

En ambos casos es más eficiente el uso del comando *mldivide* que justifica usar MATLAB.

Capítulo 5

OpenFOAM

5.1. Estructura de OpenFOAM

Para las implementaciones numéricas se utiliza el software OpenFOAM (Open Source Field Operation and Manipulation). OpenFOAM es un software libre basado en el lenguaje de programación de C++, en donde se resuelven numéricamente problemas de mecánica de medios continuos, en particular CFD (Computational Fluid Dynamics).
En OpenFOAM se pueden implementar modelos matemáticos que usen EDP's, es decir, la EDP asociada, las condiciones de borde, condiciones iniciales, parámetros, etc. Además, OpenFOAM tiene pre-procesamiento en donde se pueden generar mallas y para el post-procesamiento se puede utilizar ParaView, el cual permite visualizar la solución obtenida en formato OpenFOAM y muchos otros formatos. En OpenFOAM se trabaja en un *case*, que es

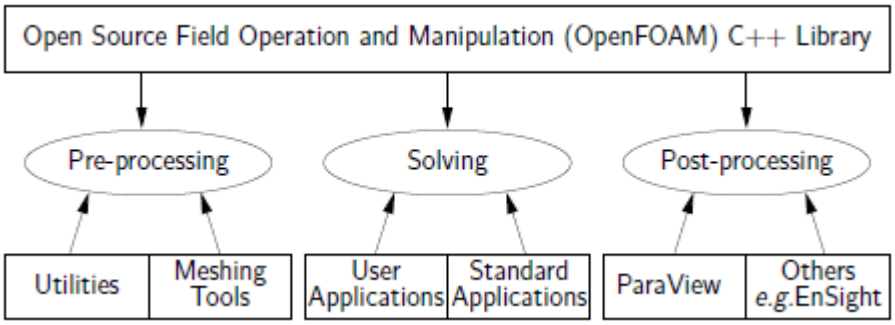


Figura 5.1: Estructura de OpenFOAM. Fuente: OpenFOAM v7 User Guide, https://cfd.direct/openfoam/user-guide.

digamos coloquialmente el directorio. Éste cuenta con una estructura básica, que se muestra en la figura 5.2.

- *system* en que se indica la EDP asociada, el intervalo de tiempo en que resuelve la ecuación y Δt (*controlDict*), se detalla la discretización de cada término de la ecuación (*fvSchemes*), las variables de la ecuación y con qué algoritmo se resuelven (*fvSolution*).

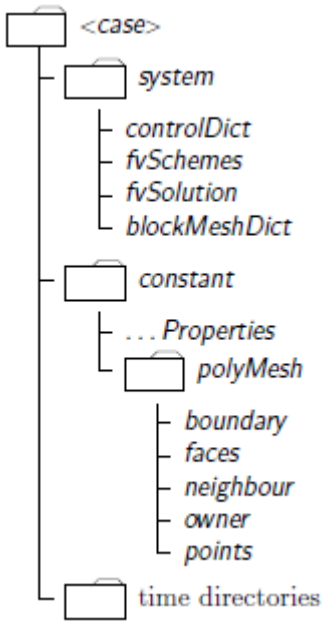


Figura 5.2: Case en OpenFOAM. Fuente: OpenFOAM v7 User Guide, https://cfd.direct/openfoam/user-guide.

Para mallas estructuradas, con geometrías básicas como cuadrados, rectángulos, se puede usar *blockMeshDict* para generar los volúmenes finitos.

- *constant* donde se establecen propiedades físicas (como la viscosidad del fluido) y las características del mallado en los archivos dentro de la carpeta *polyMesh*: *points*, *faces*, *boundary*, *owner*, *neighbour*. Estos archivos determinan completamente la malla con sus volúmenes finitos, estos se explicaran en profundidad al momento de detallar la generación de la malla.
- *time directories* en que se agregan condiciones iniciales y se guardan las soluciones en los tiempos que se indican, en 0/ van las condiciones iniciales de cada variable, junto con sus condiciones de borde. En el caso de Navier-Stokes es $0/U$ y $0/p$. Al momento de correr el solver, se comienzan a guardar las soluciones en los tiempos indicados, en que para cada campo aparecen sus valores cada volumen finito.

Para más referencias, consultar en [5].

5.2. Pre-procesado: generación de la malla

Para generar el dominio, se necesitan datos de la bahía Quellón: los puntos espaciales en coordenadas x, y, z. Estos datos se leyeron en MATLAB, para posteriormente generar la matriz de puntos y de prismas, cuyas bases son triángulos, tiene seis puntos y cinco caras. Para pasar la malla a formato OpenFOAM, se debe tener claro el *case* **polyMesh** que es la carpeta donde se almacena la información de la malla. Recordar que como OpenFOAM trabaja con volúmenes finitos es importante hacer la distinción de cara externa (perteneciente al borde) de cara interna (no perteneciente al borde), en cuyo caso conecta a dos volúmenes

finitos. Por esto, es necesario distinguir que volumen finito es *neighbour* y cual es *owner* de dicha cara interna.

points es la lista de vectores en $\mathbb{R}^3$, que representan cada punto espacial del dominio. Los puntos se numeran desde 0, es decir, el primer vector de esta lista es el nodo 0.
faces es la lista de caras de la malla, en que cada cara está se determina por numeración de sus vértices en la lista *points*.
owner es la lista de caras, en que se indica a qué volumen finito pertenece cada cara, es decir, la primera entrada es el número del volumen finito al que pertenece la cara 0, la segunda entrada es la numeración del volumen finito en el cual se encuentra la cara 1. Incluye tanto caras internas como externas.
neighbour es la lista de caras en que cada entrada es la numeración del volumen finito vecino al volumen finito en *owner* de cada cara interna. Sólo considera caras internas.
boundary son las regiones que componen el borde del dominio, en que cada región es un diccionario, en el cual se indica el tipo de región, la cara en cual comienza y el número de caras de la región.
Al momento de numerar las caras, se deben numerar las caras internas y luego las caras externas. En las caras externas van de manera seguida las caras de cada región, es decir, primero las caras de la primera región de borde, luego las caras del segunda región de borde, y así. Además, en cada volumen finito se numeran las caras internas en orden creciente según la numeración de los volúmenes finitos vecinos; esto se conoce como *UpperTriangularOrder*.
A continuación se muestra una descripción general de la generación de la malla en formato OpenFOAM y luego se procede a explicar en detalle cada parte.

Algoritmo 2 Generación de la malla en formato OpenFOAM

1: Convertir los puntos y prismas del archivo de datos de la bahía Quellón a matrices en MATLAB.
2: Recorrer cada prisma, guardando las caras, el owner y el neighbour de cada cara.
3: Identificar caras internas y caras externas. Eliminar las caras repetidas.
4: Transformar la numeración de caras según los nodos efectivamente usados.
5: Ordenar según *UpperTriangularOrder*.
6: Imprimir los puntos, caras recibiendo la etiqueta de cada punto, los owners, neighbours y el borde en sus respectivos archivos.

1. Convertir los puntos y prismas del archivo de datos de la bahía Quellón a matrices en MATLAB:

   Los datos de la Bahía Quellón recibidos se muestran en 5.3 y 5.4 respectivamente.

   Para la bahía de Quellón, en el eje Z la dimensión (en metros) va desde $-87{,}3912m$ a $0m$, mientras que en el eje X, la dimensión va desde $0m$ to $22590{,}9m = 22{,}59km$ y en el eje Y va de $0m$ a $25414{,}6m = 25{,}414km$, por lo cual el tamaño relativo del espesor del mar en el dominio es pequeño respecto al largo y ancho.
   La tabla 5.1 resume los mínimos y máximos en cada coordenada.

   En la implementación, es importante notar que las primeras cuatro filas de la matriz de puntos tienen el mismo x e y, varían sólo en la coordenada z. Este patrón se repite

	1	2	3	4
1	0	0	1.9544e+04	-17.4375
2	1	0	1.9544e+04	-15.6927
3	2	0	1.9544e+04	-1.7347
4	3	0	1.9544e+04	0.0100
5	4	125.0839	2.0077e+04	-17.0548
6	5	125.0839	2.0077e+04	-15.3483
7	6	125.0839	2.0077e+04	-1.6965
8	7	125.0839	2.0077e+04	0.0100
9	8	154.8847	1.9044e+04	-17.7205
10	9	154.8847	1.9044e+04	-15.9475
11	10	154.8847	1.9044e+04	-1.7631
12	11	154.8847	1.9044e+04	0.0100
13	12	272.5302	2.0583e+04	-16.9674
14	13	272.5302	2.0583e+04	-15.2696
15	14	272.5302	2.0583e+04	-1.6877
16	15	272.5302	2.0583e+04	0.0100

Figura 5.3: Primeras dieciséis filas de la matriz de puntos del dominio. La matriz de puntos tiene dimensiones 54176 filas y 4 columnas.

	1	2	3	4	5	6
1	253	17081	17077	254	17082	17078
2	254	17082	17078	255	17083	17079
3	255	17083	17079	256	17084	17080
4	17081	5349	17593	17082	5350	17594
5	17082	5350	17594	17083	5351	17595
6	17083	5351	17595	17084	5352	17596
7	17077	17593	5345	17078	17594	5346
8	17078	17594	5346	17079	17595	5347
9	17079	17595	5347	17080	17596	5348
10	17593	17077	17081	17594	17078	17082
11	17594	17078	17082	17595	17079	17083
12	17595	17079	17083	17596	17080	17084
13	5349	17085	17601	5350	17086	17602
14	5350	17086	17602	5351	17087	17603
15	5351	17087	17603	5352	17088	17604
16	17085	285	17597	17086	286	17598
17	17086	286	17598	17087	287	17599
18	17087	287	17599	17088	288	17600

Figura 5.4: Primeras dieciocho filas de la matriz de triprismas del dominio. La matriz de puntos tiene dimensiones 76416 filas y 6 columnas.

	X	Y	Z
mínimo[m]	0	0	-87.3912
máximo[m]	22590.9	25414.6	0.01

Tabla 5.1: Variación en cada coordenada de los puntos de la Bahía Quellón.

para las siguientes cuatro filas y así sucesivamente. Ver figura 5.4.

De la observación anterior, hay prismas en tres niveles de altura: es decir, para cualquier punto con numeración k de la fila i y columna j, el punto de la fila i $+ 1$ y columna j será $k + 1$, mientras que para el punto de la fila i $+ 2$ y columna j será $k + 2$, en la siguiente fila se parte de otro punto y se vuelve a repetir el patrón. La figura 5.5 muestra los tres niveles de prismas en altura.

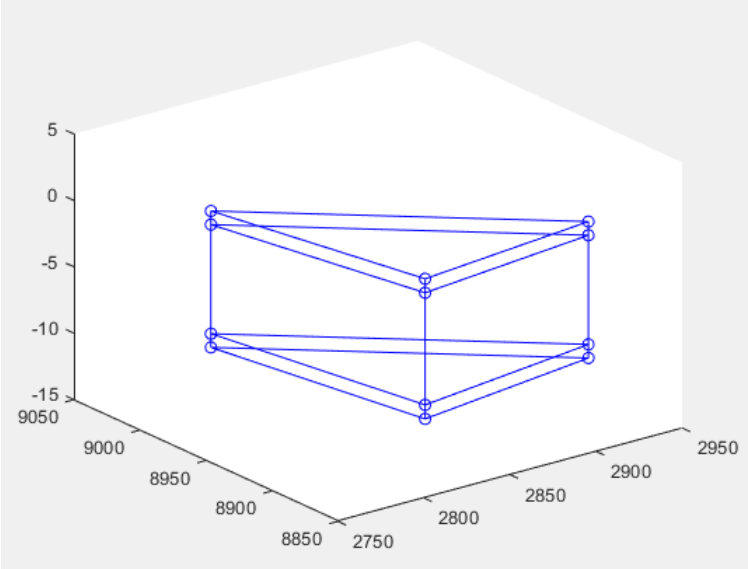


Figura 5.5: Ejemplo de los tres niveles de prismas, para las primeras 3 filas de la matriz de prismas.

2. Recorrer cada prisma, guardando las caras, el owner y el neighbour de cada cara:
   La astucia al recorrer los prismas fue aprovechar la estructura de *TriPrismas* que se explicó más arriba, para hacer un *for* sobre un tercio del número de prismas (que representan el número de prismas en cada nivel) y luego sobre los tres niveles, pues así en el primer nivel la primera cara sería de borde al pertenecer al fondo y la última cara del tercer nivel pertenecería a la superficie.
   Acá hay que tener en cuenta el orden en que entrego los puntos al momento de crear cada cara, éstos deben ser como si se estuviera recorriendo una circunferencia, puede ser en sentido horario o antihorario, pero hay que ser consistente con esta elección pues la normal exterior a la cara sigue la regla de la mano derecha.

   Observación: Al ir recorriendo prismas se decidió eliminar los prismas de dimensiones muy pequeñas, pues sino se tendría problemas con el Número de Courant y las simulaciones divergerían rapidamente, esto pues mientras menor es la variación en espacio, aumenta el número de Courant. Las simulaciones se mantienen estables mientas el número de Courant es menor a 1, cuando es mayor a 1 la velocidad crece rápidamente a infinito.

   La figura 5.6 muestra la matriz de caras en MATLAB.
3. Identificar caras internas y caras externas. Eliminar las caras repetidas:
   Es importante saber hacia donde apunta la normal exterior de cada cara, y las caras internas son caras que comparten dos prismas distintos. La información de la caras

	1	2	3	4
1	17077	17081	253	0
2	253	17081	17082	254
3	253	254	17078	17077
4	17081	17077	17078	17082
5	254	17082	17078	0
6	254	17082	17078	0
7	254	17082	17083	255
8	254	255	17079	17078
9	17082	17078	17079	17083
10	255	17083	17079	0
11	255	17083	17079	0
12	255	17083	17084	256
13	255	256	17080	17079
14	17083	17079	17080	17084
15	256	17084	17080	0
16	17593	5349	17081	0
17	17081	5349	5350	17082
18	17081	17082	17594	17593

Figura 5.6: Primeras dieciocho filas de la matriz de caras del dominio. La matriz de caras completa (sin considerar los volúmenes de dimensiones muy pequeñas) tiene 381705 filas y 4 columnas. Las caras de tres nodos, reciben un cero en la cuarta columna.

internas que se eliminan, es reemplazada por los vecinos.
Para distinguir las caras internas, una opción es para cada cara recorrer la matriz de caras y ver si en alguna fila todos los nodos coinciden con la cara actual (aunque estén en distinto orden), en dicho caso la cara es la misma. Esto se puede hacer menos costoso con el comando *find* para buscar la fila en que los nodos son los mismos de la cara actual, en dicho caso la cara actual es una cara interna. El problema de esto es que en las caras internas, los nodos no necesiamente están en el mismo orden para los dos prismas que conecta.
Por este motivo, se hace uso del comando *sort* y ordenar de forma creciente las numeraciones de los puntos en cada cara y luego pegar estas numeraciones en un sólo número, obteniendo un arreglo, en que cada número es un identificador de la cara. Aplicar nuevamente *sort* para ordenar de forma creciente los identificadores de la caras y obtener el arreglo de numeraciones original de cada cara en la matriz de caras (esto sólo se realiza con el propósito de hacer más eficiente el código, pero no es necesario). Luego el criterio para decidir si una cara es interna, es que en esta nueva matriz de caras ordenadas, es que dos filas consecutivas tengan los mismos nodos.
Al momento de guardar caras externas, estas se deben guardar después de las caras internas; la primera cara externa va después de la última cara interna. Además las filas consecutivas deben ser de la misma región del borde, es decir, si la fila i-ésima de la matriz de caras, es una cara externa del borde j, entonces la fila $\mathrm{i}+1$-ésima es una cara externa de la región j del borde, salvo cuando sea la última cara del borde, en cuyo caso la fila $\mathrm{i}+1$-ésima es una cara de la región $j+1$ del borde. Las caras de cada región de borde están en filas consecutivas, hasta pasar al siguiente borde.

4. Transformar la numeración de caras según los nodos efectivamente usados:
Ya que se eliminan los nodos asociados a caras muy pequeñas, no se utilizan todos los

nodos en la matriz de puntos, es por esto, que se eliminan los no utilizados y reenumeran. Esta reenumeración de los nodos efectivamente utilizados influye en la numeración de nodos que identifican a las caras, por lo cual hay que actualizar la numeración de nodos en cada cara.

5. Ordenar según *UpperTriangularOrder*:
   Para cada volumen finito, las caras deben estar ordenadas según el orden creciente de la numeración de los volúmenes finitos vecinos. Por ejemplo, antes de esta etapa para el prisma número 3 que tiene tres caras, en la matriz de owners los índices de estas caras son $6, 7, 8$ y las caras internas junto a los prismas vecinos se detallan en la tabla 5.2. Luego al reordenar, las filas de la matriz de owners se mantiene pero se permutan las filas de la matriz de neighbours, lo cual se puede observar de la tabla 5.3 en que se permuta las filas 7 con la 8.

Cara	Owner	Neighbour
6	3	4
7	3	42
8	3	9

Tabla 5.2: Caras internas y prismas vecinos del prisma con numeración 3, antes de aplicar el reordenamiento

Cara	Owner	Prismas vecinos
6	3	4
7	3	9
8	3	42

Tabla 5.3: Caras internas y prismas vecinos del prisma con numeración 3, luego de aplicar el reordenamiento.

6. Imprimir los puntos, caras recibiendo la etiqueta de cada punto, los owners, neighbours y el borde:
   Este se debe hacer indicando la cara en que comienza y el número de caras de cada región del borde, en formato OpenFOAM, quedando como muestra la figura 5.7.

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       vectorField;
    location    "constant/polyMesh";
    object      points;
}
54116
(
(0 19543.655192 -348.94984)
(0 19543.655192 -314.05486)
(0 19543.655192 -34.89498)
(0 19543.655192 0)
(125.083856 20076.577853 -341.29654)
(125.083856 20076.577853 -307.16688)
(125.083856 20076.577853 -34.12966)
(125.083856 20076.577853 0)
(154.884741 19043.929135 -354.61074)
(154.884741 19043.929135 -319.14966)
(154.884741 19043.929135 -35.46108)
(154.884741 19043.929135 0)
(272.530191 20582.705909 -339.54704)
(272.530191 20582.705909 -305.59234)
(272.530191 20582.705909 -33.9547)
(272.530191 20582.705909 0)
(372.932217 18582.130677 -358.7239)
(372.932217 18582.130677 -322.85152)
(372.932217 18582.130677 -35.8724)
(372.932217 18582.130677 0)
(380.400597 21108.936732 -338.49648)
(380.400597 21108.936732 -304.64684)
(380.400597 21108.936732 -33.84964)
(380.400597 21108.936732 0)
```

(a) Lista de puntos.

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       faceList;
    location    "constant/polyMesh";
    object      faces;
}
218728
(
3(241 17041 17037)
4(17040 17036 17037 17041)
3(242 17042 17038)
4(17041 17037 17038 17042)
4(17042 17038 17039 17043)
3(17041 5329 17553)
4(17040 17041 17553 17552)
4(5328 17552 17553 5329)
3(17042 5330 17554)
4(17041 17042 17554 17553)
4(5329 17553 17554 5330)
4(17042 17043 17555 17554)
4(5330 17554 17555 5331)
3(17037 17553 5325)
4(17036 17552 17553 17037)
4(17552 5324 5325 17553)
3(17038 17554 5326)
4(17037 17553 17554 17038)
4(17553 5325 5326 17554)
4(17038 17554 17555 17039)
4(17554 5326 5327 17555)
3(17553 17037 17041)
3(17554 17038 17042)
3(5329 17045 17561)
4(17044 17560 17561 17045)
```

(b) Lista de caras.

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       polyBoundaryMesh;
    location    "constant/polyMesh";
    object      boundary;
}
4
(
    Borde01
    {
        type            wall;
        nFaces          25447;
        startFace       162977;
    }
    Borde02
    {
        type            wall;
        nFaces          25447;
        startFace       188424;
    }
    Borde03
    {
        type            wall;
        nFaces          4305;
        startFace       213871;
    }
    Borde04
    {
        type            wall;
        nFaces          552;
        startFace       218176;
    }
)
```

(c) Diccionario del borde.

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       labelList;
    note        "nPoints: 54116 nCells: 76341 nFaces: 218728 nInternalFaces: 162977";
    location    "constant/polyMesh";
    object      owner;
}

218728
(
0
0
1
1
2
3
3
3
4
4
4
5
5
6
6
6
7
7
7
8
8
9
10
```

(d) Lista de owners.

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       labelList;
    note        "nPoints: 54116 nCells: 76341 nFaces: 218728 nInternalFaces: 162977";
    location    "constant/polyMesh";
    object      neighbour;
}

162977(
1
9
2
10
11
4
9
42
5
10
43
11
44
7
9
39
8
10
40
11
41
10
11
13
```

(e) Lista de neighbours.

Figura 5.7: Generación de la malla en formato OpenFOAM.

El dominio discretizado visto en ParaView se muestra en la figura 5.8.

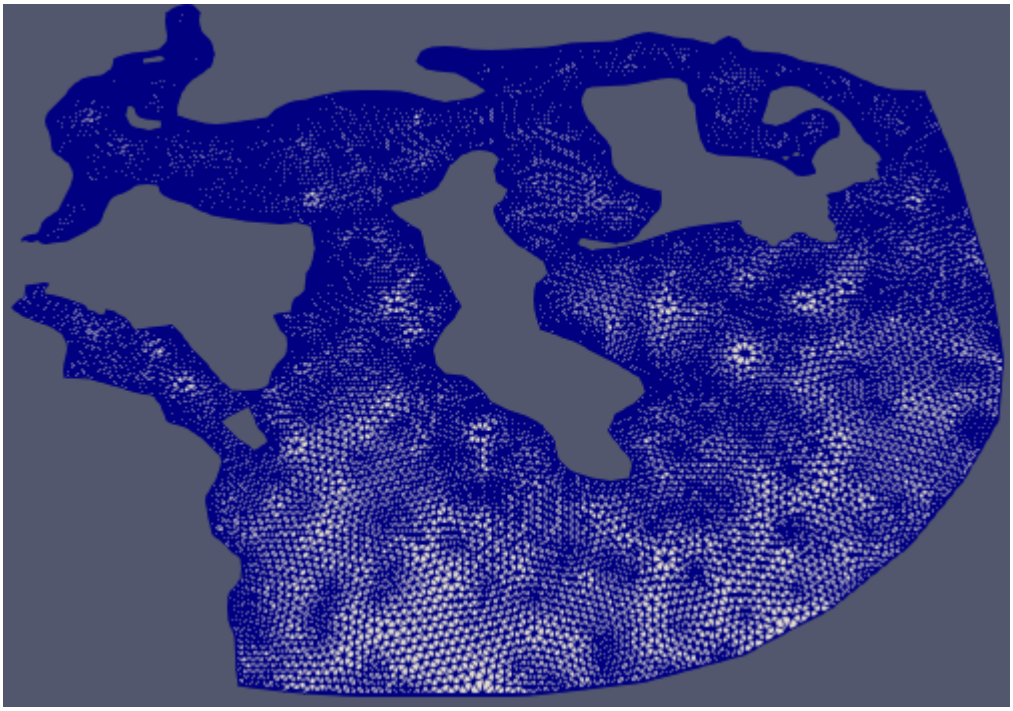
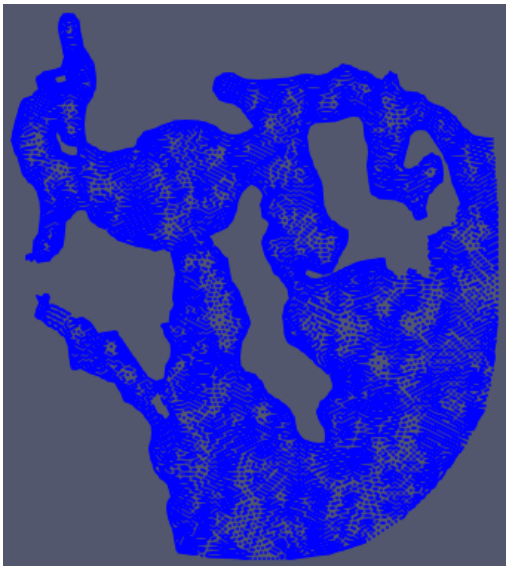
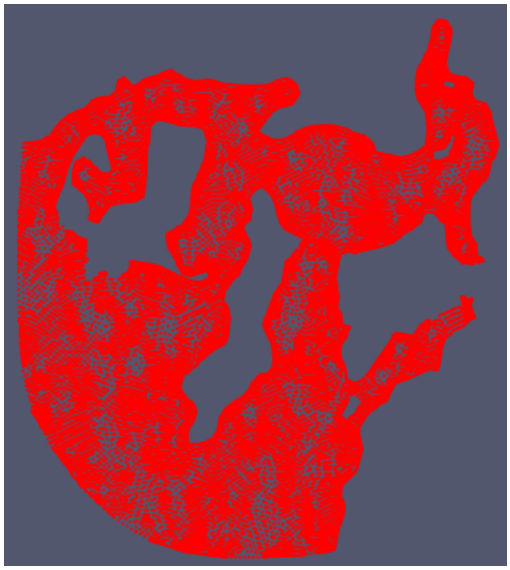


Figura 5.8: Dominio con sus volúmenes finitos.

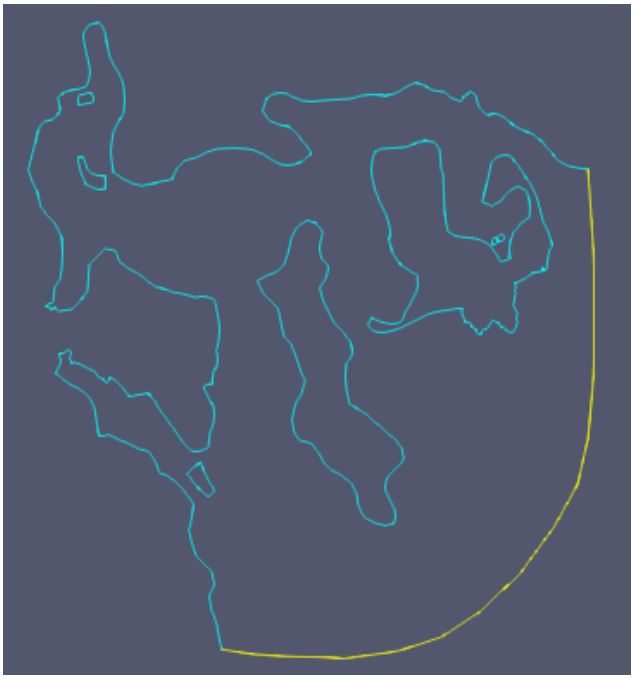
El borde separado en sus cuatro regiones se muestra en la figura 5.10.



(a) Superficie del mar.



(b) Fondo del mar.

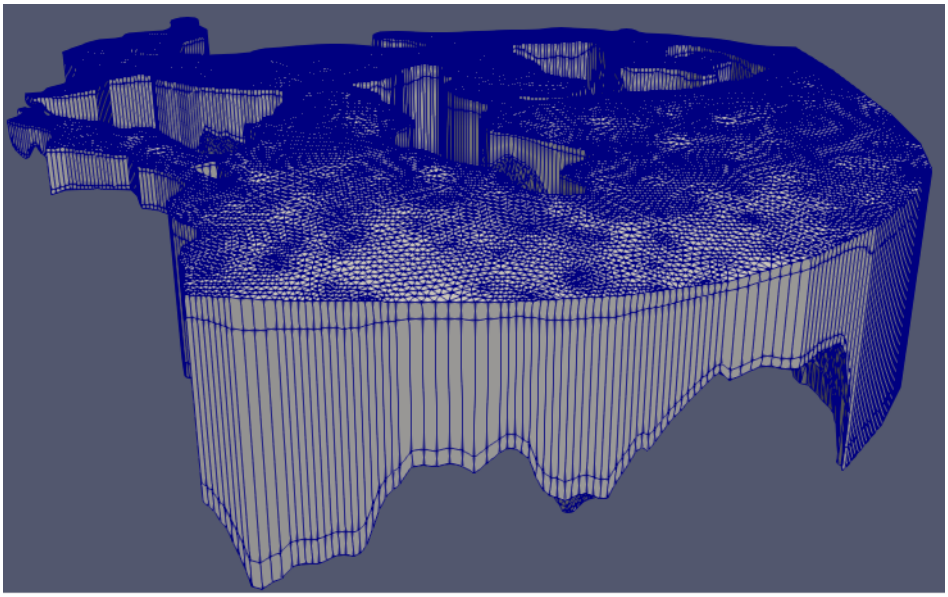


(c) Interacción mar-tierra en celeste e interacción mar-mar en amarillo.

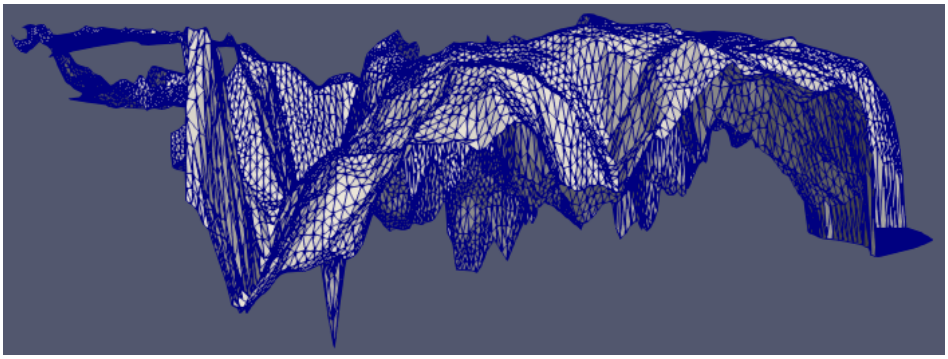
Figura 5.9: Dominio visto con Paraview.

Notar que el dominio se ve plano, esto es pues la escala vertical es mucho menor a la escala horizontal. A continuación se muestra un dominio escalado en el eje z para poder apreciar con mayor detalle la altura del mar. Esto se muestra en las figuras 5.10a, 5.10b, 5.10c.

(a) Dominio de la bahía quellón, escalado en un factor 100 las coordenadas z.



(b) Malla de la bahía quellón, escalado en 2 órdenes de magnitud menos en el eje x e y.



(c) Malla de la bahía quellón, escalado en 2 órdenes de magnitud menos en el eje x e y.

Figura 5.10: Dominio escalado y malla de volúmenes finitos.

5.3. Solvers

OpenFOAM ofrece una amplia gama de solvers de EDP's, para la ecuación del calor, de Navier-Stokes, ecuación de Black-Scholes, ecuación del transporte; también hay solvers para transferencia de calor, química, combustión, mezcla de fluidos, etc. A continuación se describen brevemente los solvers utilizados en este trabajo.

- **icoFoam:** Es el solver para un fluido Newtoniano e incompresible cuyo régimen es laminar. Es decir, se resuelven las ecuaciones clásicas de Navier-Stokes. Para resolver las ecuaciones de Navier-Stokes se usa el Algoritmo PISO.
- **pisoFoam:** Es el solver para un fluido Newtoniano e incompresible cuyo régimen es turbulento. Se pueden elegir modelos turbulentos del tipo *RAS* (Reynolds Avegared Simulation). Para resolver las ecuaciones utiliza el Algoritmo PISO.
- **pimpleFoam:** Solver para fluidos incompresibles, con dependencia temporal y turbulencias. Usa el Algoritmo PIMPLE en lugar del Algoritmo PISO para resolver las ecuaciones.
- **shallowWaterFoam:** Solver para resolver las ecuaciones de aguas poco profundas (*Shallow Water Equations*) con rotación, viscocidad nula.
- **scalarTransportFoam:** Solver para la ecuación del transporte (convección-difusión), con velocidad constante.

Para comprender mejor como funcionan estos Solvers, es necesario revisar los algoritmos de resolución que utilizan. Para resolver Navier-Stokes se utilizan Algoritmos de tipo *Splitting*, esto quiere decir que se resuelve separadamente la velocidad de la presión.
La idea de los métodos es la siguiente: En las ecuaciones de Navier-Stokes, para cada volumen finito hay 4 ecuaciones y 4 incógnitas (u_x, u_y, u_z, P), con $P := \frac{p}{\rho}$ donde la densidad es constante. No obstante, no hay una ecuación para la presión; la ecuación de continuidad de flujo (3.3) es una restricción para la velocidad y la presión se puede pensar como el multiplicador asociado a dicha restricción. Para lidiar con esta complicación, se calcula la velocidad en un instante de tiempo, usando la presión en el instante de tiempo anterior; luego se calcula la presión usando la velocidad calculada anteriormente, y finalmente es necesario corregir la velocidad para que cumpla la condición de divergencia nula.

Las ecuaciones de Navier-Stokes (3.2) se escriben en OpenFOAM como

```
fvVectorMatrix UEqn
(
    fvm::ddt(U)
  + fvm::div(phi, U)
  - fvm::laplacian(nu, U)
);
```

donde `phi` es el flujo $u \cdot n$, que aparece al aplicar el Teorema de la Divergencia en el término convectivo.

- **Etapa de Predicción de Momentum:** La ecuación de momentum se escribe matricialmente como

$$\mathcal{M}u_*^{k+1} = -\nabla P^k \tag{5.1}$$

donde se usa la presión obtenida en el tiempo anterior P^k, para calcular una velocidad tentativa en el tiempo actual u_*^{k+1}. Esta velocidad es tentativa, es decir no es la solución, pues no cumple la restricción de la ecuación de continuidad de flujo, por lo cual es necesario corregirla. En OpenFOAM esto se ve en la siguiente línea de código:

```
solve(UEqn == - fvc::grad(p));
```

- **Cálculo de la presión:** Primero se define el residuo al extraer la diagonal de la matriz de momentum

$$\mathcal{H} = \mathcal{A}u - \mathcal{M}u \tag{5.2}$$

OBSERVACIÓN: Se define como (5.2) y **NO** como $\mathcal{H}u = \mathcal{A}u - \mathcal{M}u$.
La ventaja de esta definición, es que las matrices diagonales son fáciles de invertir, y su inversa viene dada por

$$\mathcal{A}^{-1} = \begin{pmatrix} \frac{1}{m_{11}} & 0 & \dots & 0 & 0 \\ 0 & \frac{1}{m_{22}} & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & 0 & \frac{1}{m_{(N-1)(N-1)}} & 0 \\ 0 & 0 & \dots & 0 & \frac{1}{m_{NN}} \end{pmatrix}$$

Usando (5.1) y (5.2), se tiene

$$\mathcal{A}u - \mathcal{H} = -\nabla P$$

Multiplicando por $\mathcal{A}^{-1}$ y reordenando,

$$u^{k+1} = \mathcal{A}^{-1}\mathcal{H} - \mathcal{A}^{-1}\nabla P \tag{5.3}$$

Aplicando divergencia y usando que $\mathrm{div}(u) = 0$, se llega a una ecuación de Poisson para la presión

$$\mathrm{div}(\mathcal{A}^{-1}\nabla P^{k+1}) = \mathrm{div}(\mathcal{A}^{-1}\mathcal{H}) \tag{5.4}$$

En OpenFOAM,

```
volScalarField rAU(1.0/UEqn.A());
volScalarField HbyA(constrainHbyA(rAU*UEqn.H(), U, p));
fvScalarMatrix pEqn
(
    fvm::laplacian(rAU, p) == fvc::div(phiHbyA)
);
pEqn.solve();
```

- **Corrección de la velocidad:** Una vez hallada la presión, se puede corregir la velocidad usando (5.3) con la presión actual

$$u^{k+1} = \mathcal{A}^{-1}\mathcal{H} - \mathcal{A}^{-1}\nabla P^{k+1}$$

- **Actualización de la presión:** Ya están calculadas P^{k+1} y u^{k+1}, sin embargo, una vez se obtiene la velocidad corregida u^{k+1}, la presión P^{k+1} ya no satisface la ecuación (5.4) pues el término fuente (lado derecho) $\mathcal{H}$ depende de la velocidad, la cual acaba de ser actualizada. Aquí es donde aparecen los Algoritmos **SIMPLE** y **PISO**, que se diferencian en la forma de corregir la presión.

1. **Algoritmo SIMPLE**: El Algoritmo SIMPLE es un acrónimo *Semi-Implicit Method for Pressure-Linked Equations*. Para actualizar la presión, se vuelve a resolver la ecuación de predicción de Momemtum y repite todo el proceso anterior, hasta que la solución converja.

$$\begin{aligned} \mathcal{M}u_*^{k+1} &= \nabla P^{k+1} \\ \mathcal{H} &= \mathcal{A}u_*^{k+1} - \mathcal{M}u_*^{k+1} \\ \mathrm{div}(\mathcal{A}^{-1}\nabla P^{k+1}) &= \mathrm{div}(\mathcal{A}^{-1}\mathcal{H}) \\ u^{k+1} &= \mathcal{A}^{-1}\mathcal{H} - \mathcal{A}^{-1}\nabla P^{k+1} \end{aligned}$$

   Esto se conoce como *outer corrector loops*.
   La razón del nombre "Semi-Implícito", es pues las ecuaciones discretizadas de momentum y correción de presión se resuelven implícitamente, pero la correción de velocidad es explícita.
   El algoritmo se puede resumir como sigue:

Algoritmo 3 Algoritmo SIMPLE.

1: Establece las condiciones de borde.
2: Predicción de Momentum: Usando la presión calculada en el tiempo anterior P^k (se considera término fuente en esta etapa), se calcula la velocidad tentativa

$$\mathcal{M}u_*^{k+1} = -\nabla P^k$$

3: Cálculo del Residuo al extraer la matriz diagonal: $\mathcal{H} = \mathcal{A}u_*^{k+1} - \mathcal{M}u_*^{k+1}$.
4: Obtener la presión actual: $\mathrm{div}(\mathcal{A}^{-1}\nabla P^{k+1}) = \mathrm{div}(\mathcal{A}^{-1}\mathcal{H})$.
5: Corregir la velocidad actual: $u^{k+1} = \mathcal{A}^{-1}\mathcal{H} - \mathcal{A}^{-1}\nabla P^{k+1}$.
6: Volver a la etapa 2 (*outer corrector loops*) para recorregir presión, y por ende la velocidad, hasta conseguir convergencia. Si se alcanza la convergencia, entonces u^{k+1} y P^{k+1} están bien calculados, en dicho caso actualizar el tiempo $k \leftarrow k+1$ y pasar a la etapa 1.

2. **Algoritmo PISO**: El Algoritmo PISO cuyo nombre es una sigla de *Pressure Implicit With Split Operator* es para problemas no estacionarios (*transient*).
   Para actualizar la presión se actualiza directamente el residuo $\mathcal{H}$ con la velocidad obtenida en la etapa de Correción de Velocidad. Se realizan lo que se conoce como *inner loops* (pues resuelve la ecuación de momentum sólo una vez)

$$\begin{aligned} \mathcal{H} &= \mathcal{A}u^{k+1} - \mathcal{M}u^{k+1} \\ \mathrm{div}(\mathcal{A}^{-1}\nabla P^{k+1}) &= \mathrm{div}(\mathcal{A}^{-1}\mathcal{H}) \\ u^{k+1} &= \mathcal{A}^{-1}\mathcal{H} - \mathcal{A}^{-1}\nabla P^{k+1} \end{aligned}$$

   A continuación se resume este algoritmo.

Algoritmo 4 Algoritmo PISO.

1: Establece las condiciones de borde.
2: Predicción de Momentum: Usando la presión calculada en el tiempo anterior P^k (se considera término fuente en esta etapa), se calcula la velocidad tentativa

$$\mathcal{M}u_*^{k+1} = -\nabla P^k$$

3: Cálculo del Residuo al extraer la matriz diagonal: $\mathcal{H} = \mathcal{A}u - \mathcal{M}u$.
4: Obtener la presión actual: $\mathrm{div}(\mathcal{A}^{-1}\nabla P^{k+1}) = \mathrm{div}(\mathcal{A}^{-1}\mathcal{H})$.
5: Corregir la velocidad actual: $u^{k+1} = \mathcal{A}^{-1}\mathcal{H} - \mathcal{A}^{-1}\nabla P^{k+1}$.
6: Volver a la etapa 3 (*inner loops*) para calcular $\mathcal{H}$ y repetir el proceso hasta conseguir convergencia. Si se alcanza la convergencia, entonces u^{k+1} y P^{k+1} están bien calculados, en dicho caso actualizar el tiempo $k \leftarrow k+1$ y comenzar nuevamente en la etapa 1.

OBSERVACIÓN: El criterio de convergencia en cada etapa temporal es

$$\|\mathcal{M}u^{k+1} + \nabla P^{k+1}\| < \varepsilon_1, \quad \left|\mathrm{div}(u^{k+1})\right| < \varepsilon_2$$

3. **Algoritmo PIMPLE**: Es una combinación del Algoritmo PISO y del Algoritmo SIMPLE, pues utiliza tanto correctores externos (*outer correctors loops*) como correctores internos (*inner correctors loops*).
   Hay que saber distinguir los correctores externos de los correctores internos: el número de correctores externos define cuantas veces se resuelven la ecuación de momentum antes de pasar al siguiente tiempo y el número de correctores internos es el número de veces que la presión es corregida en cada iteración. Se sugiere usar al menos 50 correcciones externas y $1-3$ internas, pues las correciones internas suelen mejorar poco los resultados. Usar sólo un corrector externo es equivalente al Algoritmo PISO.
   PIMPLE es un algoritmo muy estable, útil cuando se trabaja con grandes pasos temporales y se quiera mantener la condición CFL de $Co = U\dfrac{\Delta t}{\Delta x} \leq 1$ en una dimensión y más generalmente el número de Courant en el volumen ω_i se calcula como

$$Co_\mathrm{i} = \frac{1}{2\left|\omega_\mathrm{i}\right|} \sum_{j:\Gamma_j \text{es cara de } \omega_\mathrm{i}} |u \cdot n(\Gamma_j)|\,|\Gamma_j|$$

   El Algoritmo PIMPLE en OpenFOAM permite ir variando el paso temporal, ajustando el número de Courant, fijando un número de Courant máximo y así en cada iteración el paso temporal se adapta según este.

5.3.1. Condiciones de Borde

En las condiciones de Navier-Stokes con OpenFOAM aparece algo que no ocurre con al implementación en MATLAB del método de resolver simultaneamente la velocidad y presión; se debe poner condiciones de borde tanto para velocidad como presión, debido a que el método es de splitting, es decir cuando se resuelva la ecuación de momentum se requiere de condición de borde para la velocidad y cuando se resuelva la ecuación de Poisson de la presión se

requiere de su propia condición de borde. Para que no queden sobredeterminada la solución, lo recomendable es usar condición de borde Neumann *zeroGradient* (derivada normal nula) para la variable que no lleva condición de borde tipo Dirichlet. De esta manera, las condiciones de borde se resumen a continuación en la tabla 5.4.

Regiones del borde	Condición sobre u	Condición sobre P
Fondo	$u \cdot n = 0$	$\frac{\partial p}{\partial n} = 0$
Superficie	$u_z = W = -A\omega sen(\omega t)$	$\frac{\partial p}{\partial n} = 0$
Mar-Tierra	$u = 0$	$\frac{\partial p}{\partial n} = 0$
Mar-Mar	$\frac{\partial u}{\partial n} = 0$	$P = 0$

Tabla 5.4: Tabla resumen condiciones de borde Navier-Stokes en OpenFOAM.

5.4. Shallow Water Equations

La ecuaciones Shallow Water (aguas poco profundas) son un conjunto de EDP que describen el movimiento de un fluido incompresible en un dominio cuya profundidad es considerada poco profunda respecto al largo horizontal. Estas ecuaciones se derivan al integrar en profundad las ecuaciones de Navier-Stokes, en el caso en que el largo horizontal es mucho mayor al largo vertical.
Bajo estas hipótesis, de las ecuaciones de Navier-Stokes se obtiene que el gradiente de presión vertical es aproximadamente hidrostático. Al integrar verticalmente la velocidad vertical es removida de las ecuaciones, es decir, la velocidad vertical U_z no aparece en las ecuaciones Shallow Water, lo cual no significa que sea nula. Una vez que se resuelven dichas ecuaciones, la velocidad vertical se puede recuperar de las ecuaciones de continuidad de masa.

$$\begin{aligned} \frac{\partial}{\partial t}(hu) + \nabla \cdot (hu^T u) + f \times hu &= -|g| \nabla (h + h_0) \\ \frac{\partial h}{\partial t} + \nabla \cdot (hu) &= 0 \end{aligned}$$

con $h(t, x, y)$ la distancia desde el fondo a la superficie y $h_0(x, y)$ la distancia desde el sistema de referencia (que en OpenFOAM está definido debajo del fondo del mar) al fondo del mar, $u(x, y) = (U_x, U_y)$ el vector velocidad promediado en profundidad del mar, $f := (2\Omega \cdot \hat{z})$ la fuerza de Coriolis, con Ω la rotación angular de la Tierra.

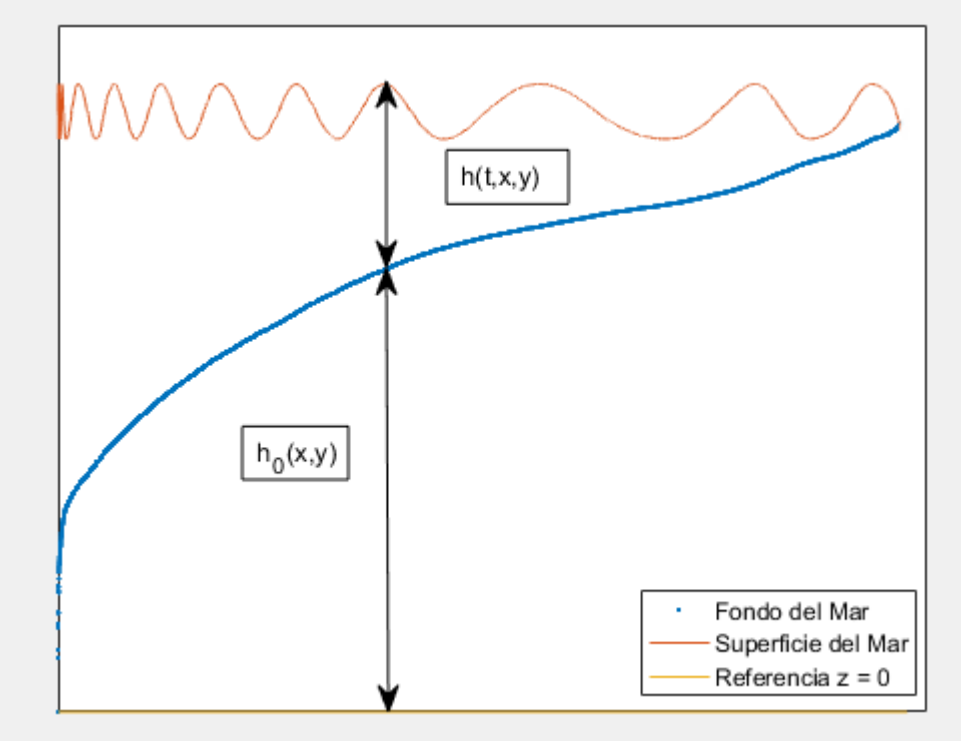


Figura 5.11: Esquema de la función h distancia del fondo a la superficie y h_0 distancia del sistema de referencia al fondo.

5.4.1. Deducción de las Shallow Water Equations

Hay que deducir una ecuación a partir de la ecuación de continuidad de masa (o fluido incompresible) y otra de la de conservación del momentum.
La condición de fluido incompresible es

$$\nabla \cdot u = 0 \text{ en } \Omega$$

Para un instante de tiempo t, al integrar en profundidad desde $z = z_f(x, y)$ el fondo hasta la superficie $z = z_s(t, x, y)$, se obtiene

$$\begin{aligned}
0 &= \int_{z_f(x,y)}^{z_s(t,x,y)} \nabla \cdot u \mathrm{d}z \\
&= \int_{z_f(x,y)}^{z_s(t,x,y)} \left(\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} \right) \mathrm{d}z + \int_{z_f(x,y)}^{z_s(t,x,y)} \frac{\partial u_3}{\partial z} \mathrm{d}z \\
&= \int_{z_f(x,y)}^{z_s(t,x,y)} \left(\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} \right) \mathrm{d}z + u_3|_{z=z_s(t,x,y)} - u_3|_{z=z_f(x,y)} \\
&= \frac{\partial}{\partial x} \int_{z_f(x,y)}^{z_s(t,x,y)} u_1 \mathrm{d}z - u_1|_{z=z_s(t,x,y)} \frac{\partial z_s}{\partial x} + u_1|_{z=z_f(x,y)} \frac{\partial z_f}{\partial x} \\
&\quad + \frac{\partial}{\partial y} \int_{z_f(x,y)}^{z_s(t,x,y)} u_2 \mathrm{d}z + u_2|_{z=z_s(t,x,y)} \frac{\partial z_s}{\partial x} + u_2|_{z=z_f(x,y)} \frac{\partial z_f}{\partial y} \\
&\quad + u_3|_{z=z_s(t,x,y)} - u_3|_{z=z_f(x,y)}
\end{aligned}$$

Con el propósito de eliminar los términos de borde, se imponen las siguientes condiciones de borde sobre el fondo y superficie del mar:

- Fondo ($z = z_f(x, y)$):

$$u_3|_{z=z_f(x,y)} = u_1|_{z=z_f(x,y)} \frac{\partial z_f}{\partial x} + u_2|_{z=z_f(x,y)} \frac{\partial z_f}{\partial y} \tag{5.5}$$

- Superficie ($z = z_s(t, x, y)$): Flujo relativo no normal

$$u_3|_{z=z_s(t,x,y)} = \frac{\partial z_s}{\partial t} + u_1|_{z=z_s(t,x,y)} \frac{\partial z_s}{\partial x} + u_2|_{z=z_s(t,x,y)} \frac{\partial z_s}{\partial y} \tag{5.6}$$

$$p(z_s(t, x, y)) = 0 \tag{5.7}$$

Así se obtiene,

$$\frac{\partial z_s}{\partial t} + \frac{\partial}{\partial x} \int_{z_f(x,y)}^{z_s(t,x,y)} u_1 \mathrm{d}z + \frac{\partial}{\partial y} \int_{z_f(x,y)}^{z_s(t,x,y)} u_2 \mathrm{d}z = 0$$

Recordando que $h(t, x, y) = z_s(t, x, y) - z_f(x, y)$, entonces

$$\frac{\partial h}{\partial t} + \frac{\partial}{\partial x} h \frac{1}{h} \int_{z_f(x,y)}^{z_s(t,x,y)} u_1 \mathrm{d}z + \frac{\partial}{\partial y} h \frac{1}{h} \int_{z_f(x,y)}^{z_s(t,x,y)} u_2 \mathrm{d}z = 0$$

Por último, definiendo las velocidades promediadas en profundidad

$$\begin{aligned} \overline{u_1}(t, x, y) &= \frac{1}{h} \int_{z_f(x,y)}^{z_s(t,x,y)} u_1 \mathrm{d}z \\ \overline{u_2}(t, x, y) &= \frac{1}{h} \int_{z_f(x,y)}^{z_s(t,x,y)} u_2 \mathrm{d}z \end{aligned}$$

Se obtiene

$$\frac{\partial h}{\partial t} + \frac{\partial (h\overline{u_1})}{\partial x} + \frac{\partial (h\overline{u_2})}{\partial y} = \frac{\partial h}{\partial t} + \nabla \cdot h\overline{u} = 0 \tag{5.8}$$

Ahora queda estudiar la ecuación de conservación del momentum. Las fuerzas actuantes en este caso son la gravedad y la fuerza Coriolis, con $F_C := f \times u = (\|f\| u_2, -\|f\| u_1, 0)$ la fuerza Coriolis y $F_g = (0, 0, -|g|)$ la fuerza de gravedad, por lo tanto la fuerza total es

$$F = F_C + F_g = (F_1, F_2, F_3) = (-\|f\| u_2, \|f\| u_1, -|g|)$$

En la componente vertical, el eje z las ecuaciones de Navier-Stokes son

$$\frac{\mathrm{d}u_3}{\mathrm{d}t} = \nu \left(\frac{\partial^2 u_3}{\partial x^2} + \frac{\partial^2 u_3}{\partial y^2} + \frac{\partial^2 u_3}{\partial z^2} \right) - \frac{\partial p}{\partial z} - \rho |g| \tag{5.9}$$

Como se están modelando aguas oceánicas, la viscocidad es muy pequeña, así que se puede considerar $\nu = 0$. Además, por hipótesis de las Shallow Water Equations la aceleración vertical es nula, luego

$$0 = -\frac{\partial p}{\partial z} - \rho |g|$$

Integrando con respecto a z desde z hasta $z_s(t,x,y)$ y usando la condición de borde de la superficie de presión nula (5.7),

$$p(z) = -\rho\,|g|\,(z - z_s)$$

Derivando con respecto a x e y,

$$\frac{\partial p}{\partial x} = \rho\,|g|\,\frac{\partial z_s}{\partial x}, \quad \frac{\partial p}{\partial y} = \rho\,|g|\,\frac{\partial z_s}{\partial y} \tag{5.10}$$

La ecuación del momentum en el eje x,

$$\frac{\partial u_1}{\partial t} + u_1\frac{\partial u_1}{\partial x} + u_2\frac{\partial u_1}{\partial y} + u_3\frac{\partial u_1}{\partial z} = \nu\left(\frac{\partial^2 u_1}{\partial x^2} + \frac{\partial^2 u_1}{\partial y^2} + \frac{\partial^2 u_1}{\partial z^2}\right) - \frac{1}{\rho}\frac{\partial p}{\partial x} + \|f\|u_2$$

Considerando $\nu = 0$, y la identidad (5.10), la ecuación de momentum en el eje x se reduce a

$$\frac{\partial u_1}{\partial t} + u_1\frac{\partial u_1}{\partial x} + u_2\frac{\partial u_1}{\partial y} + u_3\frac{\partial u_1}{\partial z} - \|f\|u_2 = -\,|g|\,\frac{\partial z_s}{\partial x} \tag{5.11}$$

La idea es usar la regla de Leibniz integral, para sacar las derivadas de la integral. De la condición $\nabla \cdot u = 0$, se tiene que $u_1\left(\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} + \frac{\partial u_3}{\partial z}\right) = 0$.
Luego,

$$u_1\frac{\partial u_1}{\partial x} + u_2\frac{\partial u_1}{\partial y} + u_3\frac{\partial u_1}{\partial z} + fu_2 = \frac{\partial}{\partial x}(u_1)^2 + \frac{\partial}{\partial y}(u_1u_2) + \frac{\partial}{\partial z}(u_1u_3)$$

Reemplazando esto en (5.11) e integrando en profundidad desde $z = z_f(x,y)$ hasta $z = z_s(t,x,y)$,

$$\int_{z_f(x,y)}^{z_s(t,x,y)} \frac{\partial}{\partial x}(u_1)^2 + \frac{\partial}{\partial y}(u_1u_2) + \frac{\partial}{\partial z}(u_1u_3) - \|f\|u_2 \mathrm{d}z = -\,|g|\int_{z_f(x,y)}^{z_s(t,x,y)} \frac{\partial z_s}{\partial z}\mathrm{d}z \tag{5.12}$$

Desarrollando cada término por separado, usando la regla integral de Leibniz y el Teorema Fundamental del Cálculo

- $-\,|g|\int_{z_f}^{z_s}\frac{\partial z_s}{\partial z}\mathrm{d}z = -\,|g|\,\frac{\partial z_s}{\partial x}\int_{z_f}^{z_s}\mathrm{d}z = -\,|g|\,h\frac{\partial z_s}{\partial x}$
- $\int_{z_f}^{z_s}\frac{\partial u_1}{\partial t}\mathrm{d}z = \frac{\partial}{\partial t}\int_{z_f}^{z_s}u_1\mathrm{d}z - u_1|_{z=z_s}\frac{\partial z_s}{\partial t} = \frac{\partial}{\partial t}\overline{u_1} - u_1|_{z=z_s}\frac{\partial z_s}{\partial t}$
- $\int_{z_f}^{z_s}\frac{\partial u_1^2}{\partial x}\mathrm{d}z = \frac{\partial}{\partial x}\int_{z_f}^{z_s}u_1^2\mathrm{d}z - u_1^2|_{z=z_s}\frac{\partial z_s}{\partial x} + u_1^2|_{z=z_f}\frac{\partial z_f}{\partial x} = \frac{\partial}{\partial x}h\overline{u_1^2} - u_1^2|_{z=z_s}\frac{\partial z_s}{\partial x} + u_1^2|_{z=z_f}\frac{\partial z_f}{\partial x}$
- $$\begin{aligned}\int_{z_f}^{z_s}\frac{\partial(u_1u_2)}{\partial y}\mathrm{d}z &= \frac{\partial}{\partial y}\int_{z_f}^{z_s}u_1u_2\mathrm{d}z - (u_1u_2)|_{z=z_s}\frac{\partial z_s}{\partial y} + (u_1u_2)|_{z=z_f}\frac{\partial z_f}{\partial y}\\ &= \frac{\partial}{\partial y}h\overline{u_1u_2} - (u_1u_2)|_{z=z_s}\frac{\partial z_s}{\partial y} + (u_1u_2)|_{z=z_f}\frac{\partial z_f}{\partial y}\end{aligned}$$
- $\int_{z_f}^{z_s}\frac{\partial(u_1u_3)}{\partial z}\mathrm{d}z = (u_1u_3)|_{z=z_s} - (u_1u_3)|_{z=z_f}$

- $\int_{z_f}^{z_s} -\|f\|u_2 \mathrm{d}z = -\|f\|u_2 \int_{z_f}^{z_s} \mathrm{d}z = -\|f\|hu_2$

Sumando estas integrales y reemplazando en (5.12), se obtiene

$$\frac{\partial h\overline{u_1}}{\partial t} + \frac{\partial(h\overline{u_1^2})}{\partial x} + \frac{\partial(h\overline{u_1u_2})}{\partial y} - \|f\|hu_2 - u_1|_{z=z_s}\left(\frac{\partial z_s}{\partial t} + u_1|_{z=z_s}\frac{\partial z_s}{\partial x} + u_2|_{z=z_s}\frac{\partial z_s}{\partial y} - u_3|z=z_s\right)$$
$$+u_1|_{z=z_f}\left(u_1|_{z=f}\frac{\partial z_f}{\partial x} + u_2|_{z=z_f}\frac{\partial z_f}{\partial y} - u_3|_{z=z_f}\right) = -|g|\,h\frac{\partial z_s}{\partial x}$$

Usando las condiciones de borde del fondo y la superficie, (5.5) y (5.6) respectivamente

$$\frac{\partial h\overline{u_1}}{\partial t} + \frac{\partial(h\overline{u_1^2})}{\partial x} + \frac{\partial(h\overline{u_1u_2})}{\partial y} - \|f\|hu_2 = -|g|\,h\frac{\partial z_s}{\partial x}$$

Definiendo como sistema de referencia $z = 0$ bajo el fondo marino y $h_0(x, y)$ la elevación del fondo respecto al sistema de referencia, se tiene que la superficie está dada por

$$z_s(t, x, y) = h(t, x, y) + h_0(x, y)$$

Se obtiene que

$$\frac{\partial h\overline{u_1}}{\partial t} + \frac{\partial(h\overline{u_1^2})}{\partial x} + \frac{\partial(h\overline{u_1u_2})}{\partial y} - \|f\|hu_2 = -|g|\,h\frac{\partial(h+h_0)}{\partial x} \tag{5.13}$$

Análogamente, de la ecuación de conservación del momentum en el eje y, se deduce la expresión

$$\frac{\partial h\overline{u_2}}{\partial t} + \frac{\partial(h\overline{u_1u_2})}{\partial x} + \frac{\partial(h\overline{u_2^2})}{\partial y} + \|f\|hu_1 = -|g|\,h\frac{\partial(h+h_0)}{\partial y} \tag{5.14}$$

5.4.2. Creación de Solver en OpenFOAM

En OpenFOAM, el solver *shallowWaterFoam* resuelve las ecuaciones con incógnitas h,q, con $q = hu$ un cambio de variable, y al final de cada iteración temporal obtiene $u = \frac{q}{h}$ junto con $h_{Total} = h + h_0$. Es importante recalcar que u es la velocidad horizontal y por tanto tiene sólo dos componentes, en x e y.
Para el caso de la Bahía Quellón, en que interesa describir la marea, es más fácil estudiar h_{Total} que h, pues si h_{Total} describe la marea, depende sólo del tiempo y no del espacio, mientras que h depende tanto del tiempo como espacio.
Por ello se crea un nuevo solver que resuelva h_{Total} y q, para luego recuperar h. Esto tiene ventajas principalmente a la hora de poner las condiciones de borde cómo se verá más adelante. Además, aprovechando que se resuelven las EDP con el algoritmo PIMPLE, se incluye otra modificación en el solver, que fija un número máximo de Courant (que es la condición CFL $Co \leq 1$ para asegurar convergencia) en cada iteración temporal, y así ajustar el paso temporal Δt en cada iteración.
Esto se hizo en el solver *hTotalshallowWaterFoamConNroMaximoCourant*.
Recordar que este solver está escrito en $C++$, en un archivo *.C* por lo cual es necesario compilarlo haciendo *wmake* en Ubuntu, para que OpenFOAM pueda reconocer la creación de este nuevo solver.

5.4.3. Condiciones de borde

Como en Shallow Water el problema pasa a ser 2-D, las condiciones de borde deben ser en el dominio 2-D. En consecuencia, la superficie y fondo no son considerados como borde.

- Fondo: *empty*.
- Superficie: *empty*.
- Mar-Tierra: Velocidad normal nula, $u \cdot n = 0$. Esta condición representa que no hay flujo en la región Mar-Tierra.
- Mar-Mar: Esta es una condición de borde abierta, artificial. No es tan claro que condición de borde utilizar, sin embargo, la más intuitiva es imponer $h_{Total}(x,t) = h_{Total}(x,0) + Asen(\omega t)$ con $x \in \partial\Omega$ en Mar-Mar, A la amplitud de la marea y $\omega = \frac{2\pi}{T}$, en que $T = 12hrs = 43200s$ el período de mareas.

	Condición sobre hu	Condición sobre h_{Total}
Mar-Tierra	$u \cdot n = 0$	$\frac{\partial h_{Total}}{\partial n} = 0$
Mar-Mar	$\frac{\partial hu}{\partial n} = 0$	$h_{Total}(x,t) = h_{Total}(x,0) + Asen(\omega t)$ o $h_{Total}(x,t) = h_{Total}(x,0) + Acos(\omega t) - A$

Tabla 5.5: Tabla resumen condiciones de borde en Shallow Water.

Las figuras 5.13 muestran la condición de borde en Mar-Mar. Comparar con la figura 3.5a.

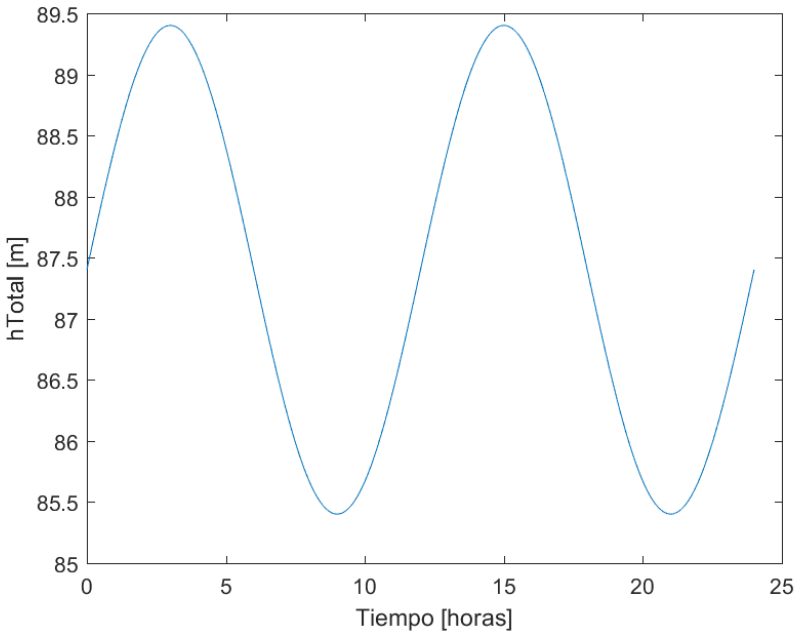


Figura 5.12: Condición de borde en Mar-Mar: nivel del mar sinusoidal $h_{Total}(x,t) = h_{Total}(x,0) + Asen(\omega t)$, con $A = 2m$, $T = 12hrs$ el período de marea.

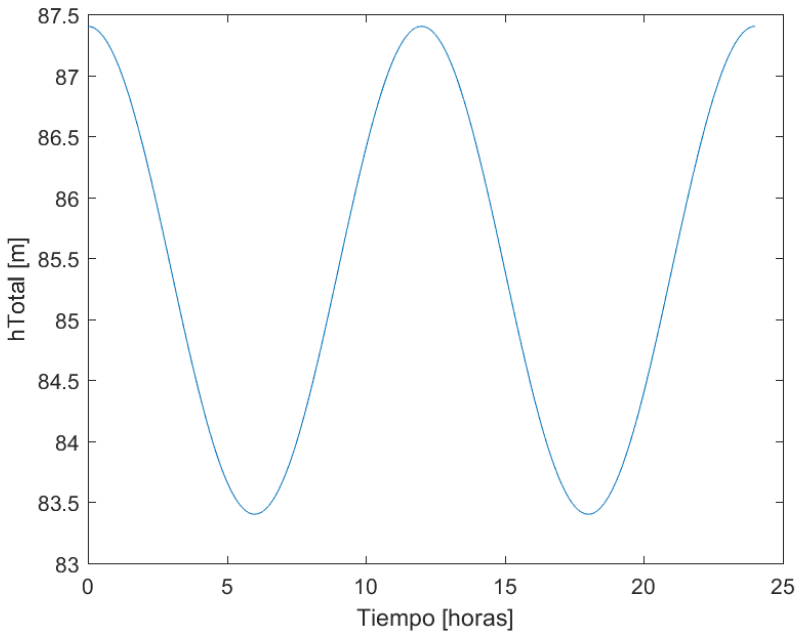


Figura 5.13: Condición de borde en Mar-Mar: nivel del mar
$h_{Total}(x,t) = h_{Total}(x,0) + Acos(\omega t)$, con $A = 2m$, $T = 12hrs$ el período de marea.

Las condiciones iniciales de h y h_0, se muestran mediante el gráfico en MATLAB

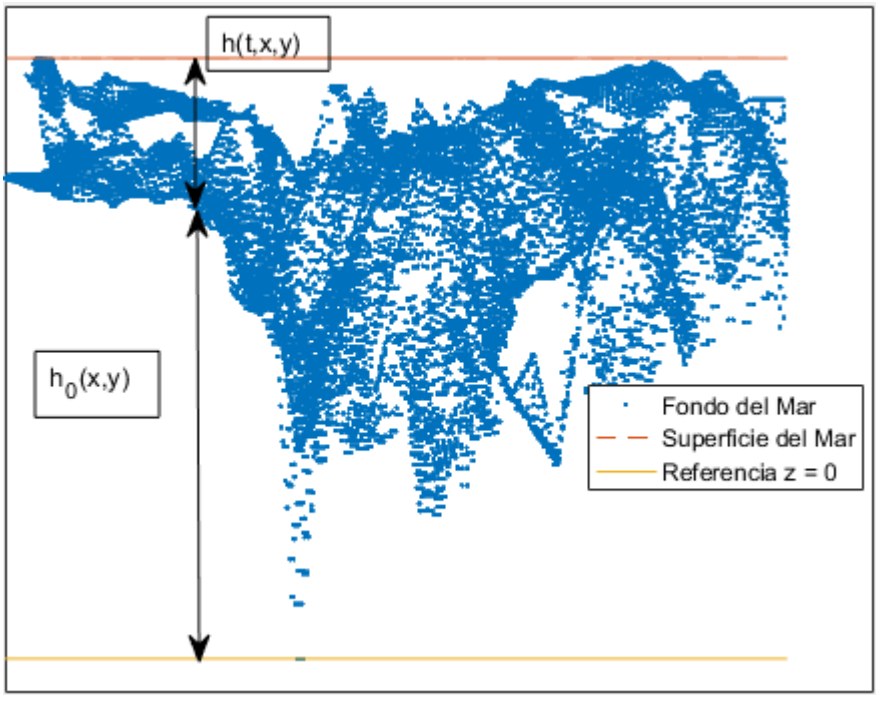


Figura 5.14: Batimetría del dominio h_0 y la altura de la columna de agua h.

5.5. scalarTransportFoam

En OpenFOAM el solver *scalarTransportFoam* resuelve la ecuación de convección-difusión. Sin embargo, este solver tiene un problema y es que la velocidad es constante respecto al tiempo. Para superar este inconviente se crea el solver *MareaRojaPimpleFoam* que resuelve Navier-Stokes y luego la ecuación de convección-difusión en cada iteración de tiempo. Para ello, en cada iteración de tiempo se calcula la velocidad dentro del loop del algoritmo *PISO* o *PIMPLE* y una vez se obtiene la velocidad actual en dicho tiempo (con las respectivas recorrecciones), se calcula la concentración C.
Primero es necesario descargar los códigos fuentes del solver tanto *icoFoam* o *pimpleFoam*, para modificar cada solver a gusto. En este caso, se explica a continuación la modificacion del solver *pimpleFoam* para acoplarle la convección-difusión.

1. Modificar el archivo *createFields.H*: Debajo de la línea

```
#include "createRDeltaT.H"
```

hay que definir la constante de difusión , inspirándose en cómo se define la viscosidad en OpenFOAM o copiando las mismas líneas del solver *scalarTransportFoam*. En dicho solver se usa la notación DT para la constante de difusión. Se puede usar cualquier nombre, pero luego hay que ser consistente a la hora de trabajar con el *case* cuando el solver ya esté compilado.

```
IOdictionary transportProperties
(
    IOobject
    (
        "transportProperties",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ_IF_MODIFIED,
        IOobject::NO_WRITE
    )
);
Info<< "Leyendo constante de difusion DT\n" << endl;
dimensionedScalar DT
(
    transportProperties.lookup("DT")
);
```

y luego de que se definen los campos de velocidad y la presión, hay que definir la concentración, de forma análoga a la presión reemplazando p por C (o cualquier nombre que se le quiera dar a la concentración).

```
Info<< "Leyendo campo escalar concentracion C\n" <<endl;
volScalarField C
(
    IOobject
    (
        "C",
        runTime.timeName(),
        mesh,
```

```
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
);
```

2. Agregar la ecuación de convección-difusión: Fuera del loop del algoritmo *PIMPLE*, agregar las siguiente líneas (simplemente haciendo la analogía $C \rightarrow U$, $DT \rightarrow nu$.

```
fvVectorMatrix CEqn
  (
      fvm::ddt(C)
    + fvm::div(phi, C)
    - fvm::laplacian(DT, C)
  );
  CEqn.solve();
```

3. Cambiar el nombre del ejecutable, en el archivo *files* de la carpeta *Make*. Por ejemplo cambiando,

```
  pimpleFoam.C
  EXE = $(FOAM_APPBIN)/pimpleFoam
```

```
  MareaRojaPimpleFoam.C
  EXE = $(FOAM_USER_APPBIN)/MareaRojaPimpleFoam
```

Finalmente ejecutar haciendo *wmake*.

Capítulo 6

Resultados

En este capítulo se muestra una selección de los resultados obtenidos. Se divide en los resultados obtenidos por MATLAB y OpenFOAM.
Los cálculos se realizaron en un computador con las siguientes características: Procesador Intel(R) Pentium(R) CPU N3540 @2.16 GHz 2.16 GHz, 8.00 GB de memoria RAM y un sistema operativo de 64 bits, procesador basado en x64. Es importante tener en cuenta esto, pues obviamente en un computador más potente aumenta la velocidad en los cálculos. Cuando esté completamente modelado el problema y en particular resuelto el problema con turbulencias, se puede lanzar un cálculo en un intervalo de tiempo mayor (aproximadamente un mes, pues interesa el fenómeno de la marea), como por ejemplo en el cluster del CMM.

6.1. MATLAB

6.1.1. Resolución de las ecuaciones de Navier-Stokes

El número de celdas efectivamente usadas son 76341. El número de caras en las que la presión es variable es 55199 (que son las caras entre las regiones fondo, superficie y martierra), por lo tanto hay un total de 131540 presiones. Luego el tamaño de la matriz de Navier-Stokes es $2 \times 76341 + 360563 = 360563$. Resulta evidente que esta matriz no se puede tratar como una matriz completa en ningún computador común y corriente, por la gran de memoria que requiere. Por ejemplo, al generar una matriz cuadrada de sólo ceros, MATLAB lanza el siguiente error:

```
>> zeros(360563, 360563)
Error using zeros
Requested 360563x360563 (968.6GB) array exceeds maximum array size preference. Creation of arrays greater than this limit may take a long time and cause MATLAB to become unresponsive. See array size limit or preference panel for more information.
```

Por lo tanto, la matriz debe tratarse como *sparse* dado que gran cantidad de sus elementos son 0. Los parámetros utilizados son los siguientes:

- Tiempo inicial $t = 0$
- $T_f = 24$ horas $= 86400$ s el tiempo final.
- $\Delta t = 60$ s.
- $T = 6$ horas el período de mareas.
- $A = 2$ m la amplitud de marea.

Dado que se aborda solamente el caso laminar, las viscosidades no deben ser demasiado pequeñas (10^{-6} viscosidad cinemática del mar). Se exploran los casos $\nu = 0{,}01$ y $\nu = 1$.

La figura 6.1 muestra el patrón de dispersividad de la matriz de Navier-Stokes para $t = 60$ s, $\nu = 0{,}01$.

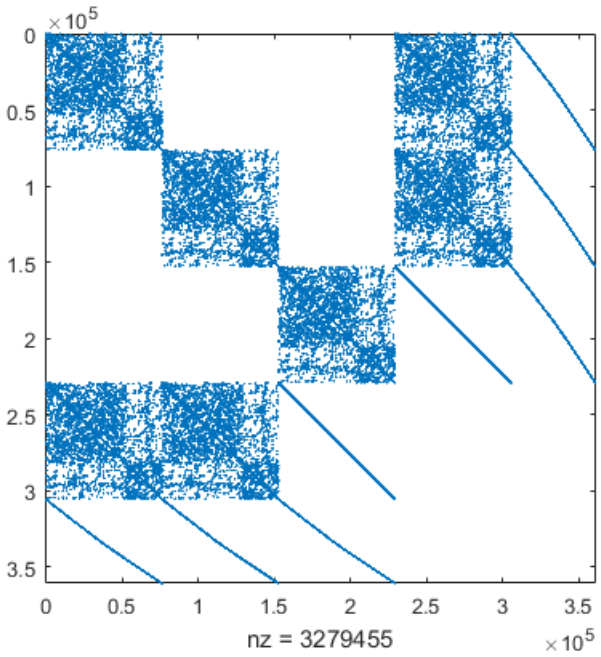
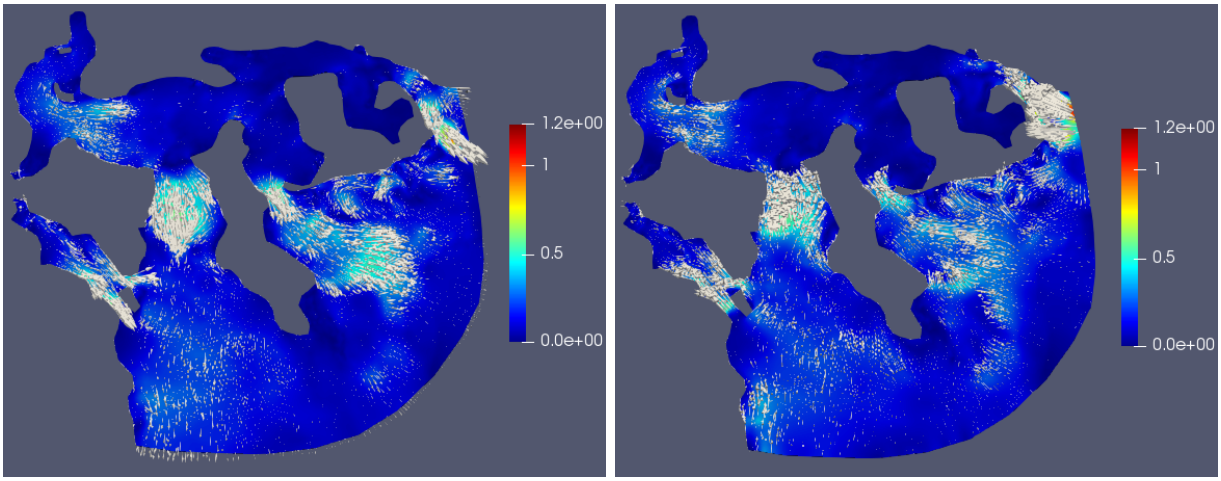


Figura 6.1: Patrón de dispersividad matriz de la ecuación de Navier-Stokes discretizada usando volúmenes finitos.

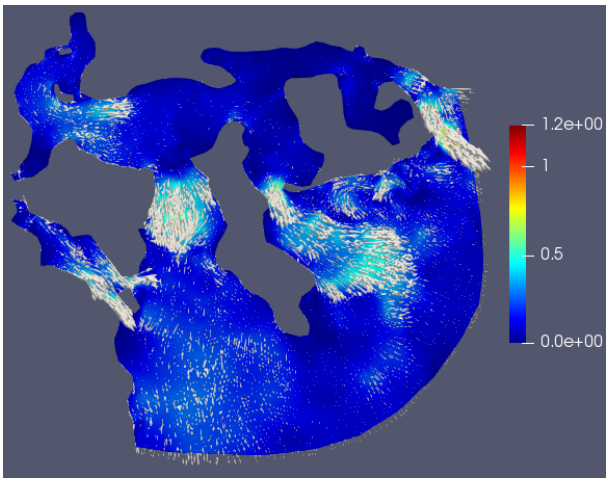
El número de elementos no nulos es 3279455. En la diagonal se logra apareciar claramente los bloques de la matriz asociados a la velocidad; A_{UU}, A_{VV}, A_{WW}. Fuera de la diagonal, están los bloques asociados a la presión definida sobre volúmenes finitos A_{UP} y A_{VP}. Las tres líneas en los costados corresponden a la presión en los bordes. Se observa además, que la matriz es simétrica (lo cual se comprueba en el computador).

Primero se prueba con $\nu = 0{,}01$.

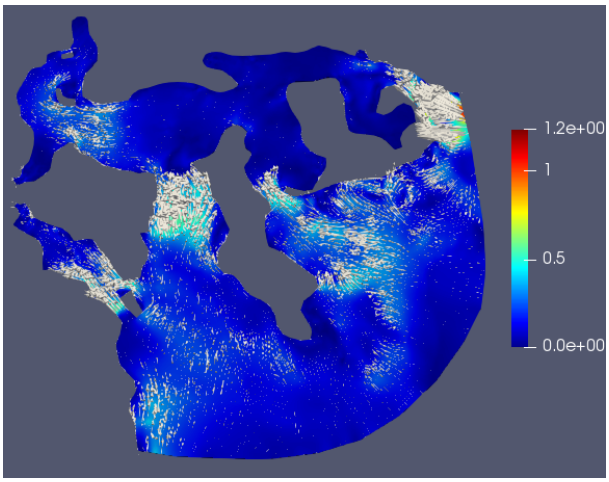


(a) t = 3 hrs, $\nu = 0{,}01$.

(b) t = 9 hrs, $\nu = 0{,}01$.



(c) t = 15 hrs, $\nu = 0{,}01$.



(d) t = 21 hrs, $\nu = 0{,}01$.

Figura 6.2: Magnitud de la velocidad de marea, obtenida de resolver las ecuaciones de Navier-Stokes programada en MATLAB.

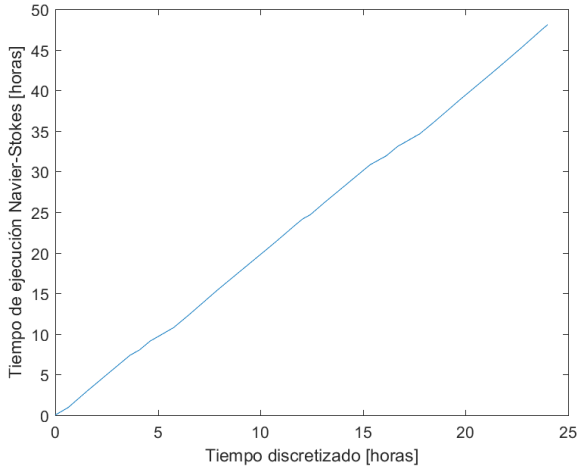


Figura 6.3: Gráfico tiempo de ejecución de la ecuación de Navier-Stokes implementada en MATLAB.

El tiempo de ejecución del programa resultó ser de 173132 s, es decir, 48 horas con 5 mins.

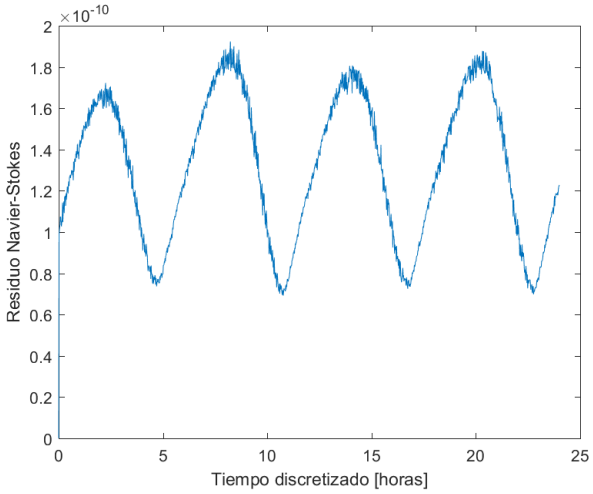


Figura 6.4: Gráfico del residuo de la ecuación de Navier-Stokes discretizada implementada en MATLAB, usando la velocidad calculada con MATLAB para una viscosidad $\nu = 0{,}01$.

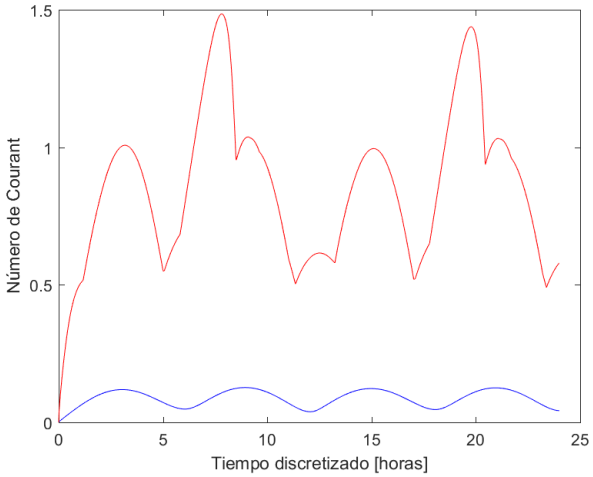


Figura 6.5: Gráfico del Número de Courant obtenido con MATLAB $\nu = 0{,}01$.

El número de Courant promedio sigue una onda, con período de aproximadamente 6 horas como muestra la figura 6.5. El número de Courant máximo pese a sobrepasar el valor 1, el método no diverge, lo cual habla de la estabilidad del método, sin embargo, cuando el valor alcanza el máximo genera perturbaciones en el gráfico del número de Courant máximo, ya que lo esperable es que tenga el mismo comportamiento que el número de Courant promedio pero con amplitud mayor.

6.1.2. Resolución de la ecuación de convección-difusión

Como en este caso no se cuenta con datos (y no tiene sentido hablar concentración inicial nula, pues no habría que transportar) se considera una concentración inicial de un cilindro en el centro del dominio, cuyas dimensiones son :

- Centro de la mancha en el eje x : $\frac{x_{min} + x_{max}}{2}$
- Centro de la mancha en el eje y : $\frac{y_{min} + y_{max}}{2}$
- Radio de la mancha: $\frac{x_{max} - x_{min}}{5}$
- $C_{\mathrm{i}} = 1$ si el volumen finito i está dentro del cilindro, $C_{\mathrm{i}} = 0$ fuera del cilindro.

Luego la concentración inicial es

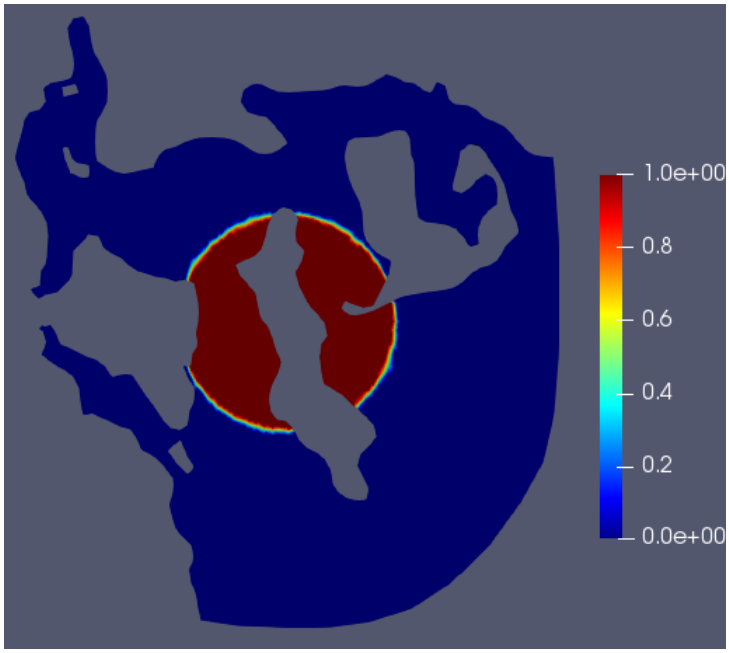


Figura 6.6: Concentración inicial: mancha circular.

La figura 6.7 muestra el patrón de dispersividad de la matriz de la ecuación de convección-difusión discretizada.

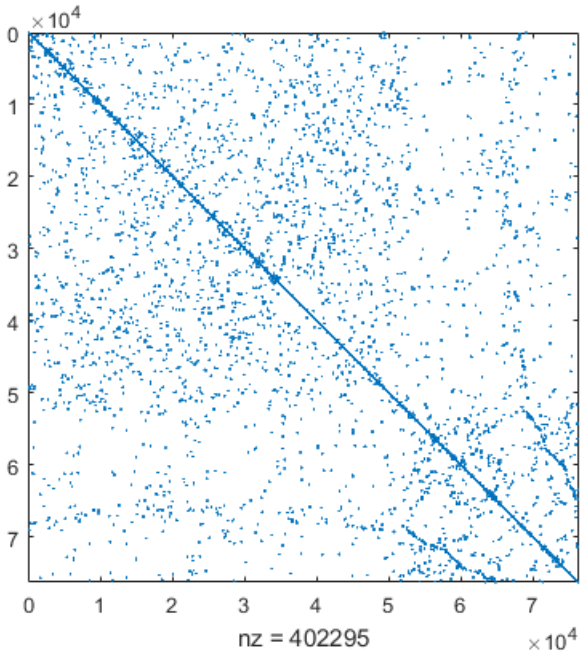
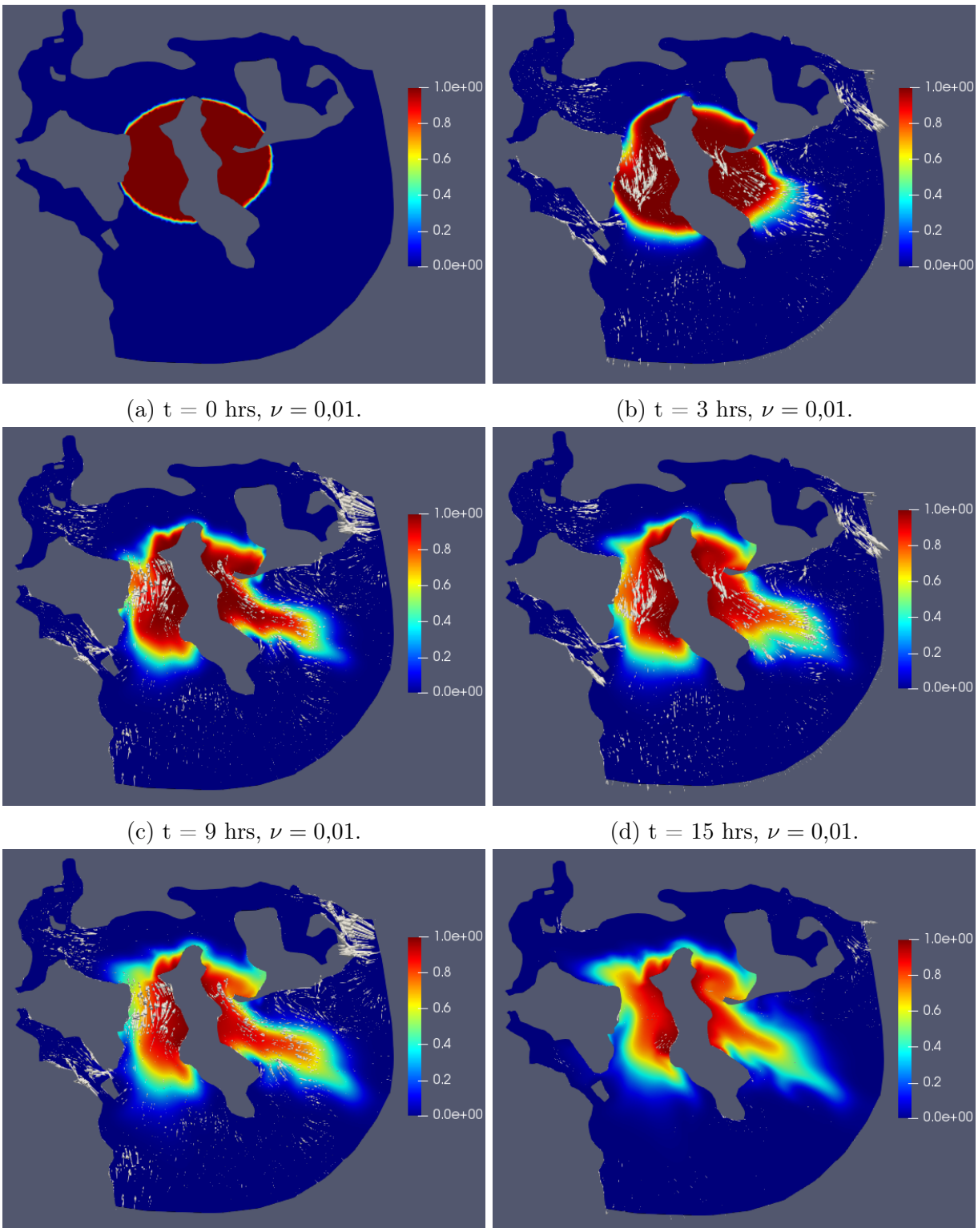


Figura 6.7: Patrón de dispersividad matriz de la ecuación de convección-difusión discretizada usando volúmenes finitos.

A continuación se muestra la concentración, obtenida al simular la ecuación de convección difusión.



(a) t = 0 hrs, $\nu = 0{,}01$.

(b) t = 3 hrs, $\nu = 0{,}01$.

(c) t = 9 hrs, $\nu = 0{,}01$.

(d) t = 15 hrs, $\nu = 0{,}01$.

(e) t = 21 hrs, $\nu = 0{,}01$.

(f) t = 24 hrs, $\nu = 0{,}01$.

Figura 6.8: Magnitud de la concentración de microalgas, obtenida de la ecuación de convección-difusión programada en MATLAB.

El programa demoró 25 minutos en obtener la solución y escribir los archivos para poste-

riormente leerlos en ParaView.

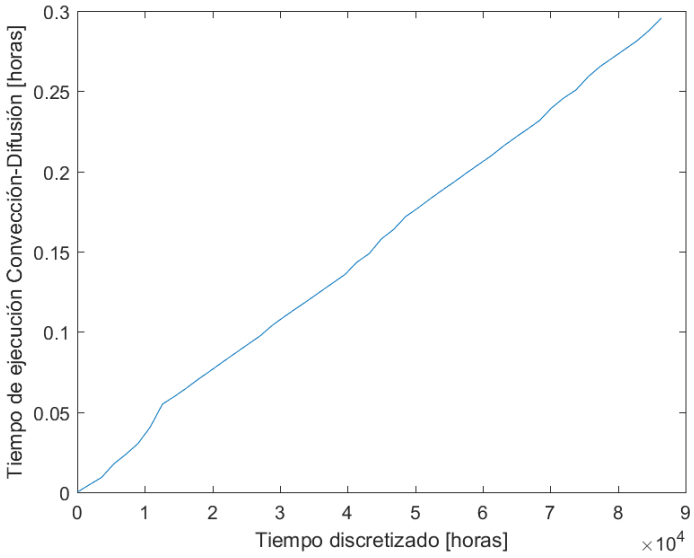


Figura 6.9: Gráfico tiempo de ejecución de la ecuación de convección-difusión implementada en MATLAB.

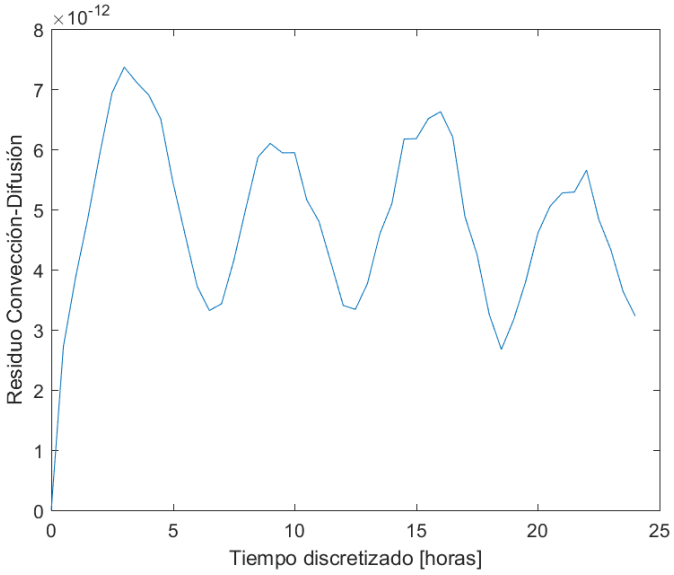


Figura 6.10: Gráfico del residuo de convección-difusión implementada en MATLAB, usando la velocidad calculada con MATLAB para una viscosidad $\nu = 0{,}01$.

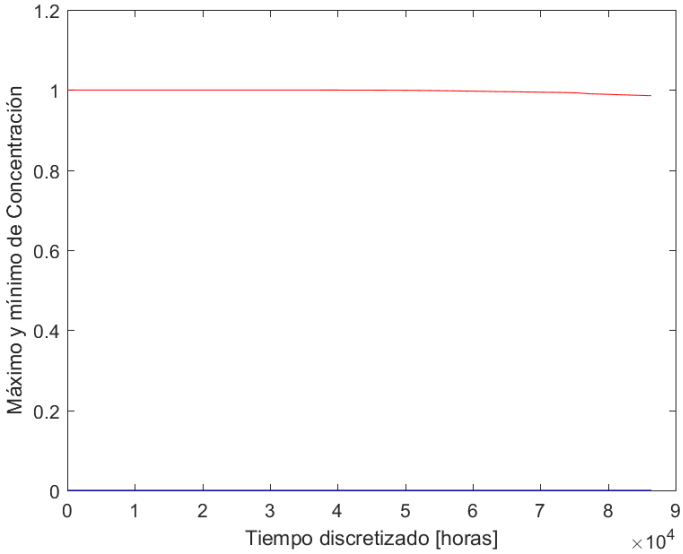


Figura 6.11: Gráfico del máximo y mínimo de concentración en alguna celda del dominio en cada tiempo, obtenido con MATLAB y usando la velocidad calculada con MATLAB para una viscosidad $\nu = 1$.

El gráfico 6.12 muestra el comportamiento del número de Peclet, definido para cada volumen como la suma de los coeficientes de la convección dividido la suma de los coeficientes de difusión. Es el análogo al número de Reynolds (que se estudia para Navier-Stokes), pero para la ecuación de convección-difusión. Este gráfico muestra que con los parámetros dados, la convección se hace más importante que la difusión y puede llegar a ser hasta 80 veces mayor el aporte de la convección.

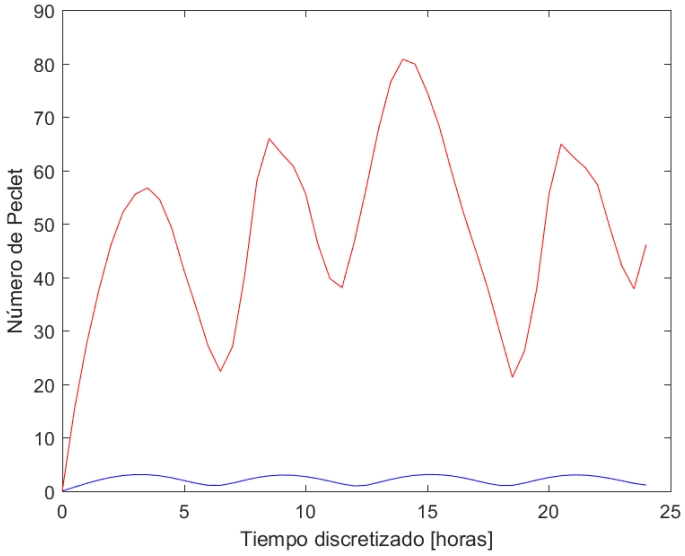


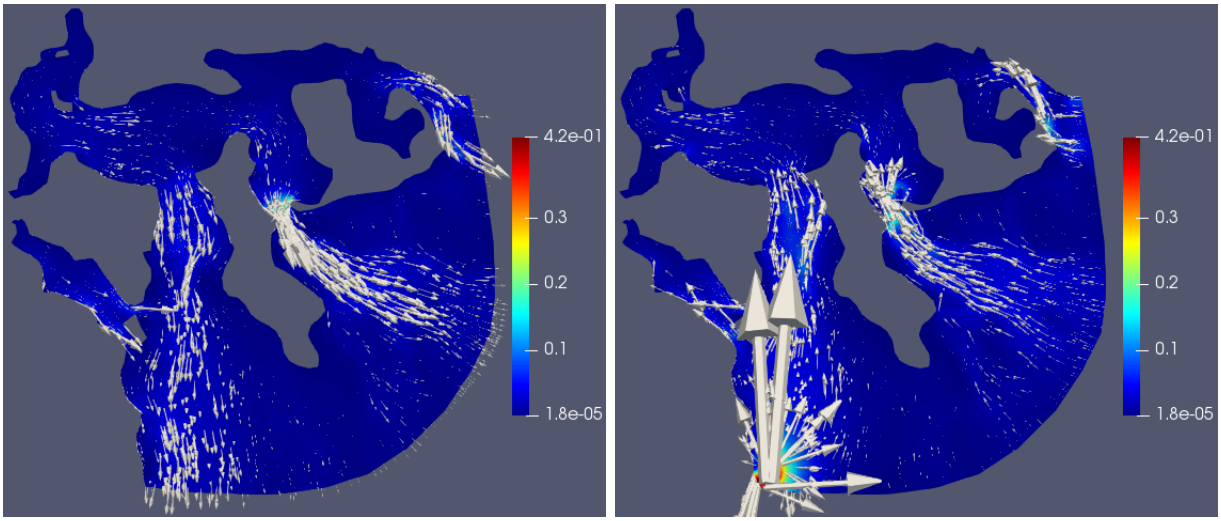
Figura 6.12: Tiempo discretizado [horas] vs Número de Peclet.

6.2. OpenFOAM

6.2.1. MareaRojaPimpleFoam

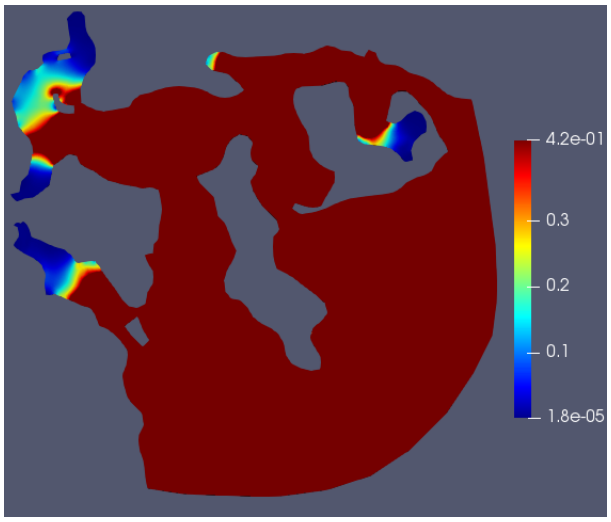
Se utilizan los parámetros $\Delta t = 60$ s el salto temporal, $A = 2$ m la amplitud y período de mareas $T = 12$ hrs.

- $\nu = 0{,}01$:



(a) t = 3 hrs.

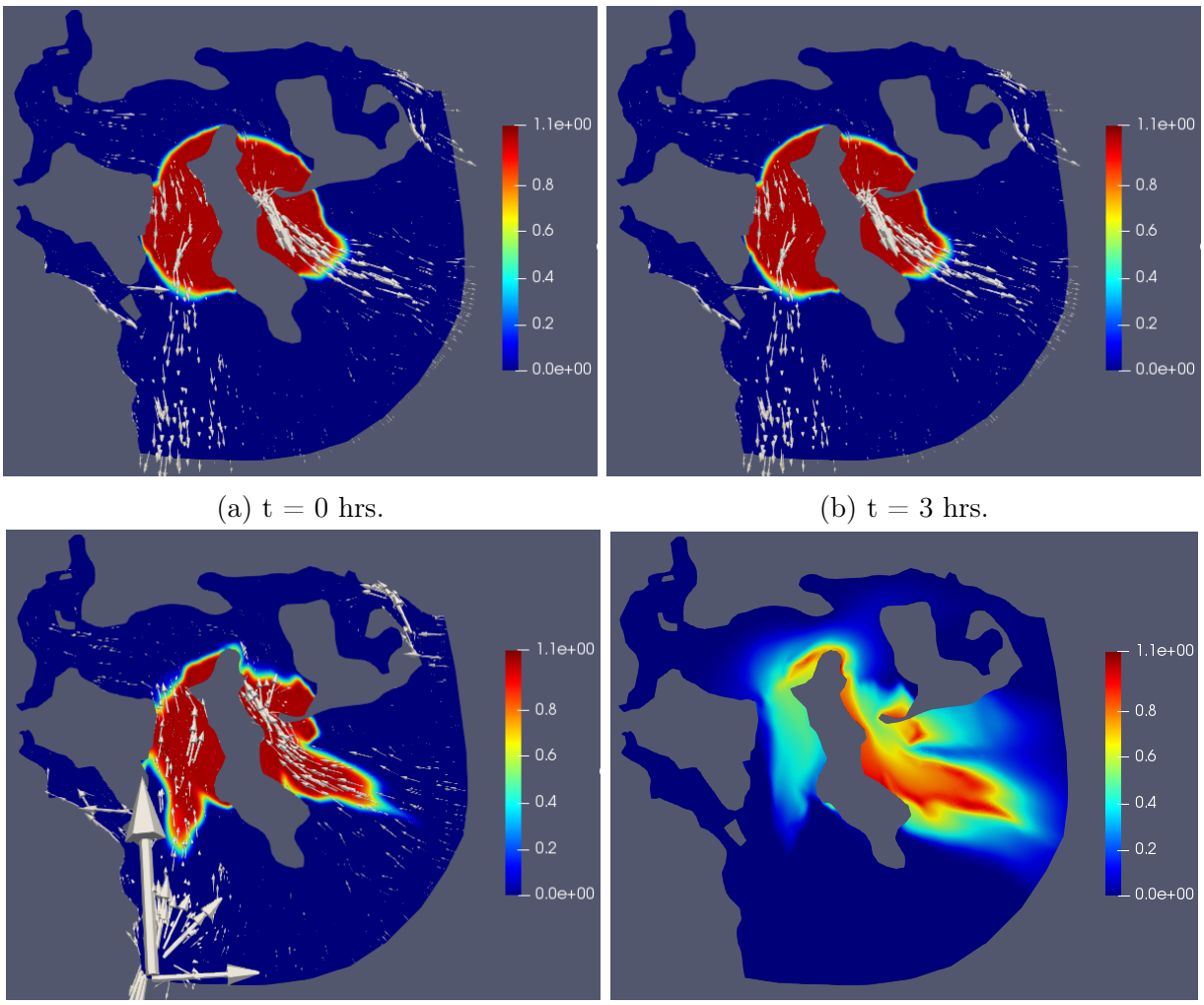
(b) t = 9 hrs.



(c) t = 9 hrs, 3 mins.

Figura 6.13: Magnitud de la velocidad de la marea obtenida por las ecuaciones de Navier-Stokes.

La figura 6.13, muestra que para viscosidad $\nu = 0{,}01$ las solución a las ecuaciones de Navier-Stokes diverge a partir de las 9 horas; la velocidad aumenta dramáticamente en 3 minutos como queda en evidencia en la figura 6.13c.



(a) t = 0 hrs. (b) t = 3 hrs.

(c) t = 9 hrs. (d) t = 9 hrs, 3 mins.

Figura 6.14: Magnitud de la concentración de la marea obtenida por la ecuación de convección-difusión.

A continuación se muestran una serie de figuras importantes para analizar los resultados y su desempeño.

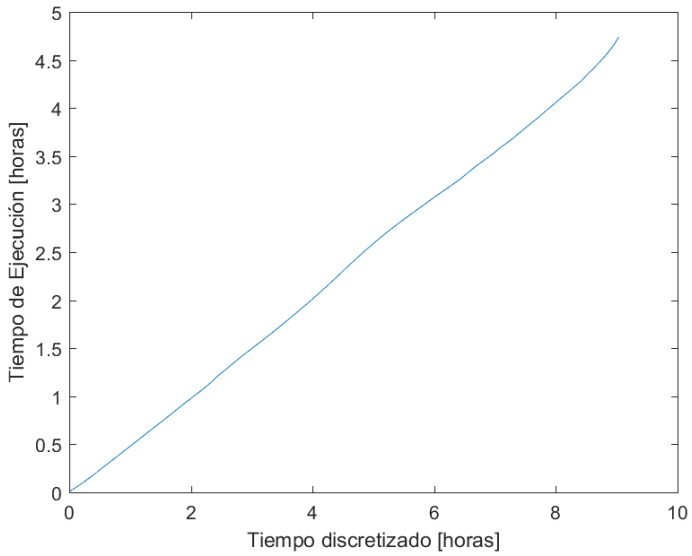


Figura 6.15: Gráfico tiempo de ejecución del solver *MareaRojaPimpleFoam*.

La figura 6.15 muestra el desempeño del solver; el cual demora aproximadamente la mitad del tiempo de simulación. En simular 9 horas y 3 minutos (aunque el tiempo final era $T = 24$ horas el solver divergió en 9 horas con 3 minutos) demoró 17142 segundos, es decir, 4 horas, 45 minutos y 42 segundos.

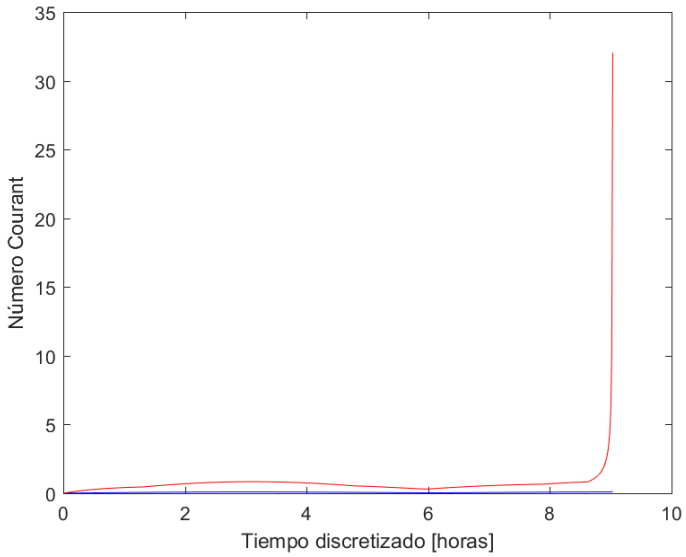
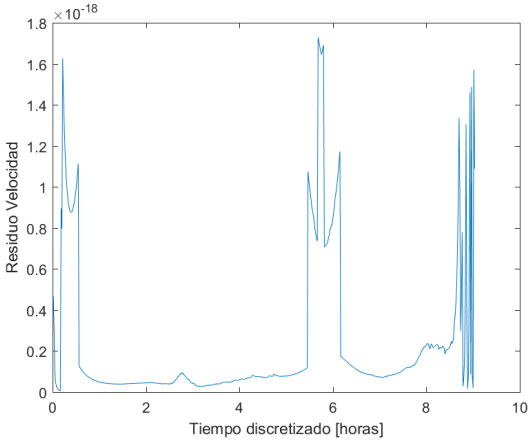


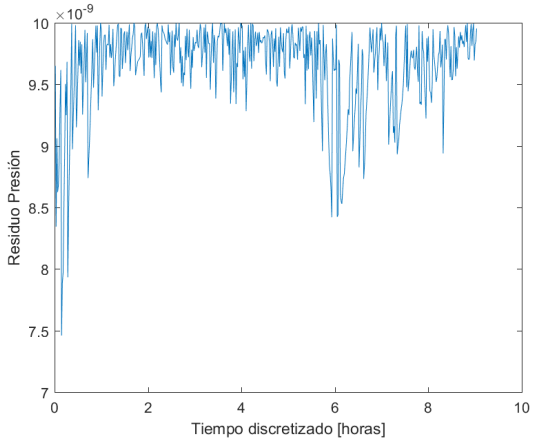
Figura 6.16: Gráfico del Número de Courant obtenido con el solver *MareaRojaPimpleFoam* para una viscosidad $\nu = 0{,}01$.

El gráfico 6.16 indica lo importante que es que se cumpla la condición CFL $Co \geq 1$. En este caso a los 31440 s el número de Courant dejó de cumplir la condición CFL 1,02867 y en cuanto se supera esta cota, la velocidad aumenta exponencialmente. En esta línea, para probar con un Δt más pequeño y ver si así se mantiene la condición CFL, se usó

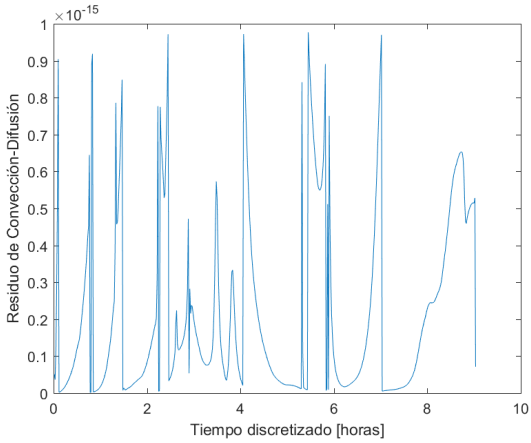
$\Delta t = 20$ s y aún así, a los 32280 s el número de Courant máximo fue 1,00087, luego volvió a diverger.



(a) Residuo de la ecuación de velocidad.



(b) Residuo de la ecuación de la presión.



(c) Residuo obtenido en la ecuación de convección-difusión.

Figura 6.17: Residuos finales obtenidos por el solver *MareaRojaPimpleFoam*.

Notar que el orden de los residuos es mayor para la concentración C, pues es la ecuación más fácil de resolver (es una ecuación cuya incógnita es campo escalar). En OpenFOAM, se debe fijar un residuo como tolerancia al momento de iterar y hallar las soluciones, en este caso se fijó 10^{-9} para la velocidad y 10^{-8} para la presión, puesto que se demoraba más en hallar la presión. Otro problema del solver, es que la concentración tiene mucho error, pues la concentración inicial tenía valores en $[0, 1]$, por lo cual el efecto de la difusión implica que la solución debe seguir tomando valores en ese intervalo, pero moviéndose a zonas de menor concentración y no a zonas de mayor concentración. El error llega a ser del 20 %. Esto motiva a probar usando las velocidades obtenidas con OpenFOAM, resolver convección-difusión con MATLAB y ver si las velocidades están mal calculadas, o es un problema de la ecuación de convección-difusión.

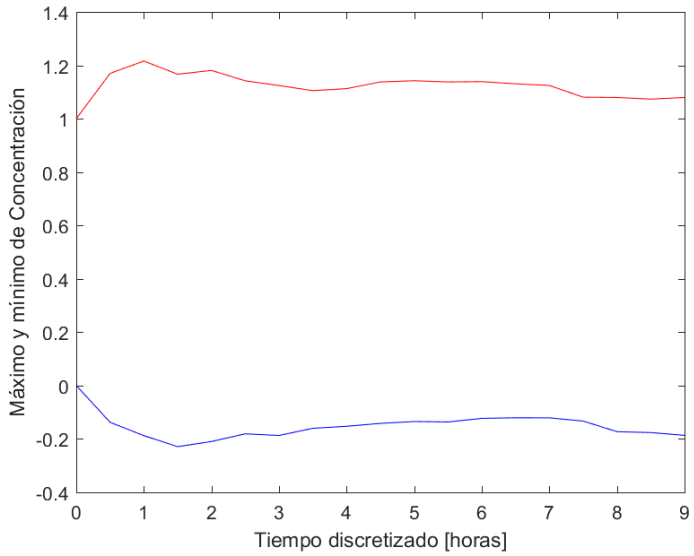
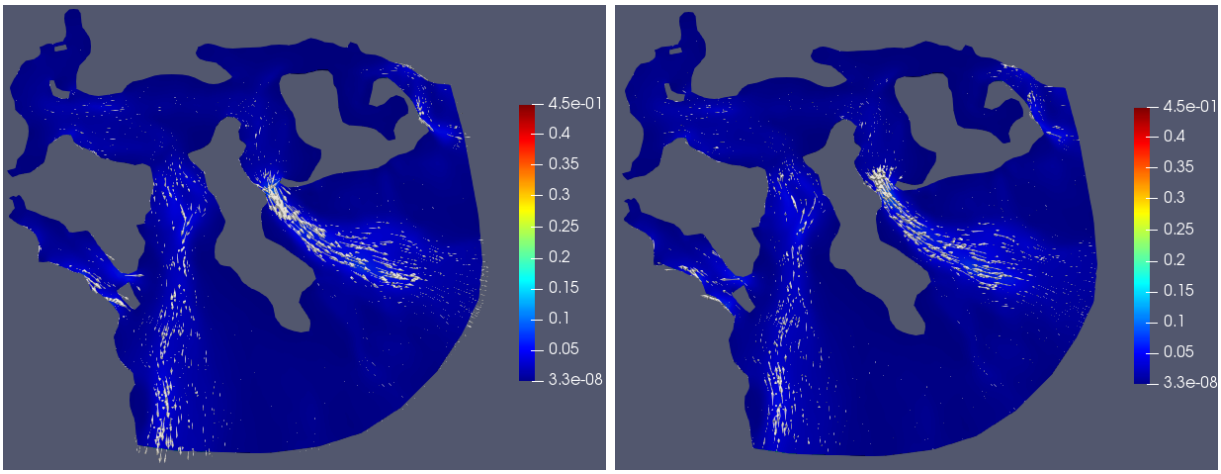


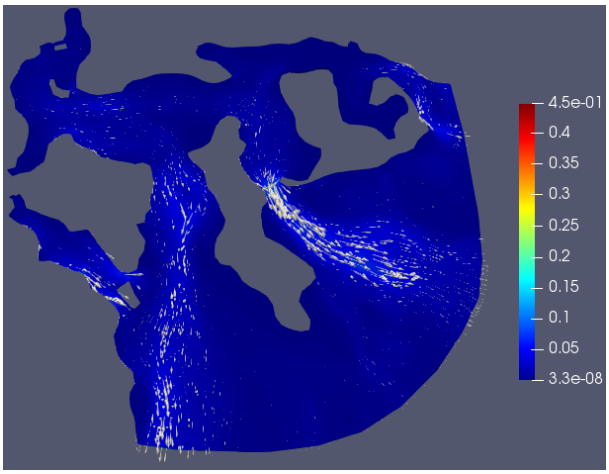
Figura 6.18: Gráfico del máximo y mínimo de concentración en alguna celda del dominio en cada tiempo, obtenido con el solver *MareaRojaPimpleFoam* para una viscosidad $\nu = 0{,}01$.

- $\nu = 1$:

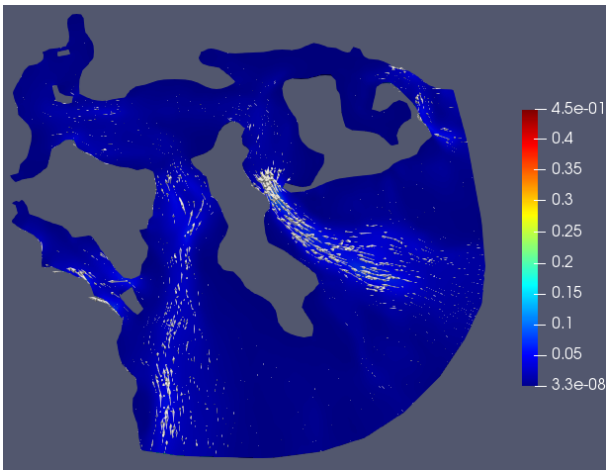


(a) t = 3 hrs.

(b) t = 9 hrs.

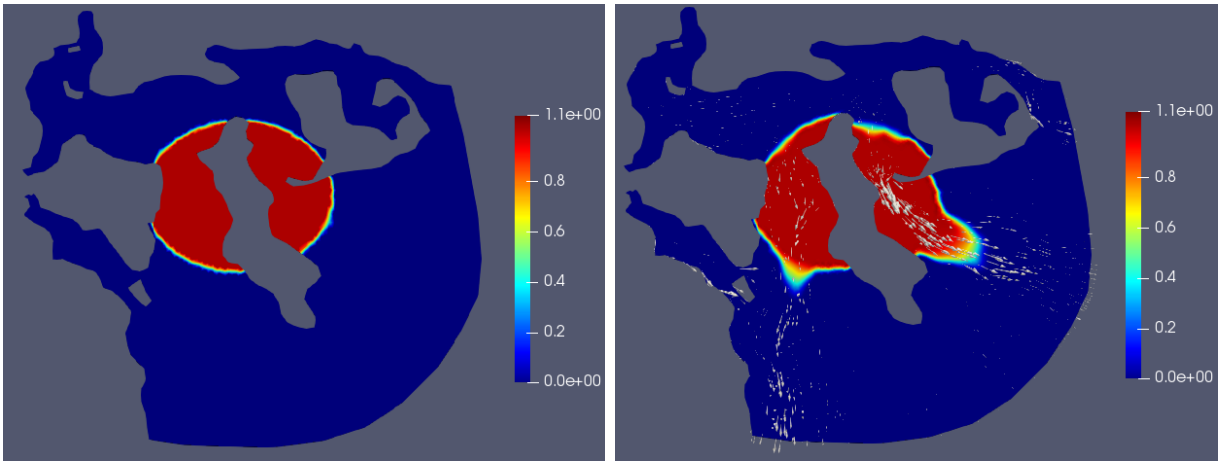


(c) t = 15 hrs.



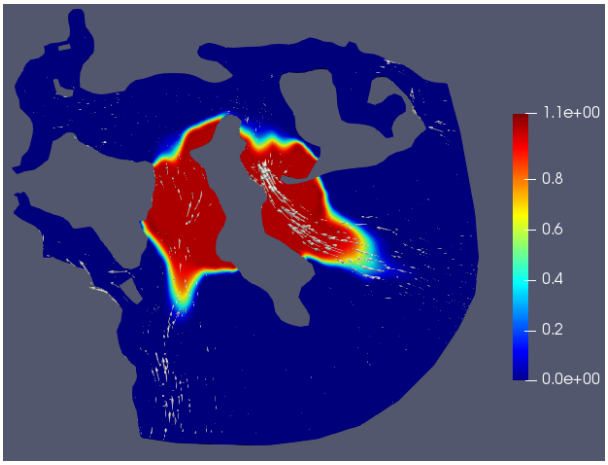
(d) t = 21 hrs.

Figura 6.19: Magnitud de la velocidad de la marea obtenida por las ecuaciones de Navier-Stokes.

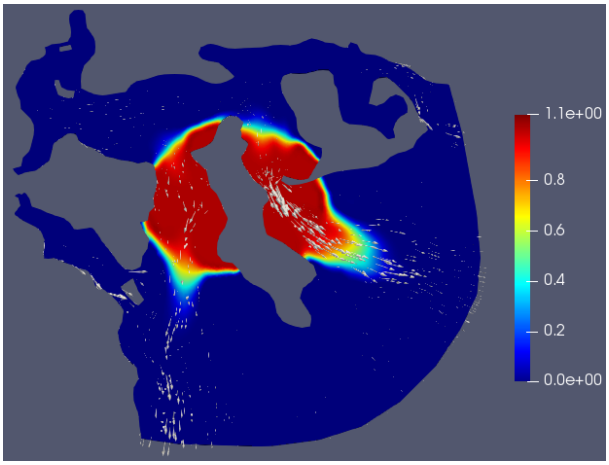


(a) t = 0 hrs.

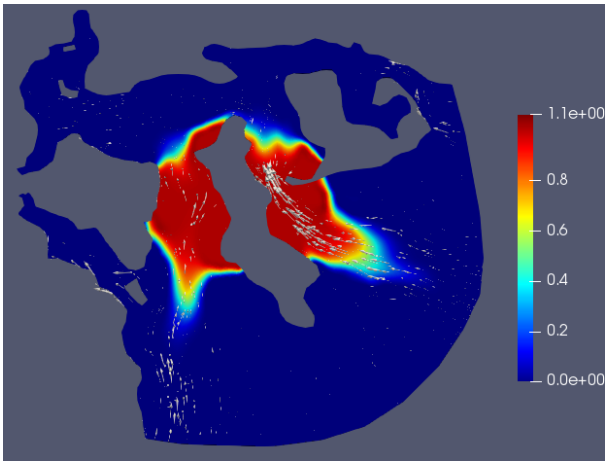
(b) t = 3 hrs.



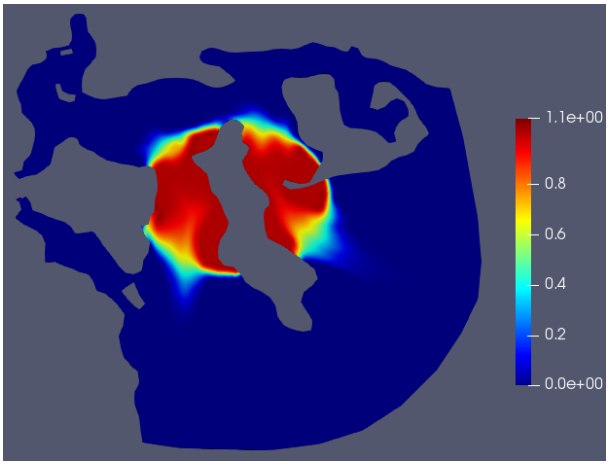
(c) t = 9 hrs.



(d) t = 15 hrs.



(e) t = 21 hrs.



(f) t = 24 hrs.

Figura 6.20: Concentración de la marea obtenida por la ecuación de convección-difusión.

En simular 1 día = 86400 s, el programa demoró 42379 s, es decir, 11 horas, 46 minutos y 19 segundos. Ver el gráfico de la figura 6.21, el cual es aproximadamente lineal.

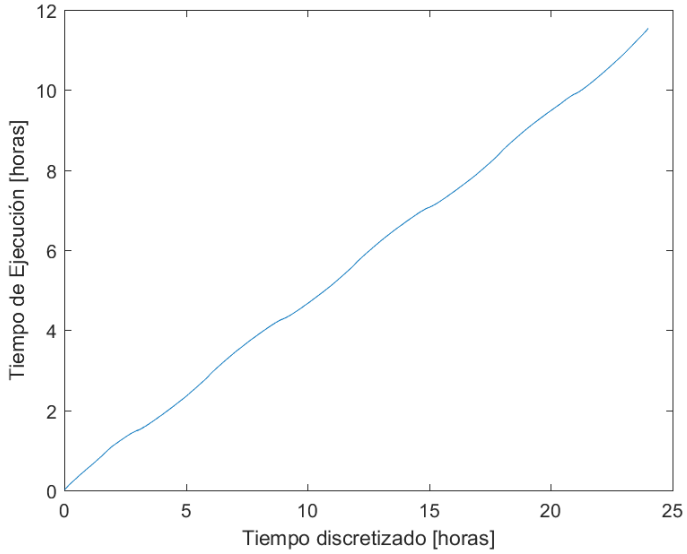


Figura 6.21: Gráfico tiempo de ejecución del solver *MareaRojaPimpleFoam*, con viscosidad $\nu = 1$.

El número de Courant en este caso nunca excede el valor 1, por lo cual no hay divergencias. Más aún, tanto el Número de Courant máximo en alguna celda como el promedio sobre el dominio muestra un comportamiento periódico. El número de Courant máximo es practicamente un orden de magnitud mayor que el número de Courant promedio.

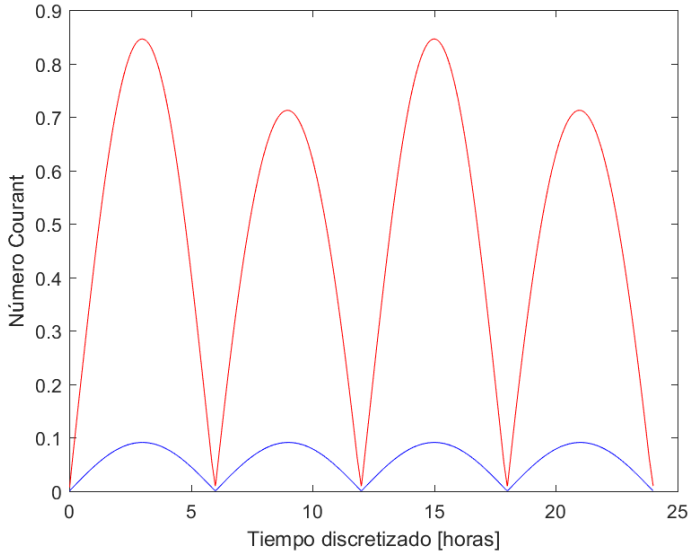
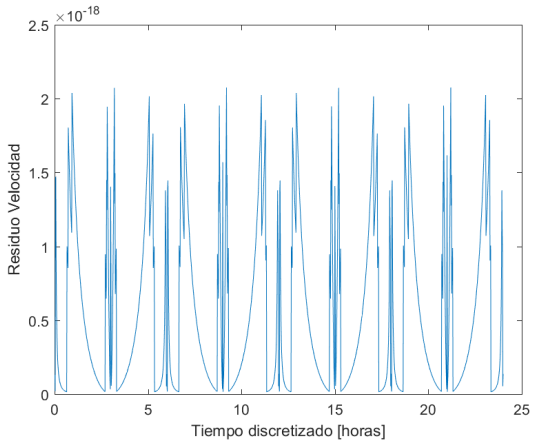
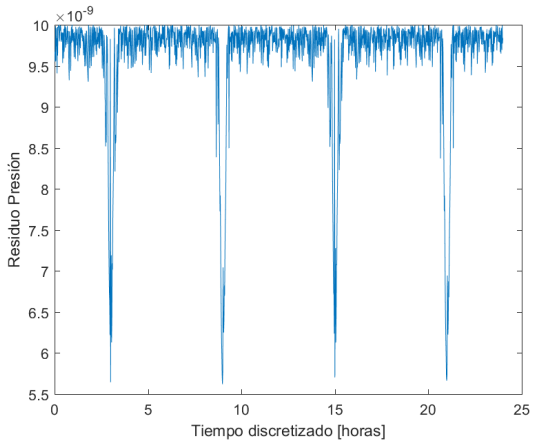


Figura 6.22: Gráfico del Número de Courant obtenido con el solver *MareaRojaPimpleFoam* para una viscosidad $\nu = 1$.

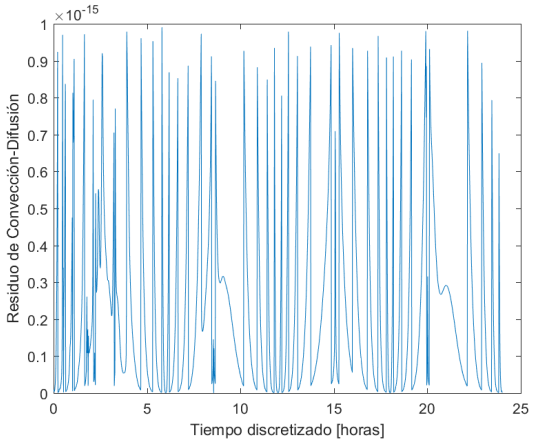
El orden de magnitud sigue siendo el mismo que en el caso anterior $\nu = 0{,}01$. Notar que el residuo de la presión se hace más pequeño en la cercanía de los $3, 9, 15, 21$ horas que es donde se aprecia el efecto de subida/bajada de la marea. Esto queda claro en la figura 6.23b.



(a) Residuo de la ecuación de velocidad.



(b) Residuo de la ecuación de la presión.



(c) Residuo obtenido en la ecuación de convección-difusión.

Figura 6.23: Residuos finales obtenidos por el solver *MareaRojaPimpleFoam* para una viscosidad $\nu = 1$.

El máximo y mínimo de la concentración en el tiempo nuevamente muestra el extraño comportamiento de que el máximo supera el valor 1 y el mínimo disminuye más que el valor 0. En este caso, el mínimo alcanza hasta un valor $-0{,}4$ lo cual representa un error de hasta el 40 %, pues la concentración inicial vive en $[0, 1]$. Ojo que los valores negativos de la concentración no es el problema, pues son sólo valores, perfectamente se puede usar otro intervalo $[c_1, c_2]$ con $c_1 > 0$, y $c_2 > 1$ así no se tomarían los valores negativos; el problema radica en que la solución deja de pertenecer al intervalo inicial.

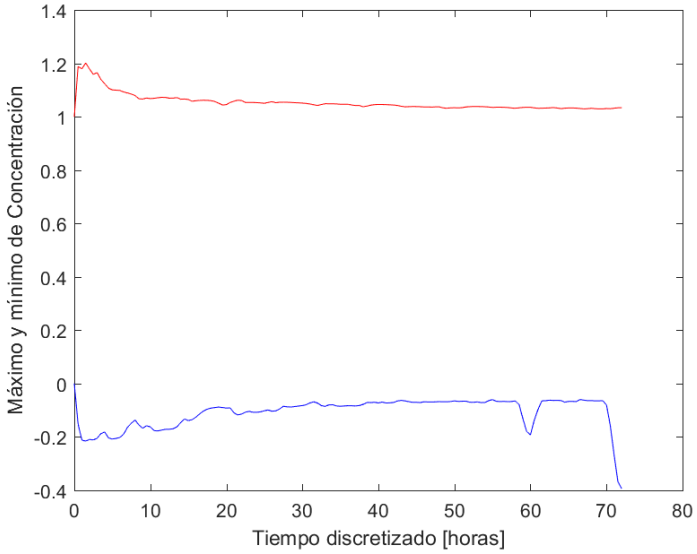
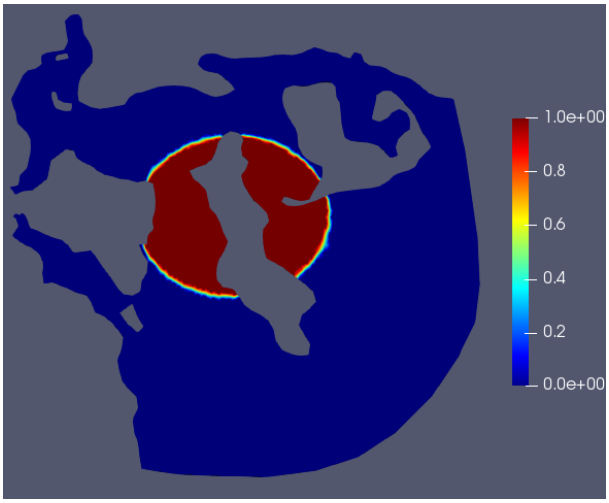


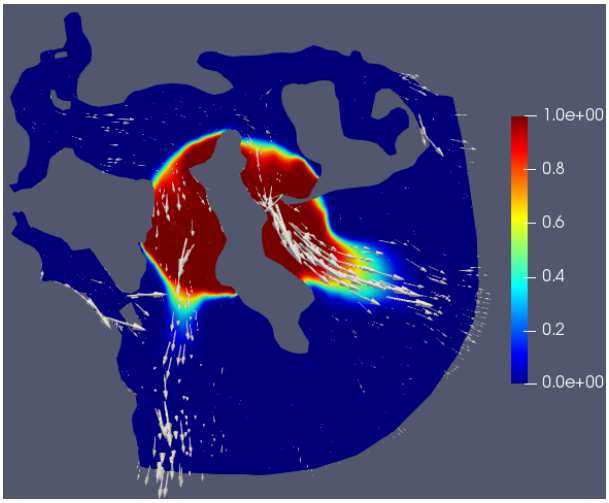
Figura 6.24: Gráfico del máximo y mínimo de concentración en alguna celda del dominio en cada tiempo, obtenido con el solver *MareaRojaPimpleFoam* para una viscosidad $\nu = 1$.

6.2.2. Velocidad con OpenFOAM y concentración con MATLAB

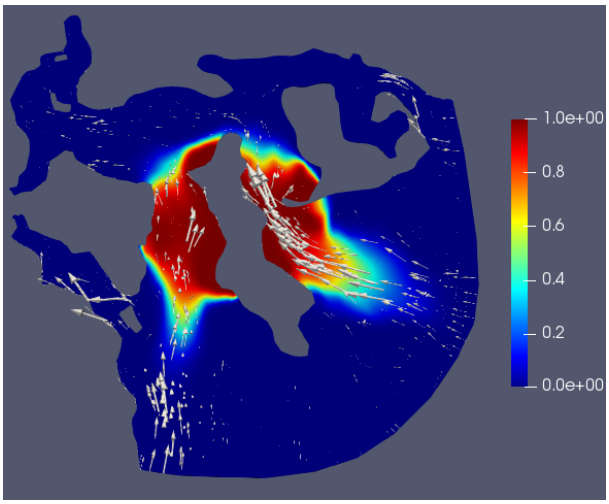
En vista del problema en la concentración que presenta el solver *MareaRojaPimpleFoam*, se utiliza la velocidad obtenida con OpenFOAM, para hallar la concentración mediante la ecuación de convección-difusión programada en MATLAB.



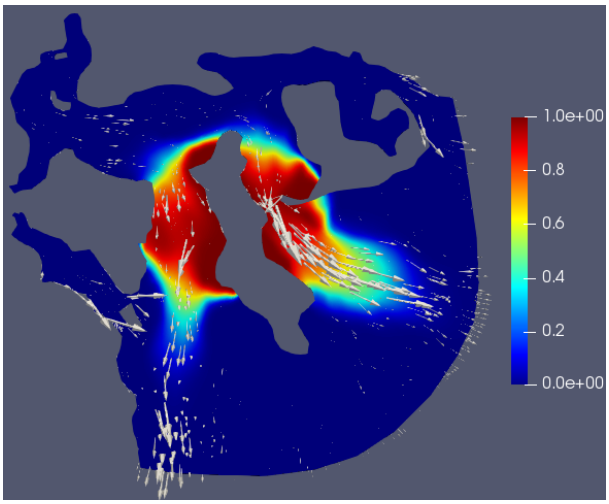
(a) t = 0 hrs.



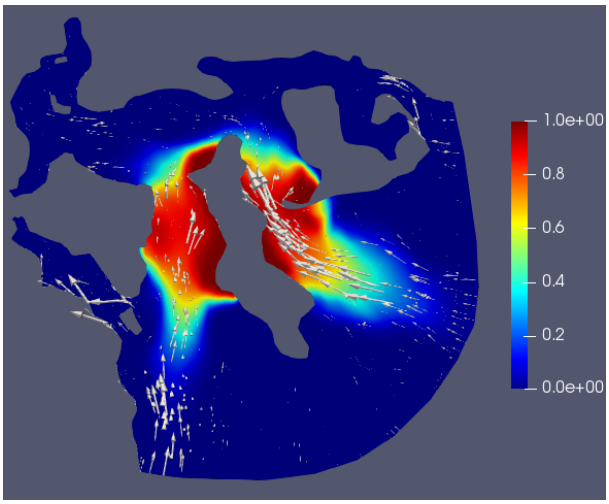
(b) t = 3 hrs.



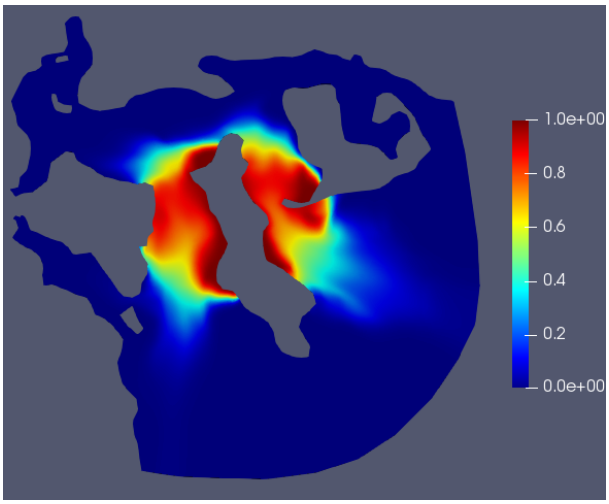
(c) t = 9 hrs.



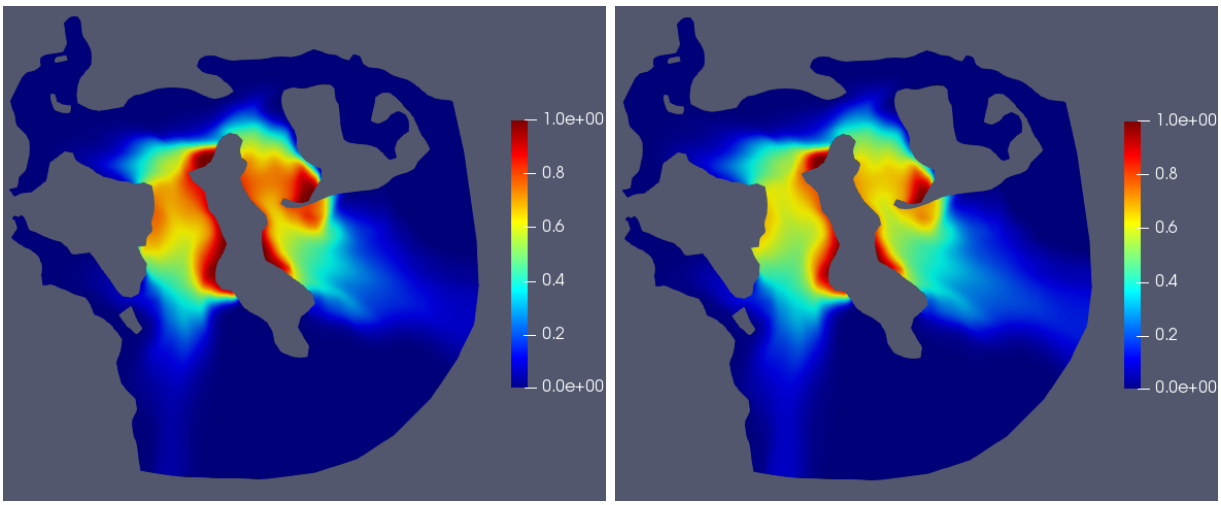
(d) t = 15 hrs.



(e) t = 21 hrs.



(f) t = 24 hrs.



(g) t = 48 hrs. (h) t = 72 hrs.

Figura 6.25: Concentración de la marea obtenida por la ecuación de convección-difusión.

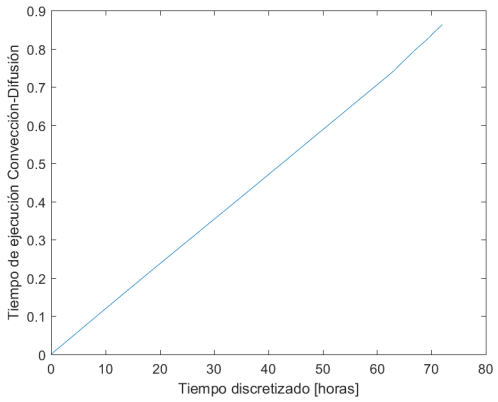


Figura 6.26: Gráfico tiempo de ejecución de convección-difusión resuelta en MATLAB y usando la velocidad calculada con *pimpleFoam* para una viscosidad $\nu = 1$.

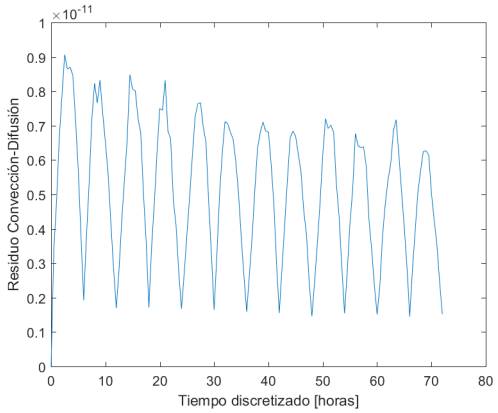


Figura 6.27: Gráfico del residuo de convección-difusión resuelta en MATLAB y usando la velocidad calculada con *pimpleFoam* para una viscosidad $\nu = 1$.

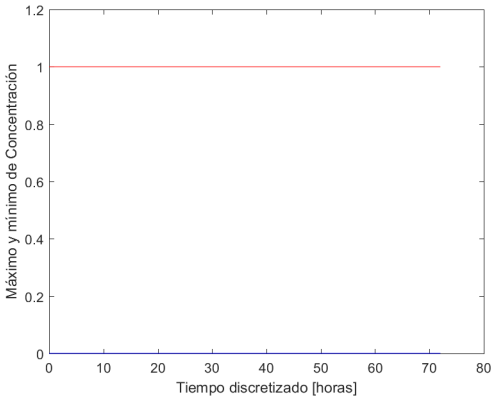
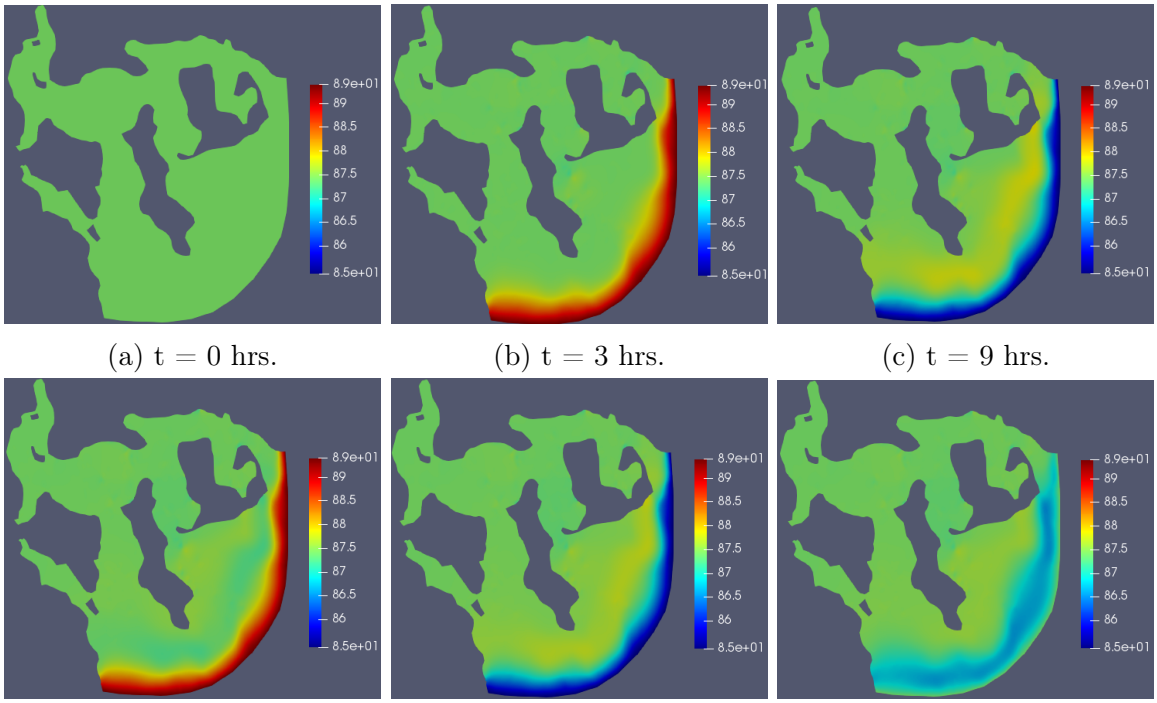


Figura 6.28: Gráfico del máximo y mínimo de concentración en alguna celda del dominio en cada tiempo, obtenido con MATLAB y usando la velocidad calculada con *pimpleFoam* para una viscosidad $\nu = 1$.

6.2.3. hTotalshallowWaterFoamConNroMaximoCourant

Usando la condición de borde de marea en Mar-Mar: $h_{Total}(x,t) = h_{Total}(x,0) + Asen(\omega t)$, con $A = 2m$ se obtuvieron los siguientes resultados en las figuras 6.29 , 6.30 y 6.31. En la



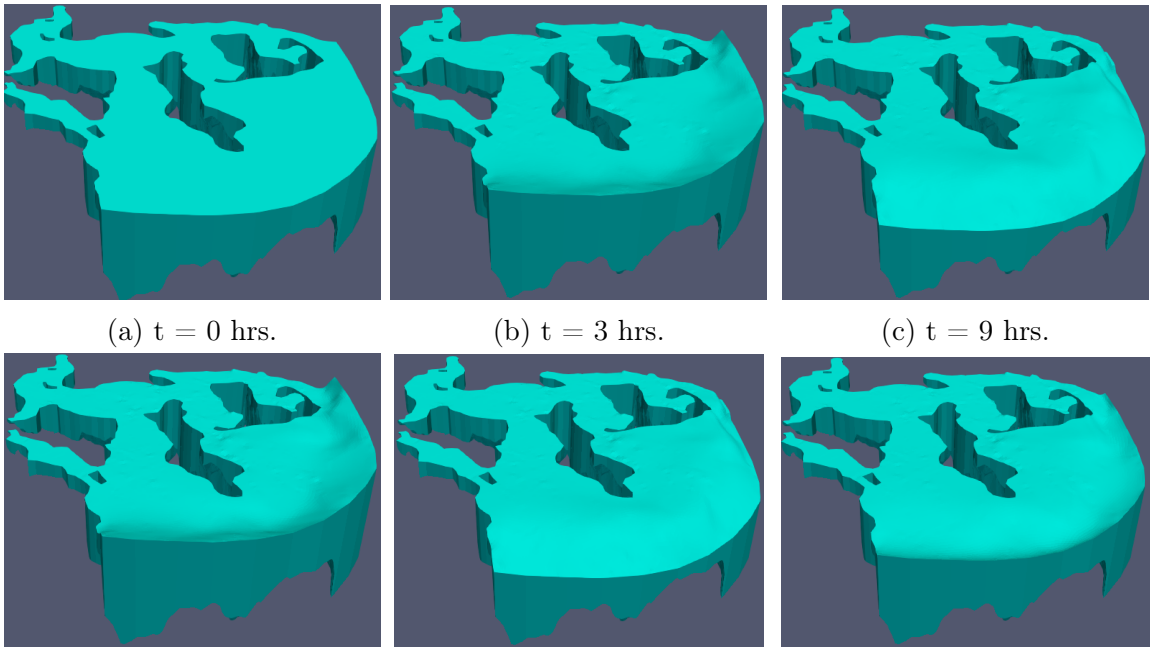
(a) t = 0 hrs. (b) t = 3 hrs. (c) t = 9 hrs.

(d) t = 15 hrs. (e) t = 21 hrs. (f) t = 24 hrs.

Figura 6.29: Nivel de la marea h_{Total} obtenido por las ecuaciones de Shallow Water.

figura 6.29 se aprecia que transcurridas las 3 horas se alcanza el máximo de altura de marea, luego comienza a bajar el mar y se alcanza el mínimo a las 9 horas. Luego se repite cada 12 horas por la periodicidad que se explicó en capítulos anteriores del fenómeno de la marea.

Notar que la onda de marea se empieza a propagar desde la región Mar-Mar y no alcanza a llegar a Mar-Tierra. A partir de estos resultados, se programa en MATLAB para construir una malla dinámica considerando la elevación de la marea obtenida por Shallow Water. Para generar esta malla evolutiva basta guardar en cada tiempo la carpeta *polyMesh* con los *points* de tal tiempo. La figura 6.30 permite ver la evolución 3-D de la marea, sin embargo no se

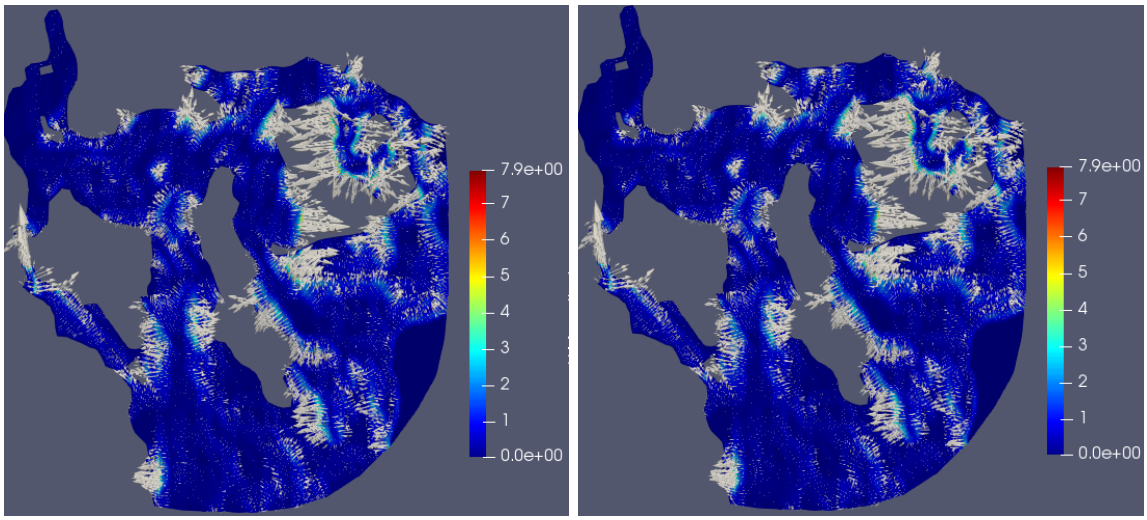


(a) t = 0 hrs. (b) t = 3 hrs. (c) t = 9 hrs.

(d) t = 15 hrs. (e) t = 21 hrs. (f) t = 24 hrs.

Figura 6.30: Dominio evolutivo, con la elevación de la superficie obtenido en Shallow Water.

alcanza a reproducir el fenómeno de la marea, sino que parece más bien oleaje (al menos en la parte Mar-Mar). En la figura 6.31 se aprecia que la velocidad obtenida no reproduce el



(a) t = 3 hrs. (b) t = 9 hrs.

Figura 6.31: Velocidad horizontal obtenida por las ecuaciones de Shallow Water.

fenómeno de la marea, pues el sentido de las flechas no apuntan hacia mar-tierra cuando sube

la marea, ni hacia mar-mar cuando baja la marea como si sucedía en los solvers anteriores.
El tiempo de ejecución del solver fue 7808 segundos, es decir, 2 horas, 10 minutos.
De las figuras obtenidas, se deduce que las condiciones de borde no reproducen bien el fenómeno. Las ecuaciones de Shallow Water presentan un comportamiento oscilatorio, por lo cual el mar no sube uniformemente o baja uniformemente como debería suceder (por la naturaleza global de la atracción gravitacional de la Luna o el Sol sobre la porción de mar). Se probaron otras condiciones de borde pero tampoco se obtuvieron los resultados esperos. Una opción que se exploró fue usar velocidad normal sinusoidal en Mar-Mar, el problema de esta condición es que si bien es acorde al problema que la velocidad normal sea sinusoidal, no se conocen los valores o la escala de estos valores, como si se saben en la superficie (que es la condición de borde que se usa en los solvers para Navier-Stokes).

Conclusión

Del estudio de los métodos implementados en OpenFOAM y MATLAB, se concluye lo siguiente

- El solver de OpenFOAM es aproximadamente cuatro veces más rápido que las implementaciones en MATLAB; el solver de OpenFOAM demora la mitad que el tiempo a simular, mientras que la implementación en MATLAB de Navier-Stokes demora el doble del tiempo a simular.
- La ecuación de convección-difusión es más simple de resolver que la de Navier-Stokes. Las principales diferencias entre estas ecuaciones son que la de convección-difusión tiene incógnita escalar y además el término convectivo no presenta no linealidad. Esto queda en evidencia en los tiempos de cálculo de la ecuación de convección-difusión y se puede usar saltos de tiempo mayores. Es por esto que los esfuerzos en este trabajo radicaron en resolver Navier-Stokes.
- El solver en MATLAB es más estable, se puede superar el valor $Co > 1$, y aún así el método no diverge. En OpenFOAM es de vital importancia cumplir esta condición CFL.
- El solver en MATLAB da flexibilidad, en el sentido que OpenFOAM es una especie de caja negra, ya que si bien se pueden hacer modificaciones (inspirándose en lo que ya está hecho) no se entiende bien la programación ni la notación de los códigos fuentes. Se pueden entender, pero es necesario ser experto en OpenFOAM y meterse de lleno en los códigos. No hay un tutorial que haga esto, sólo guía de usuario y tutorial que explican lo básico. Esto aplica sobre todo en las condiciones de borde, que en OpenFOAM no es fácil poner condiciones de borde no uniformes en espacio.
- Los resultados de la concentración son bastante similares, sin embargo, *MareaRoja-PimpleFoam* presenta problemas en los valores de la concentración. Además se puede apreciar los fenómenos de convección y difusión.
- El solver de Shallow Water si bien es bastante intuitivo para describir el problema (pues describe la altura y velocidad que es lo que interesa en el caso de la marea) no presenta los resultados esperados; no se aprecia el fenómeno de la marea. Una idea es modificar la ecuación, usando de antemano la altura $h(t)$ como dato, sin embargo habrían 3 ecuaciones (las dos de velocidad y la que se deriva de divergencia nula) y dos incógnitas (sólo la velocidad horizontal U_x, U_y), por lo tanto la EDP estaría sobredeterminada. Para resolver este problema, una opción es hacer que la gravedad dependa del tiempo al momento de derivar Shallow Water a partir de las ecuaciones de Navier-Stokes.

Como trabajo futuro queda implementar los modelos de turbulencia, en particular en modelo $\varepsilon - k$. Este modelo incorpora dos variables la disipación de energía turbulenta ε y la energía cinética turbulenta k, las ecuaciones son bastante similares a las Navier-Stokes con lo que no resulta difícil hacer las modificaciones al solver hecho en esta tesis. Sin embargo, está el problema de las *Wall Functions* que se usan en estos modelos, que son modelos y valores que hay que explorar.

Es importante mejorar la malla, pues hay zonas con altura muy pequeña (las cercanas a Mar-Tierra), en que hay una notable disminución de la profundidad del mar. Numéricamente esto genera problemas, dado que al momento de resolver el sistema se producen divisiones por números demasiado pequeños.

Los resultados obtenidos en MATLAB son buenos y permiten entender el problema, pero muy lentos, una opción es usar en C++ esta implementación. El cual disminuiría considerablemente el tiempo de ejecución, sin embargo el problema radica en la resolución del sistema lineal. Al momento de usar C++ es imperante usar UMFPACK para resolver los sistemas lineales de manera eficiente.

También se puede mejorar la velocidad en MATLAB haciendo *paralelismo* y haciendo uso del cluster. Ahí se pueden lanzar cálculos en un intervalo de tiempo mayor.

Por último, se requieren datos reales para que el modelo tenga validez, entre ellos la elevación de la marea, la velocidad normal en el borde mar-mar y la concentración inicial de algas.

Bibliografía

[1] Roberto Bagnara. A unified proof for the convergence of jacobi and gauss-seidel methods. *SIAM Review*, 37, 04 2001.

[2] C. Bègue, C. Conca, F. Murat, and O. Pironneau. A nouveau sur les équations de Stokes et de Navier-Stokes avec des conditions aux limites sur la pression. *Comptes Rendus de l'Académie des Sciences. Série I*, 01 1987.

[3] C. Conca, F. Murat, and O. Pironneau. The Stokes and Navier-Stokes equations with boundary conditions involving the pressure. 20(2):279–318, 1994.

[4] C. Conca, D. Nolte, G. Panasenko, K. Pileckas, and C. Bertoglio. Junction of models of different dimension for flows in tube structures by Womersley-type interface conditions. *SIAM Journal on Applied Mathematics*, 79:959–985, 06 2019.

[5] The OpenFOAM Foundation. *OpenFOAM v7 User Guide*.

[6] V. Girault and P. A. Raviart. *Finite Element Methods for Navier-Stokes Equations*. Springer Berlin Heidelberg, 1986.

[7] Christopher Paige and Michael Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 01 1975.

[8] Luc Tartar. *Topics in Non Linear Analysis*. Orsay : Université de Paris-Sud Département de mathématique, 1978.

[9] R. Temam. *Navier-Stokes Equations: Theory and Numerical Analysis*. American Mathematical Society, 2001.