Escuela Técnica Superior de Ingeniería

Universidad de Huelva

Máster Oficial en Ingeniería Industrial

Trabajo Fin de Máster

Simulación hidrodinámica y validación experimental de un vehículo submarino operado remotamente

Juan José Toscano Angulo

09/2021

DIRECTORES:

Dra. Inmaculada Pulido Calvo

Dr. Juan Carlos Gutiérrez Estrada

AGRADECIMIENTOS.

A mis padres y a todos los que creyeron en mí.

A los doctores de la Universidad de Huelva Inmaculada Pulido y Juan Carlos Gutiérrez, además de Samuel López, por vuestra ayuda en la realización de este trabajo.

A los creadores de contenido de los siguientes canales de YouTube. A ellos les debo gran parte de mi aprendizaje plasmado en este trabajo sobre simulación y herramientas CFD de OpenFOAM® y ParaView®.

AirShaper
The Efficient Engineer
József Nagy
Lynx
Lily Stewart
Hatari Labs
LibreMechanics

A todos, gracias.

RESUMEN

En los últimos años, los vehículos submarinos operados remotamente ROV están tomando gran relevancia en muchas aplicaciones industriales y científicas debido a su creciente desarrollo tecnológico. Para la correcta maniobrabilidad y optimización del rendimiento de estos vehículos se hace necesaria la compresión de sus propiedades hidrodinámicas de interacción con el medio acuático.

Los análisis por métodos numéricos permiten simular esta interacción flujo-ROV para facilitar la determinación de las reglas de regulación y control del mismo. Este procedimiento se presenta como una alternativa económicamente viable frente a métodos experimentales tradicionales de experiencias en canales de flujo. El presente trabajo plantea una metodología de simulación CFD (Computational Fluid Dynamics) y postprocesado mediante las herramientas open-source de OpenFOAM® y ParaView®, y de experimentación en canal de flujo para validar los resultados simulados. Tras comparar cualitativamente ambos resultados se comprueba la fiabilidad de la simulación, ofreciendo este método una estimación muy aceptable de la interacción del fluido con el ROV.

Palabras clave: ROV; métodos numéricos; simulación CFD; OpenFOAM®; ParaView®.

ABSTRACT

In recent years, remotely operated vehicles ROV are taking great relevance in many industrial and scientific applications due to their exponential technological development. In order to get the right maneuverability and performance optimization of these vehicles, it is necessary to understand their hydrodynamic characteristics of interaction with the aquatic environment.

Analysis made by applying numerical methods allow to simulate this flow-ROV interaction to facilitate the determination of its regulation and control rules. This procedure is presented as an economically viable alternative instead of traditional experimental methods of flow channels experiences. The present study proposes a CFD (Computational Fluid Dynamics) simulation and post-processing methodology made by using open-source CFD tools of OpenFOAM® and ParaView®, and an experimentation procedure in a flow channel to validate the simulated results. After qualitatively comparing both results, the reliability of the simulation is verified, resulting this method an acceptable way to estimate the fluid interaction with the ROV.

Keywords: ROV; numerical methods; CFD simulation; OpenFOAM®; ParaView®.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

Como es bien sabido, algo más del 70% de la superficie de la Tierra está cubierta por el agua de los mares y océanos. Estos son de vital importancia para la humanidad, ya que son los responsables de que la Tierra sea un planeta habitable. Es el ecosistema donde viven decenas de miles de especies subacuáticas y el gran regulador del clima atmosférico gracias a las corrientes marinas. De los océanos depende la regulación del oxígeno que respiramos y fenómenos meteorológicos fundamentales como las precipitaciones, tan importantes para la supervivencia de las especies animales y vegetales terrestres. En su aspecto socioeconómico, los océanos aportan riqueza y desarrollo, siendo esenciales para el turismo y comercio costero, y transporte de mercancías por todo el mundo. Sin olvidar que además representan fuentes críticas de alimentos y recursos energéticos renovables como lo son la energía eólica o mareomotriz, y no renovables como los son el petróleo y el gas natural. Por todas estas razones el estudio, conservación y sostenibilidad de los mares y océanos es esencial, siendo el Objetivo número14 de los ODS marcados por la ONU para 2030.

El conocimiento científico de los mares profundos está creciendo rápidamente conforme al desarrollo tecnológico. Las primeras exploraciones científicas subacuáticas se llevaron a cabo mediante vehículos submarinos tripulados, con las limitaciones de riesgo humano y costes que ello requería. Desde los años 70, los vehículos submarinos no tripulados han revolucionado la exploración del fondo marino, contando sistemas de navegación que ofrecen generalmente un control y adquisición de información de mejor calidad. Además, suponen un costo tremendamente más reducido que los sistemas convencionales, con el valor

añadido de evitar la exposición humana. Por otro lado, estos robots han permitido realizar operaciones de elevado riesgo en profundidades mayores, ejecutando misiones de inspección y manipulación de instalaciones subacuáticas entre un gran abanico de tareas de alto nivel.

Los vehículos submarinos no tripulados se diseñan siguiendo los principios fundamentales de la hidrodinámica. De esta forma, se puede estimar el comportamiento hidrodinámico del vehículo y experimentar con distintas condiciones de contorno para optimizar su rendimiento ante la exposición del vehículo a perturbaciones como oleaje y corrientes marinas. La problemática de aplicar métodos tradicionales de experimentación de laboratorio para casos complejos, es la inviabilidad técnica de su ejecución debido a los elevados costes que ello supondría.

En estos casos, los análisis por métodos numéricos de Dinámica de Fluidos Computacional (CFD, por sus siglas en inglés de *Computational Fluid Dynamics*) están tomando gran relevancia en el diseño y desarrollo de este tipo de vehículos. Esto es debido a que permiten obtener resultados con una precisión muy aceptable cuando el vehículo se encuentra en la fase de diseño, abaratando tremendamente los costes y tiempos de ejecución. De tal forma, conociendo las formas, los apéndices, y las condiciones de contorno, se puede optimizar y modificar la geometría sin necesidad de fabricar el modelo físico. También son muy útiles en casos de menor complejidad para complementar las tradicionales pruebas experimentales, aportando resultados adicionales.

La principal razón de realizar análisis hidrodinámicos es la obtención de los llamados coeficientes hidrodinámicos, ya que con estos se puede determinar la cinemática y dinámica del vehículo. Hasta el desarrollo de los métodos numéricos CFD, la única forma de obtener el valor de estos coeficientes es de forma empírica en aguas abiertas o en canales de experiencias hidrodinámicas. Estos métodos son caros ya que precisan de la construcción física del modelo. De tal forma, el uso de técnicas CFD se perfila como un método alternativo y adecuado para la estimación de estos coeficientes. Entre las herramientas informáticas CFD destaca Ansys Fluent®, en el cual se han realizado trabajos como Satria *et al.* (2014) y Ramírez-Macías *et al.* (2016) que constatan la fiabilidad de las mismas en obtener los coeficientes. También existen herramientas *open source* como OpenFOAM® que tiene la gran ventaja de ser un software fiable con todas las funcionalidades, aunque menos conocido, en el que se han realizado con éxito trabajos parecidos como Katsui *et al.* (2012), Li *et al.* (2020) y Bravo-Córdoba *et al.* (2021).

1.1. Antecedentes

El presente Trabajo de Fin de Máster ha sido realizado por Juan José Toscano Angulo, estudiante del Máster Oficial de Ingeniería Industrial, por la Universidad de Huelva. Se lleva a cabo bajo la supervisión de los doctores Inmaculada Pulido Calvo y Juan Carlos Gutiérrez Estrada, profesores titulares de la Universidad de Huelva de las áreas de conocimiento de Mecánica de Fluidos y Tecnologías del Medio Ambiente respectivamente, pertenecientes al Grupo de Investigación Análisis y Planificación del Medio Natural dentro del Departamento de Ciencias Agroforestales.

La realización de este trabajo se incluye en las actividades del Proyecto KTTSeaDrones 'Conocimiento y transferencia de tecnología sobre vehículos aéreos y acuáticos para el desarrollo transfronterizo de ciencias marinas y pesqueras (POCTEP 0622_KTTSEADRONES_5_E) cofinanciado por el Fondo Europeo de Desarrollo Regional FEDER a través del Programa Interreg V-A España-Portugal (POCTEP) 2014-2020. Los objetivos principales de este proyecto son (Gutiérrez Estrada *et al.*, 2019, 2020a, 2020b):

- Objetivo 1: Diseñar sensores adaptados a vehículos aéreos y marinos para el desarrollo del sector pesquero-acuícola y la gestión del litoral.

- Objetivo 2: Aumentar la competitividad y sostenibilidad en sectores económicos implicados en la gestión del litoral y del sector pesquero usando nuevas tecnologías.

- Objetivo 3: Crear nuevos nichos de empleo especializados, para ello tratan de introducir iniciativas innovadoras basadas en nuevas tecnologías.

1.2. Objetivos

En primer lugar analizar, evaluar y caracterizar las propiedades e interacciones del fluido alrededor de un robot submarino de tipo ROV mediante simulación CFD, principalmente en su movimiento de avance a velocidad constante. En segundo lugar diseñar y construir un modelo a escala del robot para la comparación y validación experimental de los resultados CFD en un canal de flujo. Los resultados a obtener serán la presión y fricción superficial, los coeficientes hidrodinámicos y fuerzas hidrodinámicas que actúan sobre el ROV, y las líneas de corriente del propio fluido.

1.3. Metodología y fases del proyecto

1. Modelado 3D del ROV mediante el programa de diseño SolidWorks®, a partir de las medidas tomadas del robot real adquirido con el presupuesto del Proyecto KTTSeaDrones.

2. Aprendizaje en materia de Dinámica de Fluidos Computacional aplicada en hidrodinámica, comandos básicos de Linux, y herramientas informáticas CFD de OpenFOAM® y ParaView®.

3. Configuración y simulación CFD del caso de estudio del ROV Sibiu Pro en OpenFOAM®.

4. Postprocesado de datos simulados y obtención de resultados en ParaView®.

5. Fabricación del ROV a escala mediante impresión 3D, a partir del Modelado 3D previamente realizado.

6. Experimentación con el ROV a escala en el canal de flujo del laboratorio de Mecánica de Fluidos, y obtención de resultados.

7. Análisis y validación de los resultados simulados y experimentales.

8. Redacción del trabajo

Capítulo 2
Estado del arte

En este Capítulo se presenta una revisión del estado del arte sobre distintos aspectos del campo de la robótica submarina. Se realiza una clasificación de los robots submarinos para centrarse posteriormente en los ROVs y en las aplicaciones de los mismos.

2.1. Clasificación de los Robots Submarinos

De acuerdo con Moreno *et al.* (2014), los robots submarinos se pueden clasificar según su nivel de autonomía, el tipo de misión a realizar y su sistema de propulsión. En la Figura 1 se presenta una clasificación de los mismos. La manera más común de clasificarlos es según su nivel de autonomía. Cabe diferenciar los robots submarinos autónomos AUVs (por sus siglas en inglés de *Autonomous Underwater Vehicle*) cuya autonomía es total, de los que deben ser monitorizados continuamente, llamados vehículos operados a remotamente ROVs (por sus siglas en inglés de *Remotely Operated Vehicle*). En el punto intermedio se encuentran los IAUVs, que en tareas de desplazamientos puede actuar de manera autónoma, mientras que necesitan de control supervisado en tareas de mayor complejidad. Sin embargo, la visión de futuro para estos robots es que lleguen a ser completamente autónomos, con la única intervención de un operador que defina la tarea a realizar mediante comandos de alto nivel.

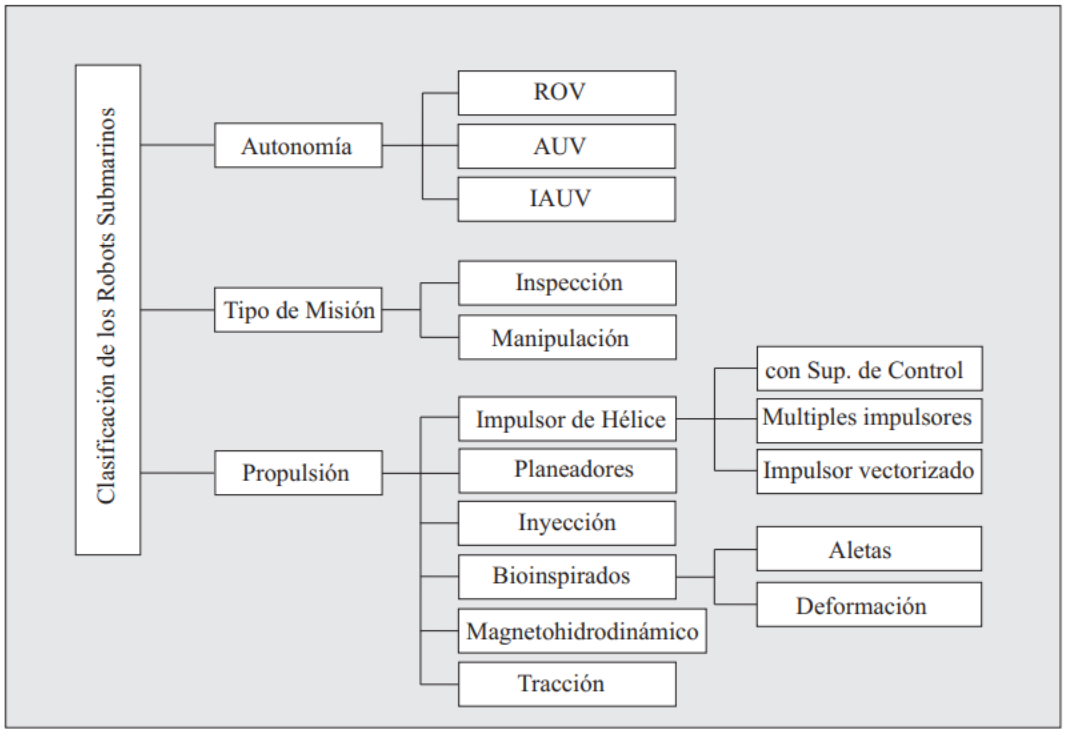


Figura 1. *Clasificación de los robots submarinos (Moreno et al., 2014).*

De igual manera, estos vehículos submarinos se pueden clasificar por el tipo de misión a realizar. Se diferencian dos tipos de misiones: de inspección (u observación) y de manipulación (o intervención). Los robots diseñados para realizar tareas de intervención cuentan con alguna herramienta o brazo robótico para realizarla, a diferencia de los que solo realizan inspecciones. En la mayoría de los casos, las misiones para la que está diseñado un robot submarino definen sus necesidades de componentes (sensores, actuadores, etc).

También existe la posibilidad de clasificar estos vehículos según sus propulsores, los cuales especifican la capacidad de movimientos y maniobras que pueden realizar. La elección de los sistemas de propulsión es un aspecto delicado, ya que repercute en el consumo de energía del mismo, el peso y volumen del robot, y el impacto ambiental que genera en los ecosistemas marinos. Los sistemas de propulsión típicos son los impulsores de hélice (el más común), los planeadores, y los bioinspirados. En fase experimental y en desarrollo están los basados en inyección de agua a presión y los magnetohidrodinámicos. Otro sistema de propulsión es la tracción, que puede ser con el fondo marino o con superficies artificiales.

2.2. Definición de ROV

Los vehículos operados remotamente ROVs son una clase de robot submarino que se controla a distancia por un operario. En la actualidad tienen su principal aplicación en el reconocimiento, inspección, observación y desarrollo de otras misiones específicas bajo el agua. Los ROVs se constituyen habitualmente por una cabina sobre la cual se instalan una serie de impulsores de hélice que permiten su desplazamiento y manejo a través del medio acuático. Según la misión para la cual están diseñados, contarán con un conjunto de sensores y actuadores para realizar dicha misión (Gutiérrez Estrada *et al.*, 2019).

La principal diferencia entre los ROVs y otros tipos de robots submarinos es que no son autónomos. De tal manera, necesitan conexión ininterrumpida por un conjunto de cables llamado cordón umbilical, por el cual se intercambian de datos y alimentación eléctrica con la superficie (normalmente una embarcación). A través de la interfaz gráfica de usuario del centro de control de la superficie, el operario define los comandos que el robot deberá ejecutar mediante sus actuadores. A su vez el ROV envía la información recopilada por sus sensores (presión, temperatura, imágenes, etc.) al centro de control, para que el usuario conozca el estado del vehículo y el ambiente que lo rodea.

Entre sus distintos tipos de actuadores, suelen contar con gran variedad de brazos robóticos según el tipo de misión a realizar, con cámaras de video que transmiten el entorno en tiempo real, o simples cámaras fotográficas para captar imágenes del fondo marino.

El cordón umbilical del ROV presenta ventajas e inconvenientes. La principal ventaja es la posibilidad de transmitir fácilmente alimentación y datos, con el inconveniente de la energía que se requiere para mover el peso del cordón. Otra desventaja se presenta cuando las tareas se realizan en grandes profundidades, ya que la fuerza de arrastre que se ejerce sobre la superficie del cordón se ve incrementada, sumándose a la generada por el propio ROV. Esto hace que el vehículo pierda manejabilidad. Aunque los cables han aumentado su diámetro por los mayores requerimientos de alimentación, el aumento de la superficie de incidencia de fuerza de arrastre depende en mayor medida de la extensa longitud de los mismos. Sin embargo, existen soluciones como implementar un Sistema de Manejo de Cable (TMS, por las siglas en inglés de *Tether Management System*) que se ancla en el fondo marino para soportar el arrastre del cordón en el tramo desde la embarcación al TMS, permitiendo que el ROV navegue con más facilidad (Moreno *et al.*, 2014).

2.3. Aplicaciones generales de los ROVs

En la actualidad, los ROVs son herramientas muy útiles y fiables para distintos ámbitos, debido a que su diseño permite en muchas ocasiones realizar misiones peligrosas sin necesidad de exposición humana. Se cuenta con una amplia gama de dimensiones, potencia, profundidad máxima y variedad de sensores y actuadores específicos para la misión de diseño. En este apartado se describen los campos más comunes de aplicación de los ROVs.

Acuicultura. Prevención e inspección de bioincrustaciones (cúmulo indeseable de microorganismos, plantas, algas y/o animales sobre estructuras mojadas), análisis de zonas estuáricas y explotaciones acuícolas, control y estudio de especies (tamaño, posición, etc), entre otras (Gutiérrez Estrada *et al.*, 2019).

Manipulación e Inspección. Para instalaciones petroleras o de gas, presas, tuberías, barcos, cables, cimentaciones etc. Las exigencias del trabajo realizado en infraestructuras submarinas son altas, ya que frecuentemente se requiere de inspección de grietas y defectos e intervención para realizar tareas tales como, mantenimiento, ensamblaje, limpieza, vigilancia, apertura y cierre de válvulas entre otras. La tendencia en el uso de ROVs se ve incrementada a medida que las infraestructuras cerca de la costa se trasladan hacia aguas más profundas, tales como instalaciones gas y petróleo (Moreno *et al.*, 2014).

Investigación científica. Realización de estudios de la fauna y flora submarina, recolección de muestras para estudios (arqueológicos, geológicos o ecológicos) y análisis físico-químico del agua (composición, pH, presión, temperatura etc). El ROV que se describirá en el presente trabajo tiene su campo de aplicación en la investigación de la población de fauna pesquera y muestreo de la calidad del agua.

Salvamento. Investigación policial, recuperación de cuerpos, misiones de rescate.

Construcción submarina. Excavaciones y enterrado de zanjas, tendido de tubos, cimentaciones, extensión de cables, perforaciones, etc.

Operaciones militares. Espionaje, detección y desactivación de minas, vigilancia etc.

Vigilancia de puertos, recuperación de objetos, búsqueda de tesoros etc.

Capítulo 3

Modelo hidrodinámico y CFD

En este Capítulo se presenta una descripción analítica del modelo dinámico de un ROV, y las principales fuerzas hidrodinámicas que actúan sobre éste, definiendo los coeficientes hidrodinámicos. Además se describen los fundamentos básicos de Dinámica de Fluidos Computacional que se aplican en la simulación CFD de ROVs en régimen turbulento.

3.1. Dinámica de ROVs

En navegación marítima y en robótica submarina se aplica la convención de la SNAME (por sus siglas en inglés de *Society of Naval Architects and Marine Engineers*) para nombrar los movimientos posición, velocidad y fuerzas de arrastre que se ejercen sobre el vehículo. En la Tabla 1 se presenta dicha nomenclatura en inglés y español, y su notación.

Tabla 1. *Convención SNAME de navegación marítima y robótica submarina.*

Movimiento	Español	Inglés	Posición	Velocidad	Fuerza
Translación en x	Avance	*Surge*	x	u	X
Translación en y	Desvío	*Sway*	y	v	Y
Translación en z	M. Vertical	*Heave*	z	w	Z
Rotación en x	Alabeo	*Roll*	ϕ	p	K
Rotación en y	Cabeceo	*Pitch*	θ	q	M
Rotación en z	Guiñada	*Yaw*	ψ	r	N

En la Figura 2 se puede visualizar la notación comentada en el esquema de un submarino. Los movimientos de rotación siguen la regla de la mano derecha.

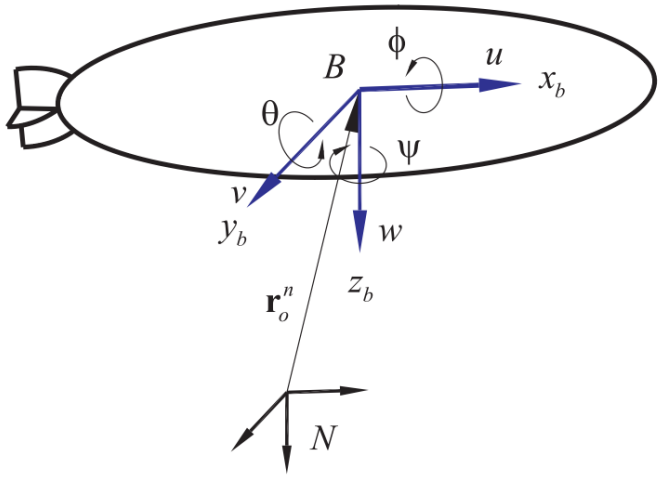


Figura 2. *Esquema de notación robótica submarina. Convención SNAME (Moreno et al., 2014).*

Siguiendo esta notación, el modelo dinámico de un ROV expresa la relación entre la cinemática del robot y las fuerzas de arrastre que se ejercen sobre el mismo. Esta expresión permite calcular la potencia de los impulsores necesaria para que el vehículo se mueva de una determinada forma, o de manera inversa permite calcular las perturbaciones generadas por el arrastre sometido.

El modelo dinámico se puede describir según la ecuación diferencial no lineal del movimiento de Newton-Euler para los 6 grados de libertad de un vehículo submarino. Su formulación vectorial fue desarrollada por Fossen (2011) inspirada en el modelo matemático de manipuladores. En su forma compacta se presenta como se muestra en las ecuaciones (1) y (2).

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau + g_o + w \quad (1)$$

$$\tau = B_t u_t \quad (2)$$

Donde M es la matriz de inercia y masa añadida, C es la matriz de fuerzas del cuerpo rígido y masa añadida, y D es la matriz de fuerzas viscosas. Ambas matrices son dependientes de la velocidad relativa ν entre el fluido y el submarino. El vector $g(\eta)$ representa las fuerzas hidrostáticas de restitución (fuerza de gravedad y la fuerza de flotación).

Por otro lado el vector τ representa las fuerzas que ejercen los propulsores (o cualquier otro elemento consumidor de potencia de movimiento) sobre el vehículo. Los elementos de la matriz de control B_t dependen de la configuración de componentes del submarino, sus superficies de control, el número de propulsores, y su ubicación y orientación. El vector u_t lo componen las fuerzas que generan cada propulsor. Por otra parte, los vectores g_o y w representan respectivamente la fuerza generada por el control de lastre y las perturbaciones del ambiente marino como olas, viento, corrientes marinas, etc (Moreno *et al.*, 2014).

Para un ROV open-frame (marco abierto), cómo es el ROV de este trabajo (descrito en el siguiente Capítulo), la posición del centro de gravedad proporciona una estabilidad estática adecuada que garantiza los movimientos pasivos de cabeceo y alabeo, y conduce a una pequeña amplitud de sus ángulos de giro. Por lo tanto, las componentes no lineales de las fuerzas y momentos pueden considerarse como las únicamente causadas por los efectos viscosos del flujo, que se vuelven menos importantes a medida que el ángulo de inclinación es más pequeño. Por lo tanto, el comportamiento hidrodinámico de avance (*surge*), desvío (*sway*), movimiento vertical (*heave*) y giñada(*yaw*) se tratan como los cuatro grados principales de libertad de ROVs open-frame, de los cuales se estudiarán las fuerzas aplicadas para los tres primeros movimientos, al ser los más comunes.

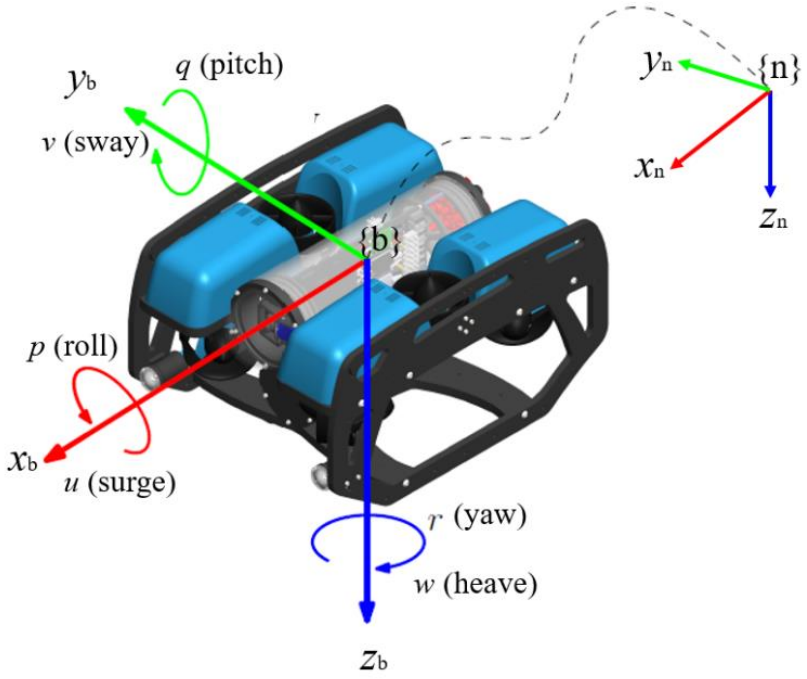


Figura 3. *Sistema de coordenadas del ROV open-frame (Li et al., 2020).*

3.2. Definición de fuerzas y coeficientes hidrodinámicos

Las fuerzas hidrodinámicas que someten a los robots submarinos son bastantes complejas ya que dependen de muchas variables como lo son las propiedades del fluido, la presión, la temperatura o la geometría del vehículo entre otras. Las fuerzas hidrodinámicas se dividen en las fuerzas de masa añadida y las fuerzas viscosas.

La masa añadida es una fuerza que se opone al movimiento del vehículo a través fluido y que depende de la aceleración del vehículo. Esta fuerza se suele asociar con la cantidad de fluido que se encuentra alrededor del robot submarino, y que se acelera cuando este también lo hace. El estudio realizado en este trabajo se realiza a velocidad relativa constante, por lo que se supone una influencia despreciable de fuerza de masa añadida.

Por otra parte, las fuerzas viscosas son el resultado de la fricción que se produce entre el fluido y el vehículo, debido a la viscosidad del fluido. Se diferencian según el efecto que producen en el vehículo como fuerza de arrastre y fuerza de sustentación. La ecuación (3) expresa la definición general de la fuerza hidrodinámica total viscosa.

$$F = \int_A (P - \tau_w)\, dA \tag{3}$$

Las componentes de esta fuerza se desarrollan en los siguientes apartados.

Tal como se ha comentado previamente, para obtener las fuerzas hidrodinámicas y sus coeficientes asociados, la forma más común aplicada históricamente es la experimentación. Sin embargo, debido al avance y uso de las herramientas CFD disponibles, se ha constatado que las estimaciones de coeficientes obtenidos por dichos métodos numéricos son bastante fiables.

3.2.1. Fuerza y coeficiente de arrastre

La fuerza de arrastre o resistencia, es la fuerza de rozamiento que se genera opuesta al movimiento relativo de un cuerpo a través de un fluido, debida a la fricción entre ambos. Esta fuerza depende de la presión y fricción superficial que el fluido ejerce sobre el cuerpo en la dirección del flujo, dependientes a su vez de la viscosidad del fluido, la velocidad relativa entre el cuerpo y el fluido, y de la geometría y superficie del cuerpo. La expresión analítica de la fuerza de arrastre se muestra en la ecuación (4).

$$F_D = \int_A (-P\cos\theta + \tau_w \sin\theta)\, dA \tag{4}$$

Donde P es la presión superficial (*surface pressure*), τ_w es la fricción superficial (*wall shear stress*), θ es el ángulo que forma la fuerza hidrodinámica total con la fuerza de arrastre, y A es la superficie del cuerpo.

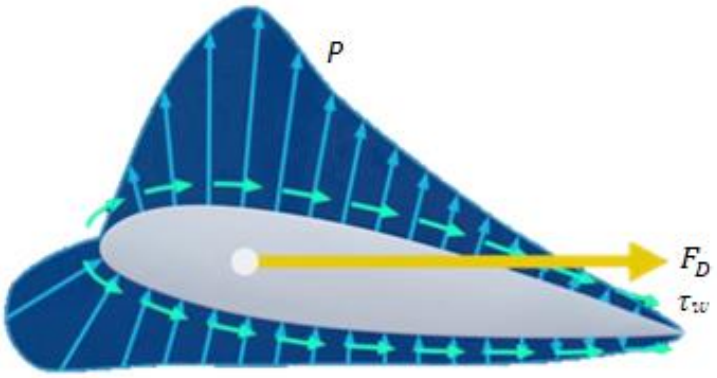


Figura 4. *Distribución de P y τw en un perfil de ala para F_D (The Efficient Engineer).*

La fuerza que se genera por presión surge de la diferencia de presión entre dos lados opuestos del cuerpo, siendo la suma de todas las fuerzas que empujan o tiran perpendicularmente a la superficie. Por otra parte, la fuerza de fricción surge del deslizamiento del fluido a través de la superficie del cuerpo, siendo la suma de todas las fuerzas de fricción paralelas a la superficie.

La fuerza total de arrastre sobre un vehículo repercute negativamente en el consumo de combustible y rendimiento de un vehículo, por lo que normalmente es indeseable (a no ser que se pretenda frenarlo). De tal forma, su reducción y minimización es de vital importancia para la industria aeronáutica y automovilística.

Sobre un perfil cilíndrico (no afilado) existe gran influencia de presión superficial debido a la gran región de separación de flujo que genera la geometría cilíndrica. Además existe poca influencia de fricción superficial ya que la separación de flujo reduce la superficie de contacto entre fluido y geometría.

Sobre un perfil de ala (afilado) pasa exactamente lo contrario. En este, existe gran influencia de fricción superficial debido a la mínima región de separación de flujo que genera la geometría de ala, maximizando así la superficie de contacto entre fluido y geometría. Por otro lado, existe poca influencia de presión superficial debido a la comentada mínima región de separación de flujo que se genera. Este diseño permite minimizar la fuerza de arrastre de las alas de los aviones.

En la Figura 5 se muestra la influencia de la presión y fricción superficial sobre ambos perfiles.

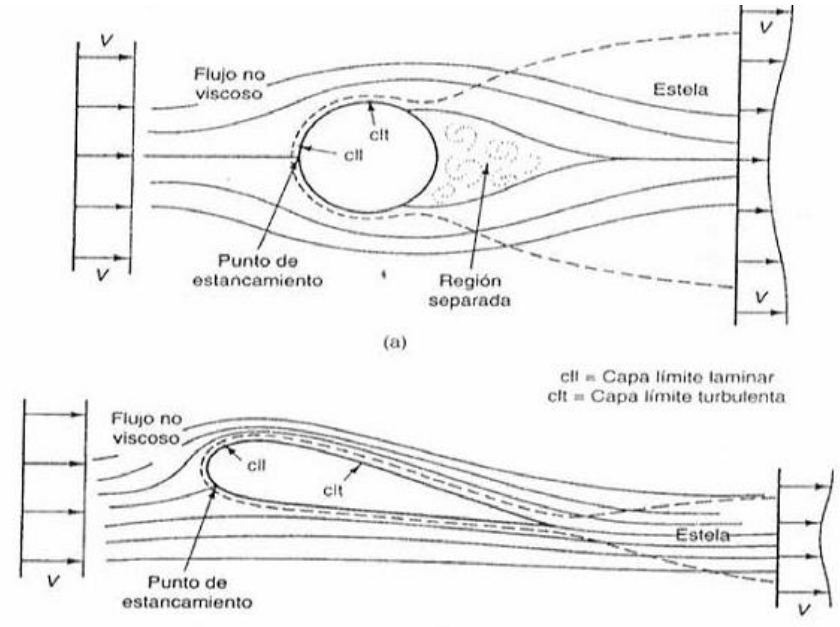


Figura 5. *Separación de las líneas de corriente sobre un perfil cilíndrico y de ala (https://sites.google.com/site/0902eliezerc/generador?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1).*

Las líneas de corriente permiten detectar zonas de flujo laminar y turbulentas. La transición de laminar a turbulenta a menudo ocurre cuando el flujo no puede seguir la superficie del objeto, porque el "ángulo negativo" de la superficie es demasiado grande o debido a una alteración geométrica en la superficie que desprende de vórtices.

Para el diseño de perfiles de ala en particular, se debe tener cuidado con reducir la presión superficial demasiado mediante el aumento de la relación geométrica L/D, ya que aumenta la superficie de contacto entre fluido y geometría, y de tal forma la fuerza de fricción que genera la fricción superficial o esfuerza cortante. Se debe hallar una solución de compromiso para L/D que minimice la fuerza de arrastre total.

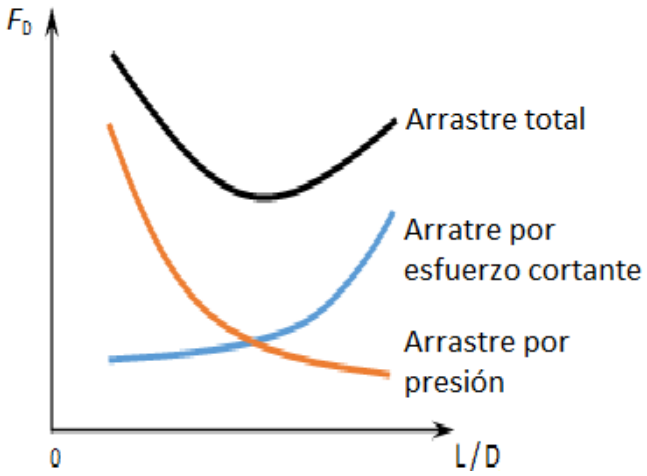


Figura 6. *Componentes de la fuerza de arrastre según L/D (Wikipedia).*

Por otra parte, el problema de la expresión analítica anterior de la fuerza de arrastre, es que P y τ_w son imposibles de obtener en la mayoría de los casos. De tal forma, en aerodinámica e hidrodinámica la fuerza de arrastre se calcula mediante la expresión de la ecuación (5).

$$F_D = \frac{1}{2}\rho\, C_D\, A_D\, U^2 \qquad (5)$$

Donde U es la velocidad del fluido (o relativa entre cuerpo y fluido) y C_D es el coeficiente hidrodinámico de arrastre y ρ es la densidad del fluido. Tanto la fuerza de arrastre F_D como el coeficiente de arrastre C_D se suelen designar mediante el subíndice D del inglés "*drag*" (arrastre). A_D es el área transversal del cuerpo que interactúa con el fluido en el plano ortogonal al sentido de flujo para geometrías no afiladas, o bien, el área longitudinal del cuerpo que interactúa con el fluido en el plano paralelo horizontal para geometrías afiladas.

El coeficiente de arrastre o resistencia, es un parámetro adimensional usado para cuantificar la resistencia de un cuerpo en el medio fluido. Este coeficiente depende en gran medida del número de Reynolds, y de la geometría y su superficie (Vélez Bermejo, 2020).

En la Figura 7 se muestra cómo se reduce C_D a medida que aumenta Re debido a la transferencia de momento que se produce por la mezcla de capas durante la transición a flujo turbulento, reduciéndose así la separación de flujo. Posteriormente C_D vuelve a aumentar debido al aumento de fricción superficial en función de Re.

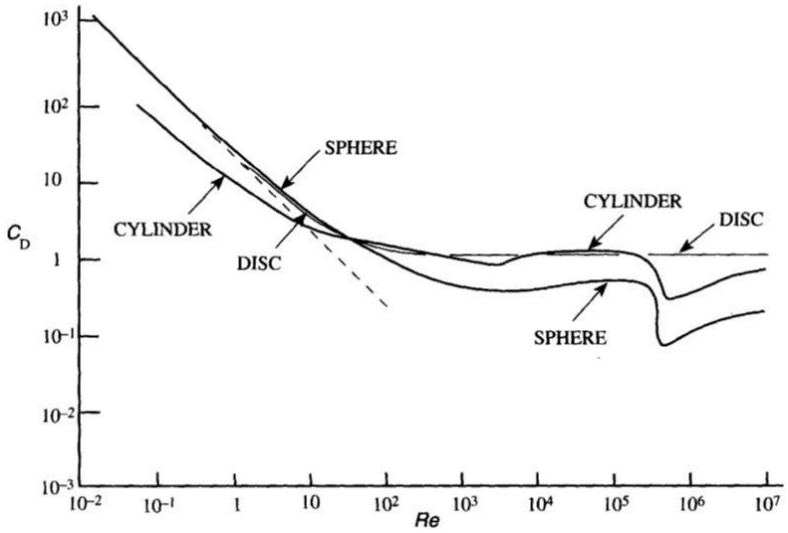


Figura 7. *C_D(Re) para diferentes formas geométricas en flujo de aire (The Efficient Engineer).*

La influencia de distintas formas geométricas sobre el coeficiente de arrastre para un número de Reynolds fijo, puede verse en la Figura 8.

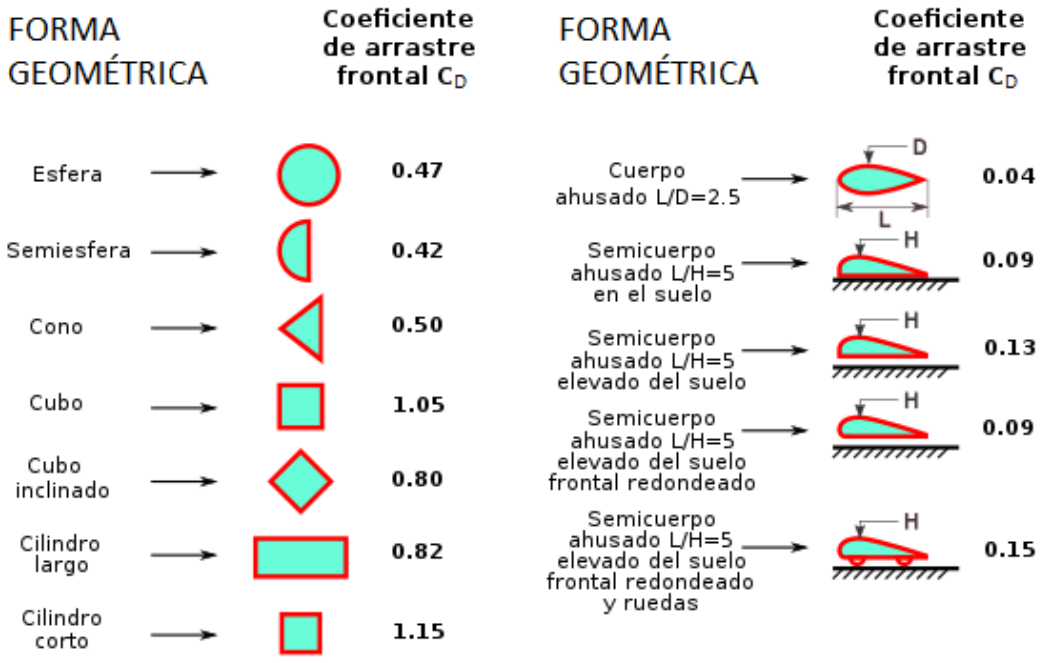


Figura 8. *C_D para diferentes formas geométricas en flujo de aire con Re=10.000 (Wikipedia).*

3.2.2. Fuerza y coeficiente de sustentación

La fuerza de sustentación es la fuerza de rozamiento que se genera perpendicular al movimiento relativo de un cuerpo a través de un fluido debido a la fricción entre ambos. Esta fuerza depende de la presión y fricción superficial que el fluido ejerce sobre el cuerpo en la dirección del flujo, dependientes a su vez de la viscosidad del fluido, la velocidad relativa entre el cuerpo y el fluido, y de la geometría y superficie del cuerpo. La expresión analítica de la fuerza de sustentación se muestra en la ecuación (6).

$$F_L = \int_A (-P \sin\theta - \tau_w \cos\theta)\, dA \tag{6}$$

Donde P es la presión superficial (*surface pressure*), τ_w es la fricción superficial (*wall shear stress*), θ es el ángulo que forma la fuerza hidrodinámica total con la fuerza de arrastre, y A es la superficie del cuerpo.

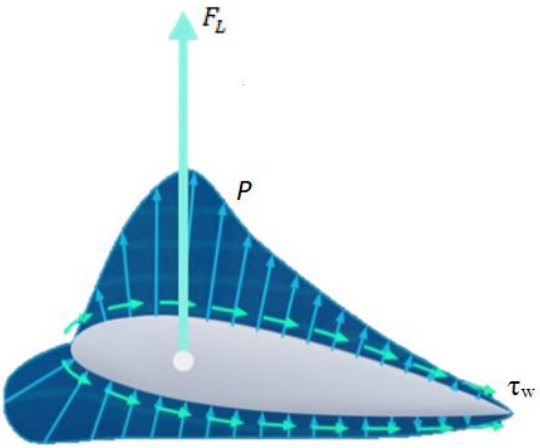


Figura 9. *Distribución de P y τw en un perfil de ala para F_L (The Efficient Engineer).*

La fuerza de sustentación es la principal responsable de que las aeronaves mantengan el vuelo gracias el perfil de presión superficial. La razón de ser de dicho perfil de presiones es un problema complejo del que hoy en día sigue sin tener consenso científico.

Una posible explicación a este fenómeno se puede extraer del Principio de Bernoulli, que siguiendo el principio de conservación de la energía, sugiere que las zonas de bajas presiones coinciden con una aceleración del fluido y viceversa. Otra posible explicación puede extraer de la Tercera Ley de Newton o ley de acción-reacción que se produce entre la zona superior de bajas presiones y la zona inferior de altas presiones.

En aerodinámica e hidrodinámica la fuerza de sustentación se calcula mediante la expresión de la ecuación (7).

$$F_L = \frac{1}{2}\rho\, C_L\, A_L\, U^2 \qquad (7)$$

Donde U es la velocidad del fluido (o relativa entre cuerpo y fluido), A_L es el área transversal del vehículo que interactúa con el fluido en el plano horizontal paralelo a la velocidad, C_L es el coeficiente hidrodinámico de sustentación y ρ es la densidad del fluido. Tanto la fuerza de sustentación F_L como el coeficiente de sustentación C_L se suelen designar mediante el subíndice L del inglés "*lift*" (sustentación).

El coeficiente de sustentación es un parámetro adimensional usado para cuantificar la sustentación de un cuerpo en un medio fluido. Este coeficiente no depende tanto del número de Reynolds, como del ángulo de ataque, la geometría y la superficie del cuerpo.

En la Figura 10 se muestra el coeficiente de sustentación con respecto al ángulo de ataque sobre un de perfil ala, que alcanza su máximo entorno a los 16º, momento en el que se vuelve a reducir debido al aumento del arrastre por separación de flujo. Por ejemplo en el momento de despegue de un avión, interesa un C_L máximo para facilitar su elevación.

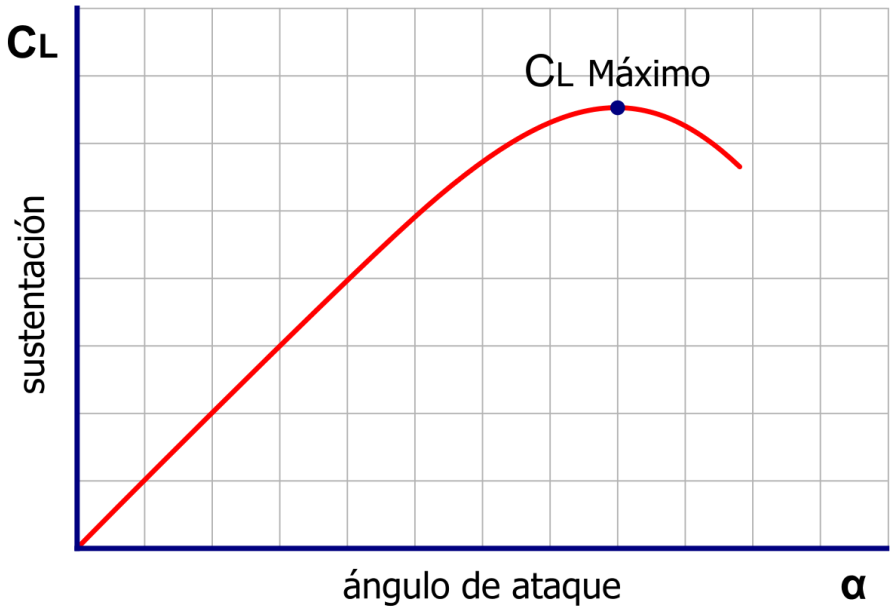


Figura 10. *C_L según el ángulo de ataque para un perfil de ala (Wikipedia).*

3.3. Dinámica de Fluidos Computacional

Conocer una estimación de los coeficientes hidrodinámicos (para un determinado número de Reynolds) de un vehículo submarino en su fase de diseño es de vital importancia, ya que de estos depende la maniobrabilidad del mismo, su consumo de energía y la potencia de los motores a instalar. Tal como se ha comentado, esta estimación se puede obtener mediante herramientas de simulación CFD.

La Dinámica de Fluidos Computacional (CFD) hace referencia a la utilización de métodos numéricos y algoritmos para resolución de problemas de mecánica de fluidos. En esta rama de la Física, se define el comportamiento de fluidos viscosos mediante las ecuaciones de Navier-Stokes, las cuales son un conjunto de ecuaciones en derivadas parciales no lineales. Los problemas que involucran estas ecuaciones son especialmente complejos, y resolubles mediante herramientas CFD, obteniendo ecuaciones simplificadas si se toman suposiciones que en muchos casos solo permiten alcanzar resultados aproximados.

El desarrollo de herramientas CFD permite la incorporación de software que reduce la velocidad de cálculo así como el margen de error, posibilitando además analizar problemas cada vez más complejos como flujos transónicos o turbulentos. La validación de los resultados obtenidos por CFD suele ser realizada de manera empírica en túneles de viento u otros modelos físicos a escala como tanques de experiencias hidrodinámicas (Anderson, 1995).

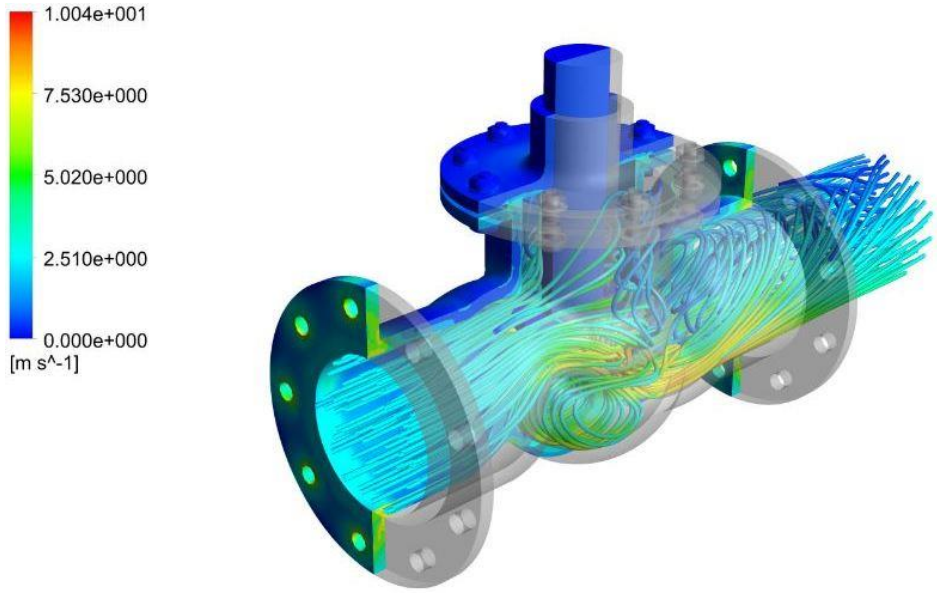


Figura 11. *Ejemplo de simulación CFD de una válvula de globo (Integral Innovation Experts).*

3.3.1. Aplicaciones de herramientas CFD

Se describen los siguientes campos de aplicación habituales de herramientas CFD:

Arrastre y diseño de propulsores: Las aplicaciones de herramientas CFD se centran fundamentalmente en ese campo. El desarrollo exponencial de las aplicaciones del mismo ocurrió a finales de los 90, cuando empezó a considerarse de la viscosidad y formación de olas, complicando así los cálculos y asemejándose al modelo real. De esta forma se obtienen cálculos más precisos del rendimiento del propulsor para una mejor aproximación a los resultados empíricos. En este se aplican las técnicas BEM (por sus siglas en inglés de *Boundary Element Methods)* para obtener soluciones aproximadas de las ecuaciones en derivadas parciales (Garrido Pellicer, S.F.).

Maniobrabilidad: Esta capacidad de las embarcaciones se valora cada vez más debido a las nuevas regulaciones de la IMO (por sus siglas en inglés *International Maritime Organization*) con el fin de evitar colisiones o accidentes marítimos de las embarcaciones. Las herramientas CFD permiten analizar flujos alrededor de apéndices del casco, calcular momentos generados y evaluar la maniobrabilidad del conjunto del buque.

Comportamiento marítimo: Campo de aplicación poco maduro pero en auge, donde cada día más existen más códigos comerciales que permiten simular y calcular los movimientos y cargas sobre el buque frente a condiciones externas. La problemática está en la complejidad computacional que supone definir modelos matemáticos de estados marinos y condiciones atmosféricas. Estos además requieren de mallados dinámicos y de diferente tamaño (por ejemplo para el modelo del oleaje), con una calidad del mismo aceptable.

Flujos en el interior de tuberías, conductos y válvulas. Hace referencia al campo de aplicación que se centra en el diseño redes de distribución a partir de resultados simulados de presiones y caudales que someten conductos, así como la perdida de carga que se produce. También permite realizar estudios de cavitación.

Tal como se ha venido comentando, el campo de aplicación de las herramientas CFD del presente trabajo se centra en el cálculo y análisis de la resistencia al avance y sustentación de un ROV en medios marítimos, y en la obtención de los coeficientes hidrodinámicos que determinan el comportamiento y maniobrabilidad del vehículo.

3.3.2. Ventajas e inconvenientes de las herramientas CFD

Algunas de las ventajas del uso de herramientas CFD son las siguientes:

- Reducción importante de tiempo y costes de I+D.
- Posibilidad de analizar y resolver casos muy difíciles o inviables por métodos empíricos.
- Capacidad de simular casos fuera de las condiciones límites de funcionamiento.
- Resolución de problemas en intervalos de tiempo ilimitados. Los métodos experimentales son más caros y lentos cuantos más puntos de medida se tomen, mientras que los métodos CFD solo requieren de capacidad de procesado y almacenamiento de datos.
- Valor añadido del producto. Posibilidad de generación de gráficos postprocesados que permiten una mejor comprensión del resultado y fomentar el marketing del producto.

Entre las principales desventajas se pueden citar:

- Se necesita de personal experimentado en CFD con conocimiento de las ecuaciones que modelan ciertos fenómenos físicos.
- En determinados casos es imposible obtener resultados lo suficientemente precisos, dando lugar a grandes errores.
- En determinados casos se necesita simplificar el fenómeno a estudiar para que el software del PC en cuestión pueda ejecutarlo.
- En determinados casos aún no existen modelos para simular efectos complejos de la turbulencia, fenómenos multifásicos o la combustión, entre otros.
- Tendencia humana a creerse los resultados del ordenador sin aplicar criterio alguno.

En definitiva, en función de distintos factores, el uso de herramientas CFD es un método de análisis tan válido como lo es experimentación empírica, y según el criterio del ingeniero/investigador se debe decidir qué método aplicar (si uno o ambos) dependiendo de las características del caso (Garrido Pellicer, S.F.).

3.3.3. Metodología CFD

La metodología consiste en discretizar una región del espacio, con las superficies volumétricas que en ella se incluyan, creando lo que se conoce como mallado, dividiendo la región en pequeños volúmenes de control llamadas celdas. La simulación resuelve en cada centro de cada celda el sistema de ecuaciones que define el modelo físico-matemático del problema en cuestión, y de manera discretizada. Esto consiste en resolver una matriz algebraica, cuyos elementos son las celdas, de forma iterativa hasta que el error sea suficientemente pequeño. En la actualidad se cuenta con diversos métodos, códigos y softwares CFD para resolver sistemas de ecuaciones, pero todos ellos siguen el mismo procedimiento básico (Torres Parish, 2017).

Preprocesado

Se diseña la geometría del problema mediante algún software CAD (por sus siglas en inglés de *Computer Aided Design*) con su posterior exportación o mediante la propia herramienta CFD si cuenta con algún módulo de diseño. Según esta geometría, se especifica el mallado, que puede ser uniforme o no uniforme, estructurado o no estructurado, y consistente en una combinación de celdas que pueden ser hexaédricas, tetraédricas, prismáticas, piramidales o poliédricas.

Se establece el modelo físico-matemático que define el sistema de ecuaciones a resolver, el algoritmo de resolución, las variables a calcular y el alcance de estos cálculos.

Se especifican las condiciones de frontera, o lo que es lo mismo, definir el comportamiento y propiedades de la interacción del fluido con las regiones y superficies de la geometría. Además se deben establecer las condiciones o valores iniciales de cada variable en cada región y superficie del mallado, para que los cálculos tengan un punto de partida inicial.

Simulación

Se ejecuta la simulación y se resuelven las ecuaciones de forma iterativa.

Postprocesado

Se postprocesan los resultados mediante módulos de cálculo, análisis y visualización de datos para proceder a su interpretación. Muchos de estos módulos permiten capturar imágenes y animaciones que muestran la evolución de los resultados.

3.3.4. Métodos de discretización

Método de Volúmenes Finitos

El método de volúmenes finitos FVM (por sus siglas en inglés de *Finite Volume Method*) es el método más utilizado en herramientas CFD por sus ventajas de optimización de memoria y velocidad de análisis, especialmente en problemas que requieren de grandes volúmenes de datos. En este método, el modelo físico-matemático se reescribe en forma conservativa y se resuelve en volúmenes de control discretos, garantizando la conservación de los flujos a través de las celdas. La siguiente expresión define la ecuación de volumen finito discretizada.

$$\frac{\partial}{\partial t}\iiint Q\, dV + \iint F\, dA = 0 \tag{8}$$

Donde Q es el vector de conservación de variables, F es el vector de flujos, V es el volumen de la celda, y A es el área superficial de la celda (Torres Parish, 2017).

Método de Elementos Finitos

El método de los elementos finitos FEM (por sus siglas en inglés de *Finite Element Method*) es conocido por su uso en análisis estructural, y aplicable en herramientas CFD. Este método es más estable que el FVM, pero requiere mucha más memoria y tiempo de cálculo (Torres Parish, 2017). La ecuación ponderada de error posee la siguiente forma.

$$R_i = \iiint W_i\, Q\, dV^e \tag{9}$$

Donde R_i es el error ponderado del vértice i de un elemento, Q es el vector de conservación de variables, W_i es el factor de ponderación, y V^e es el volumen del elemento.

Método de Diferencias Finitas

El método de diferencias finitas FDM (por sus siglas en inglés de *Finite Difference Method*) es una técnica numérica que se basa en la resolución de las ecuaciones diferenciales aproximando las derivadas como diferencias finitas. Actualmente se usa solo en aplicaciones muy concretas por las elevadas necesidades de CPU y memoria que requieren (Torres Parish, 2017). La siguiente expresión define la ecuación de diferencias finitas.

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} = 0 \tag{10}$$

Donde Q es el vector de conservación de variables, y F, G y H son respectivamente los flujos en las direcciones $x\ y\ z$ en sentido positivo.

3.3.5. Turbulencia y técnicas de simulación CFD

Tal y como fue mencionado previamente, la resolución de las ecuaciones de Navier-Stokes es un problema complejo. Uno de los principales inconvenientes se presenta para flujos turbulentos, ya que tal y como se ha demostrado de manera empírica, se trata de un sistema caótico. La teoría del caos es una rama de la ciencia que estudia ciertos tipos de sistemas complejos y sistemas dinámicos muy sensibles pequeñas variaciones en las condiciones iniciales, suponiendo grandes diferencias de comportamiento futuro, imposibilitando su predicción. Para estudiar estos sistemas se deben considerar cantidades medias y definir el modelo de turbulencia como un sistema caótico determinista, y así obtener precisiones de cálculo aceptables (Torres Parish, 2017).

La turbulencia se puede definir como un estado de movimiento del fluido caracterizado por perturbaciones caóticas que incluyen fluctuaciones o variaciones bruscas de presión, velocidad y dimensiones, dependientes tanto de la posición como del tiempo. Durante la separación de flujo entre el fluido y el cuerpo, se generan los llamados vórtices, que pueden ser de gran tamaño incluso en flujos laminares. Conforme la velocidad se incrementa, y se desarrolle completamente el flujo turbulento, estos vórtices tienden a contraerse a lo largo de la dirección perpendicular al flujo, y alargarse en la dirección del flujo para conservar el momento angular. Este alargamiento es el responsable de una ampliación local de la intensidad en la vorticidad, al mismo tiempo que forma pequeñas estructuras en el fluido (Garrido Pellicer, S.F.).

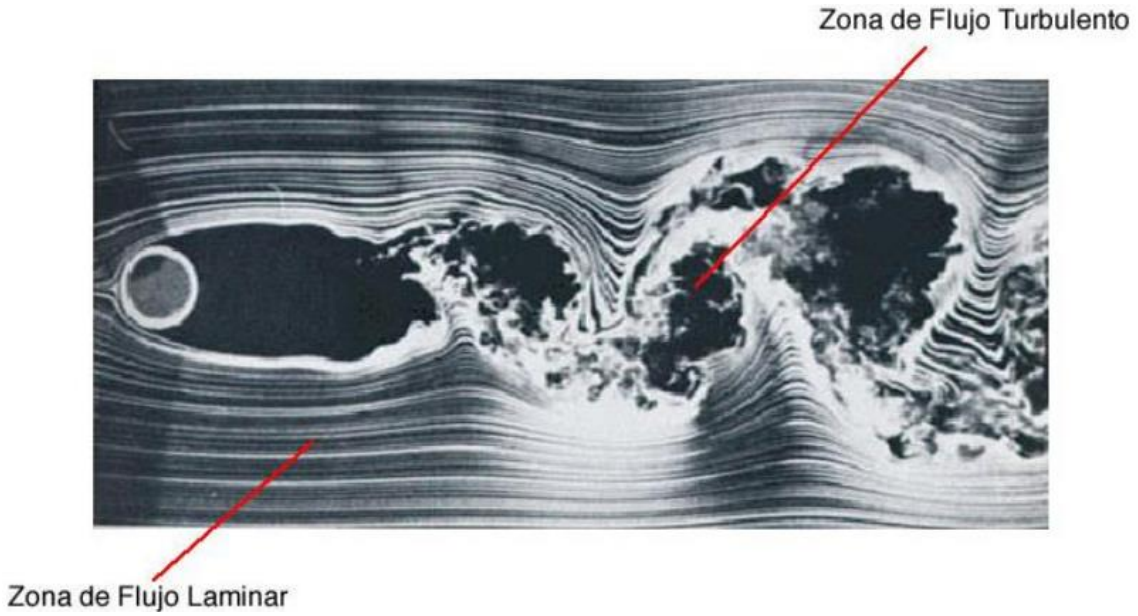


Figura 12. *Flujo externo a un cilindro. Generación de vórtices (TextosCientificos.com).*

La gran mayoría de los flujos que se estudian en las ramas de ingeniería son turbulentos, ya que tienden a dominar el comportamiento del flujo. La turbulencia influye en la determinación de muchos parámetros importantes, como la fuerza de arrastre, la transferencia de calor, el desprendimiento de la capa límite, etc. La presencia de turbulencia generalmente domina sobre el resto de fenómenos de flujo, incrementándose así la disipación, la transferencia de materia y la transferencia de calor.

La complejidad de los fenómenos turbulentos y sus aplicaciones hace que entender cómo funcionan, su magnitud, predicción, simulación y control se haya convertido en uno de los problemas más dificultosos e importantes de la ciencia e ingeniería. La simulación de turbulencias constituye una de los principales obstáculos de las herramientas CFD, ya que no permiten obtener soluciones exactas desde el punto de vista de los métodos numéricos. Existen 3 técnicas CFD para simular flujos turbulentos que se diferencian en el cálculo y suposiciones que se aplican sobre los vórtices.

Por una parte está la técnica de simulación DNS (por sus siglas en inglés de *Direct Numerical Simulation*) que resuelve las ecuaciones de Navier-Stokes en toda la gama de escalas espaciales y temporales de vórtices, incluso en los más pequeños (*small eddies*), y sin aplicar modelos de turbulencia. El problema de esta técnica es que hoy en día solo sería aplicable para un número muy limitado de casos muy simples y de interés puramente académico.

Por otra parte está la técnica de simulación LES (por sus siglas en inglés de *Large Eddy Simulation*) que resuelve hasta la gama de vórtices largos (*large eddies*), aplicando modelos de turbulencia a los más pequeños. Se trata entonces de una técnica intermedia entre la simulación directa de las ecuaciones de Navier-Stokes y los modelos de turbulencia RANS. Aunque sin llegar al extremo de la técnica DNS, su alcance de aplicaciones solo llega hasta problemas simplificados, ya que se requiere de unas capacidades de cálculo muy elevadas.

Por último esta existe la técnica de modelos de turbulencia RANS (por sus siglas en inglés de *Reynolds Averaged Navier-Stokes equations*) que no resuelve ningún vórtice explícitamente, pero si su efecto usando del concepto de viscosidad turbulenta. Esta técnica es la más utilizada históricamente debido a su versatilidad y exactitud de cálculos, definidos a partir del promediado por Reynolds de las ecuaciones de Navier-Stokes, que le dan su nombre.

En las Figuras 13 y 14 se muestra como la técnica utilizada influye sobre la exactitud de los cálculos, el coste computacional y la complejidad de la geometría. Cuanto más directa sea la simulación más exactos serán los cálculos, con las limitaciones de alto coste computacional para únicamente geometrías simples de casos simples. En cambio, cuanto más se modele el comportamiento de la turbulencia, menos exactos serán los cálculos, suponiendo un menor coste computacional para una mayor gama de complejidad geométrica y de casos.

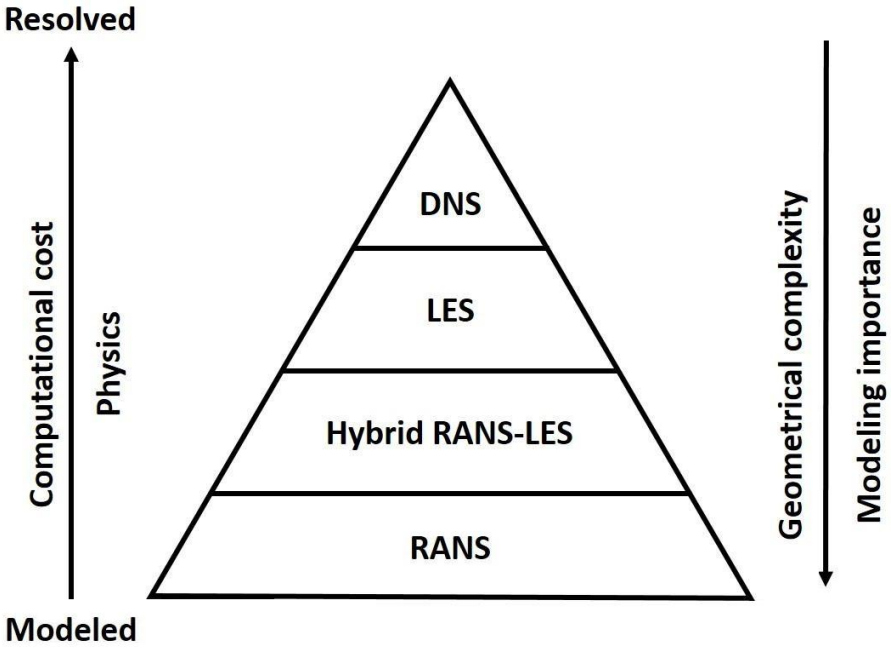


Figura 13. *Pirámide de características de DNS, LES y RANS (analyticsindiamag).*

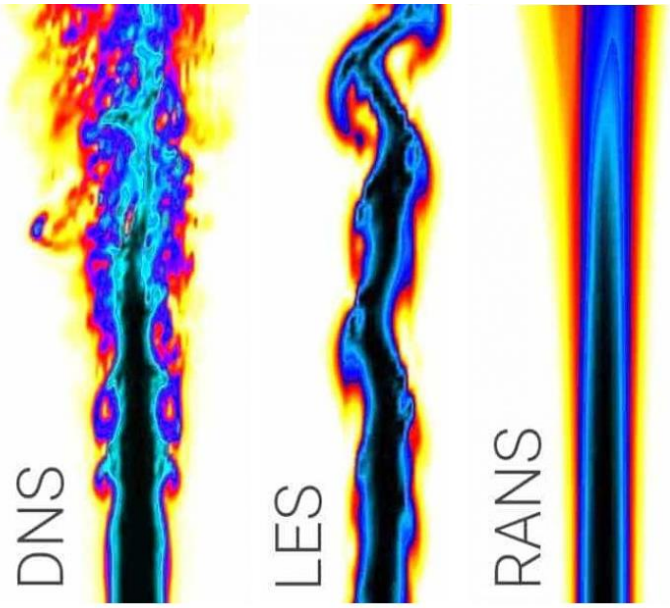
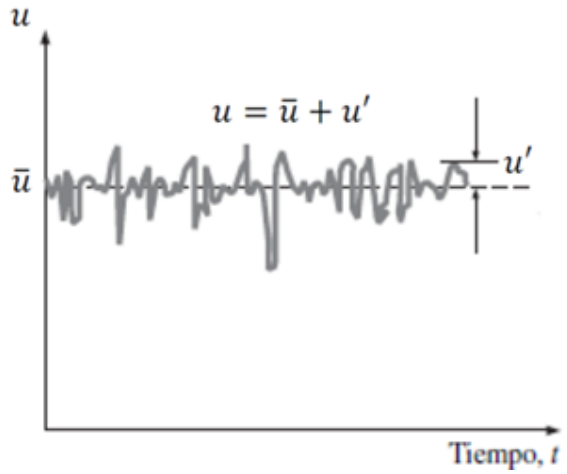


Figura 14. *Simulación de flujo DNS, LES Y RANS (idealsimulations).*

3.3.6. Modelos de turbulencia RANS

Los modelos de turbulencia RANS han sido muy estudiados y resultan de bastante utilidad en la mayoría de los problemas prácticos resueltos mediante técnicas numéricas. El procedimiento se basa en aplicar las leyes del promedio de Reynolds sobre las ecuaciones de Navier-Stokes, con el fin de obtener los comportamientos turbulentos promediados de las distintas variables.



$$\boxed{f = \overline{f} + f' \qquad g = \overline{g} + g}$$

$$(1): \overline{f'} = 0 \qquad (6): \overline{\overline{f}\overline{g}} = \overline{f}\overline{g}$$

$$(2): \overline{\overline{f}} = \overline{f} \qquad (7): \overline{fg} = \overline{f}\overline{g} + \overline{f'g'}$$

$$(3): \overline{f\overline{g}} = \overline{f}\overline{g} \qquad (8): \overline{\frac{\partial f}{\partial s}} = \frac{\partial \overline{f}}{\partial s}$$

$$(4): \overline{f'g} = 0$$

$$(5): \overline{f+g} = \overline{f} + \overline{g} \qquad (9): \overline{\int f ds} = \int \overline{f} ds$$

Figura 15. *Promedio de velocidad turbulenta. Leyes del Promedio (Lynx).*

De tal forma, se trata de obtener una descomposición del valor medio y valor fluctuante de las variables. El resultado es un campo de flujo turbulento promediado y simulado más uniforme que el flujo turbulento real, reduciéndose así drásticamente el número de puntos tomados para la discretización espacial y temporal, necesaria para resolver las variables buscadas. De este modo se obtienen unas soluciones aceptables aunque no exactas (Garrido Pellicer, S.F.).

Así, RANS parte de las ecuaciones de Navier-Stokes clásicas de continuidad y cantidad de movimiento para fluidos newtonianos, incompresibles y viscosos.

$$\frac{D}{Dt}\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \frac{\partial}{\partial t}\begin{pmatrix} u \\ v \\ w \end{pmatrix} + (u \cdot \nabla)\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \frac{-\nabla P}{\rho} + g + \nu \nabla^2 \begin{pmatrix} \bar{u} \\ \bar{v} \\ \bar{w} \end{pmatrix} \qquad (11)$$

Se aplican las leyes del promedio de Reynolds y se deriva cada componente, obtiéndose las ecuaciones RANS, donde queda añadido el término final de la divergencia del tensor de tensiones de Reynolds.

$$\frac{D\bar{u}}{Dt} = \frac{\partial}{\partial t}\begin{pmatrix}\bar{u}\\ \bar{v}\\ \bar{w}\end{pmatrix} + (\bar{u}\cdot\nabla)\begin{pmatrix}\bar{u}\\ \bar{v}\\ \bar{w}\end{pmatrix} = \\ = \frac{-\nabla\bar{P}}{\rho} + \bar{g} + \nu\nabla^2\begin{pmatrix}\bar{u}\\ \bar{v}\\ \bar{w}\end{pmatrix} + \nabla\cdot\begin{pmatrix}[-\overline{u'u'} & -\overline{u'v'} & -\overline{u'w'}\,]\\ [-\overline{v'u'} & -\overline{v'v'} & -\overline{v'w'}\,]\\ [-\overline{w'u'} & -\overline{w'v'} & -\overline{w'w'}\,]\end{pmatrix} \quad (12)$$

Si se expresan las ecuaciones RANS en notación tensorial y se asume el flujo en 2D por simplificación, se puede obtener la expresión de la ecuación (13).

$$\rho\left(\frac{\partial\bar{u}_i}{\partial t} + \bar{u}_j\frac{\partial\bar{u}_i}{\partial x_j}\right) = -\frac{\partial\bar{P}}{\partial x_i} + \rho\bar{g}_i + \mu\frac{\partial^2\bar{u}_i}{\partial {x_j}^2} - \frac{\partial}{\partial x_j}\rho\overline{u'_i u'_j} \quad (13)$$

De tal forma, las ecuaciones RANS quedan con un término adicional en función de la componente desconocida $-\overline{u'_i u'_j}$ del tensor de tensiones de Reynolds (simétrico de 6 componentes asumiendo el flujo en 2D).

Analíticamente, el objetivo de los modelos de turbulencia RANS es relacionar las componentes desconocidas del tensor de tensiones de Reynolds con los términos de las ecuaciones RANS en su forma tensorial, para así poder calcular las variables de presión y velocidad.

Una línea del modelado conocida como "modelos de viscosidad parásita" (*eddy viscosity models*) consiste en relacionar las componentes del tensor de tensiones de Reynolds con la velocidad media del tensor de deformación a través del factor llamado viscosidad turbulenta. Esto se conoce como la "aproximación de Boussinesq" para la turbulencia, y se expresa según la ecuación (14).

$$-\overline{u'_i u'_j} = \nu_t\left(\frac{\partial\bar{u}_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}k\delta_{ij} \quad (14)$$

Donde δ_{ij} es la función delta Kronecker que vale 1 para $i = j$ y vale 0 para $i \neq j$. k es la energía cinética turbulenta y ν_t es la viscosidad turbulenta. Como cierre de estas dos últimas variables, se debe hacer la elección del modelo de turbulencia RANS a emplear.

3.3.6.1. Modelo de turbulencia k-épsilon

El modelo de turbulencia RANS k-épsilon (k-ε) es el más utilizado en CFD para simular las características de flujo medio en condiciones de turbulencia mediante el cierre de las ecuaciones RANS. A diferencia de otros modelos de turbulencia, el modelo k-ε se centra en los mecanismos que afectan la energía cinética turbulenta, siendo útil para flujos cerca de la capa límite con gradientes de presión relativamente pequeños. La hipótesis del mismo es que la viscosidad turbulenta es isotrópica, en otras palabras, la relación entre el esfuerzo de Reynolds y la tasa media de deformaciones es la misma en todas las direcciones. El problema de este modelo es que no funcionan bien en casos de grandes gradientes de presión adversos (Wilcox, 1998).

- La 1ª ecuación (15) del modelo k-ε es la ecuación de la mencionada de energía cinética turbulenta k (m^2/s^2).

$$\frac{D}{Dt}(\rho k) = \nabla \cdot (\rho D_k \nabla k) + P - \rho\varepsilon \tag{15}$$

Donde D_k es la difusividad efectiva de k, P(m^2/s^3) es la tasa de producción de energía cinética turbulenta y ε (m^2/s^3) es la tasa de disipación de energía cinética turbulenta.

- La 2ª ecuación (16) del modelo k-ε es la ecuación de la tasa de disipación de energía cinética turbulenta ε (m^2/s^3).

$$\frac{D}{Dt}(\rho\varepsilon) = \nabla \cdot (\rho D_\varepsilon \nabla\varepsilon) + \frac{C_1\varepsilon}{k}\left(P + C_1\frac{2}{3}k\nabla \cdot u\right) - C_2\,\rho\frac{\varepsilon^2}{k} \tag{16}$$

Donde D_ε es la difusividad efectiva de ε, y C_1 y C_2 son los coeficientes del modelo, de valores 1,44 y 1,92 respectivamente.

La viscosidad turbulenta ν_t(m^2/s) se define en función del modelo de turbulencia empleado. Para el caso del modelo de turbulencia k-ε se define según la ecuación (17).

$$\nu_t = C_\mu \frac{k^2}{\varepsilon} \tag{17}$$

Donde C_μ es una constante empírica de valor *0,09* k y ε son las variables mencionadas del modelo de turbulencia k-ε.

3.3.6.2. Modelo de turbulencia k-omegaSST

El modelo de turbulencia RANS k-omegaSST (k-ωSST) es una combinación entre las mejores características de los modelos de turbulencia comunes RANS k-épsilon y k-omega, mediante formulación de transporte de esfuerzo cortante (SST). Esto proporciona un rendimiento simular al de k-ω, con un comportamiento de corriente libre similar al de k-ε, evitándose el problema común de k-ω por su alta sensibilidad a las propiedades de turbulencia de la corriente libre de entrada. Destaca así por su buen comportamiento en gradientes de presión adversos y flujo de separación (Menter, 1993).

- La 1ª ecuación (18) del modelo k-ωSST es la ecuación de la mencionada de energía cinética turbulenta k (m^2/s^2).

$$\frac{D}{Dt}(\rho k) = \nabla \cdot (\rho D_k \nabla k) + \rho G - \frac{2}{3}\rho k(\nabla \cdot u) - \rho \beta^* \omega k + S_k \tag{18}$$

- La 2ª ecuación (19) del modelo k-ωSST es la ecuación de la tasa de disipación específica turbulenta ω (1/s).

$$\frac{D}{Dt}(\rho \omega) = \\ = \nabla \cdot (\rho D_\omega \nabla \omega) + \frac{\rho \gamma G}{\nu} - \frac{2}{3}\rho \gamma \omega (\nabla \cdot u) - \rho \beta^* \omega^2 - \rho (F_1 - 1) C D_{k\omega} + S_k \tag{19}$$

Por otra parte, tal como se ha comentado previamente, la viscosidad turbulenta ν_t (m^2/s) se define en función del modelo de turbulencia empleado. Su valor es dependiente de las variables mencionadas k y ω del modelo de turbulencia k-ωSST.

Capítulo 4
Material y métodos

En este Capítulo se detallan las herramientas utilizadas y el procedimiento seguido para el análisis hidrodinámico del ROV Sibiu Pro, por una parte la simulación CFD y por otra parte de manera experimental en el laboratorio de Mecánica de Fluidos.

De tal forma, primero se realiza una descripción técnica de Sibiu Pro, así como el procedimiento para el modelado 3D del mismo. Seguidamente se explica la configuración del caso de OpenFOAM® SibiuPro al que se le aplican comandos de preprocesado y simulación. En el apartado de postprocesado en ParaView® se detalla cómo se obtienen las fuerzas hidrodinámicas sobre el ROV, y el cálculo sus coeficientes hidrodinámicos asociados.

El apartado final del Capítulo detalla el material y metodología seguida en el laboratorio la generación de líneas de flujo de interacción con el ROV. En este se explica además el procedimiento de diseño y fabricación mediante impresión 3D del colector de flujo y el ROV a escala.

Para seguir adecuadamente este Capítulo se entiende que el lector cuenta con conocimientos de las herramientas de simulación CFD OpenFOAM® y postprocesado datos ParaView®. Si el lector no cuenta con dichos conocimientos será necesario que se lea los anexos de este trabajo escritos para tal propósito (ver Anexos V y VI).

4.1. Descripción del ROV

Tal y como se ha nombrado, en este estudio se utiliza el ROV disponible comercialmente Sibiu Pro (*Nido Robotics, Murcia, Murcia, España*), adquirido por la Universidad de Huelva para su participación en el Proyecto KTTSeaDrones. Cuenta con una estructura open-frame, con un peso en seco de 12,45 kg y mide 520 mm de largo, 390 mm de ancho y 290 mm de alto (sin contar dispositivos añadidos). Sibiu Pro es ideal para operaciones de investigación, inspección y manipulación en aguas poco profundas a moderadas de 100 hasta 300 metros de amarre. Está compuesto por ocho propulsores T200 que proporcionan suavidad y estabilidad de navegación junto con un chasis resistente y baterías intercambiables rápidamente. Además cuenta con una cámara 1080p optimizada para medios acuáticos con cuatro faros de 1.500 lúmenes que permiten obtener una imagen clara en entornos de poca visibilidad. Se ofrecen más detalles técnicos sobre Sibiu Pro en (NidoRobotics). El ROV se muestra en la Figura 16.

Figura 16. *Sibiu Pro (Nido Robotics).*

En la Figura 16 no se muestran todos los dispositivos que se han añadido para las actividades del ROV de la Universidad de Huelva, tales como: ecosonda, sónares de barrido lateral, localizador s1, sonar 360 y todos los soportes que unen estos dispositivos al chasis, además del cordón umbilical. Estos se deben tener en cuenta en la hidrodinámica del vehículo, y es por ello que se han modelado en 3D junto con el resto de piezas del ROV.

En el Anexo VII pueden verse los planos de ensamblaje y explosionado con la lista de piezas del conjunto del ROV, diseñado a partir del modelado 3D del mismo en SolidWorks®.

4.2. Modelado 3D del ROV en SolidWorks®

En este apartado se explica el proceso de modelado de piezas, ensamblaje y exportación del diseño del ROV en la herramienta CAD SolidWorks®, siendo fundamental tanto para el procedimiento de simulación CFD como para el procedimiento experimental. En este, no se entra en detalles sobre pasos repetitivos de modelado o realización de planos.

4.2.1. Modelado de piezas

Para el modelado 3D de todas y cada una de las piezas no se dispone de planos con las cotas de cada pieza. Por este motivo, se opta por el diseño de cada pieza conforme se van tomando medidas del ROV propio de la Universidad de Huelva, mediante instrumentos de medida como metro o calibre.

De tal manera, se abre SolidWorks® y se elige la plantilla *Pieza*, para comenzar con el modelado de piezas.

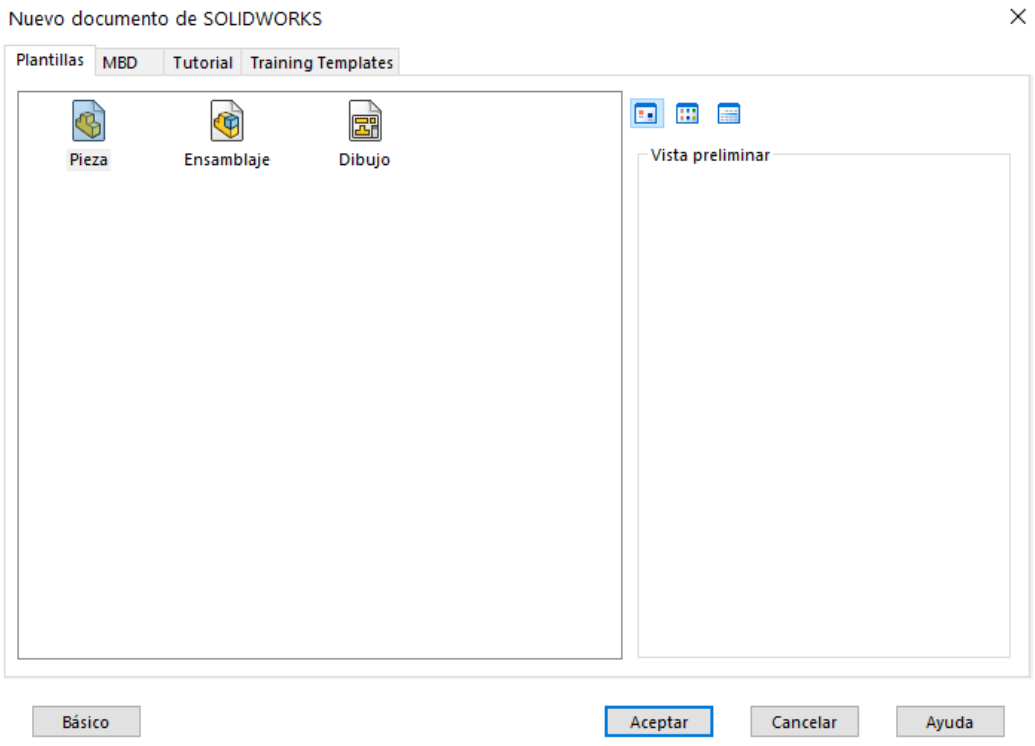


Figura 17. *Nuevo documento de SolidWorks®.*

Como ejemplo sencillo de modelado de piezas se procede a explicar cómo se modela la pieza *nº 13: Flotador delantero derecho* (ver AnexoVII). De tal forma, una vez seleccionada la plantilla *Pieza*, debe aparecer el siguiente entorno gráfico:

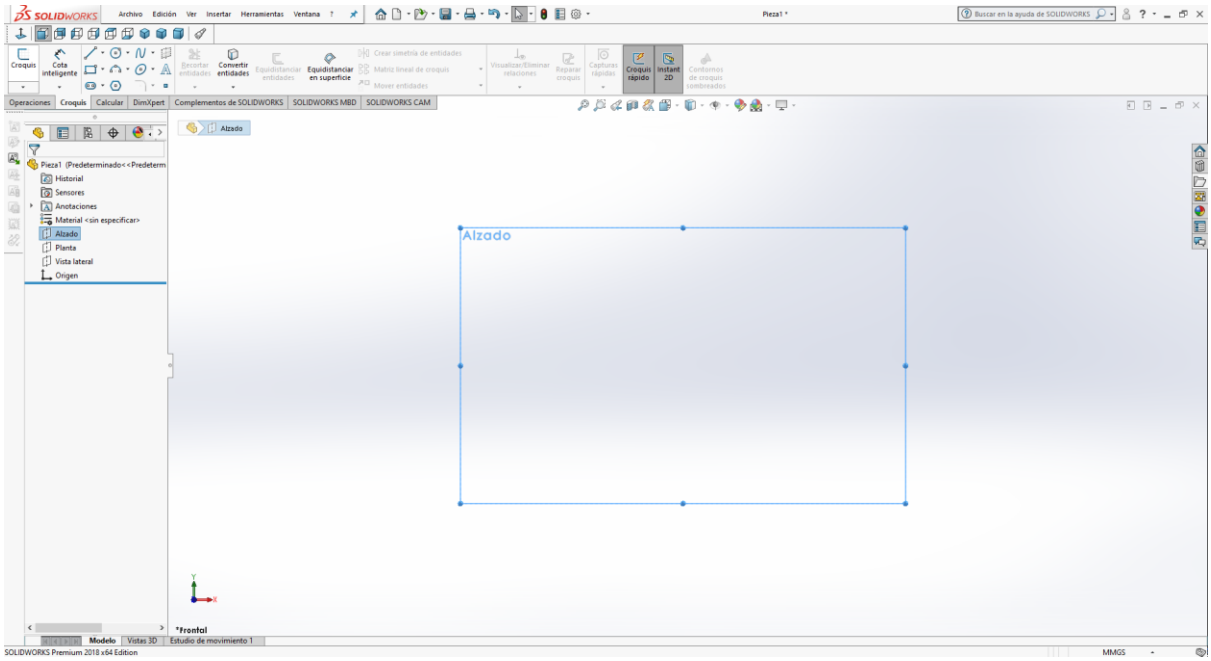


Figura 18. *Interfaz de SolidWorks® de Pieza.*

El proceso de modelado de piezas en SolidWorks®, y en la gran mayoría de herramientas de diseño 3D, consiste en primero seleccionar un plano de trabajo entre *Alzado*, *Planta* y *Perfil*. En cualquiera de estos se debe crear un croquis o boceto, el cual es una disposición de líneas rectas y/o curvas en 2D relacionadas entre sí, formando un conjunto cerrado o abierto dependiendo de la operación que posteriormente al croquis se quiera realizar.

De tal manera se puede elegir el plano *Alzado* y seleccionar *Croquis* para comenzar el primer croquis. En este se utilizan los comandos de croquis y acotación para representar una cara plana de la pieza nº13, de manera que todas las medidas queden definidas según se ha medido con el metro y el calibre. Todas las líneas deben quedar acotadas y restringidas entre ellas y el centro del croquis, quedando así completamente definido. La forma de comprobarlo es ver si alguna línea aparece en azul en vez de negro, estando entonces no restringida.

Además para saber si un croquis ha quedado completamente cerrado, la región interior a las líneas debe quedar sombreada. Esto es necesario para proceder posteriormente con las operaciones básicas de *Extruir saliente/base* y *Extruir corte*. El primer boceto realizado para la pieza nº13 puede verse en la Figura 19.

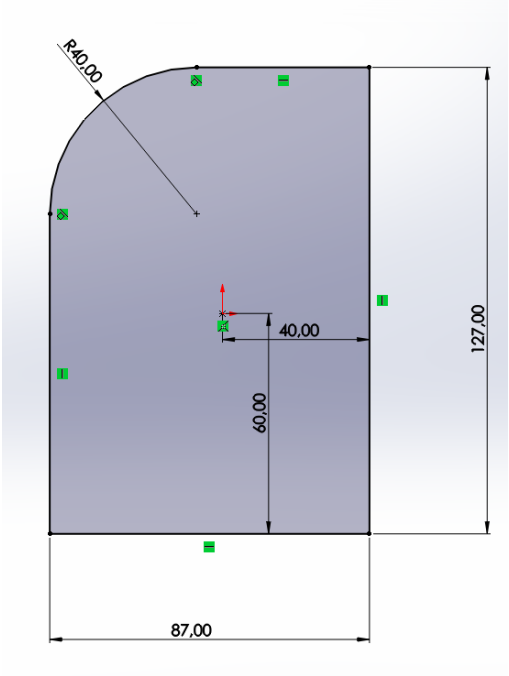


Figura 19. *Primer croquis completamente definido y cerrado.*

En este momento se puede cerrar el croquis y realizar la primera operación. Las operaciones son un conjunto funciones que sirven para hacer crear o transformar la pieza. La primera operación que se realiza para esta pieza es la más básica de todas: *Extruir saliente/base.* Esta operación crea volúmenes prismáticos con una altura especificada a partir de un croquis cerrado. Para ello se debe seleccionar el croquis previamente dibujado en el *Gestor de Diseño del FeatureManager* y seleccionar la operación comentada en la pestaña de *Operaciones*. En este, simplemente se escribe la cota de altura, tal y como se ve en la Figura 20.

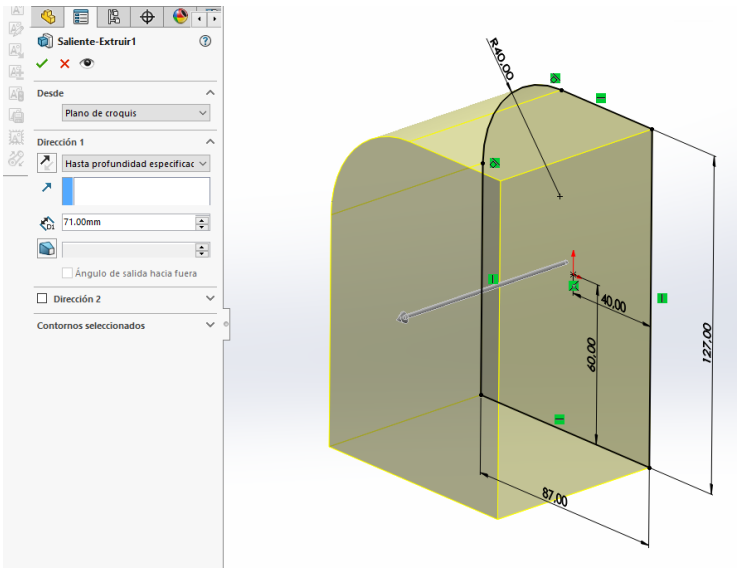


Figura 20. *Primera operación: Extruir saliente/base.*

El resultado de la extrusión debe corresponderse con la Figura 21, en la que se ha pinchado con el ratón la cara lateral izquierda. Sobre esta cara, se va proceder a realizar un nuevo croquis, de igual manera que se hizo para el plano de *Alzado*.

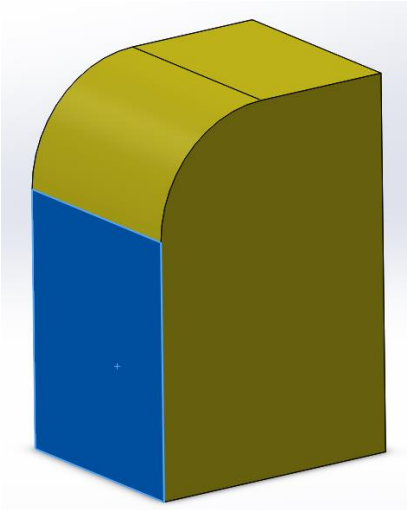


Figura 21. *Resultado de la primera operación. Cara lateral izquierda seleccionada.*

El segundo croquis debe quedar tal y como se muestra en la Figura 22, el cual tiene la misma forma que el primer croquis pero con distintas dimensiones. Debe quedar restringido desde su esquina inferior derecha con la esquina inferior derecha de la cara donde se está realizando el croquis.

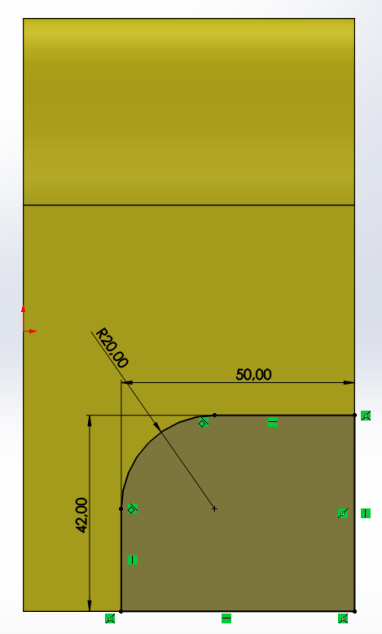


Figura 22. *Segundo croquis completamente definido y cerrado.*

Se puede entonces cerrar el croquis, seleccionarlo y proceder con la segunda operación de *Extruir corte* en la pestaña de *Operaciones*. Esta operación funciona igual que la operación *Extruir saliente/base* pero quitando material en vez de crearlo. Se puede especificar la cota hasta donde se quiere cortar, o bien seleccionar en la pestaña desplegable de *Dirección 1* la opción *Hasta el siguiente* que realiza el corte de todo el material que encuentre en el sentido perpendicular al croquis hacia la pieza.

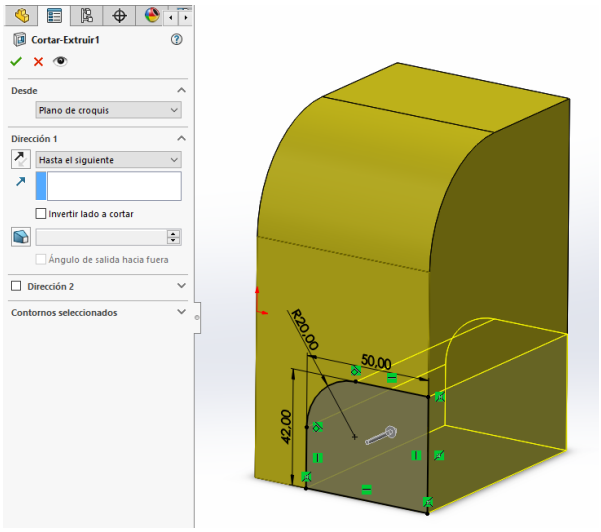


Figura 23. *Segunda operación: Extruir corte.*

De tal forma, queda la pieza completamente definida como puede verse en la Figura 24. Se puede entonces guardar con su correspondiente nombre para posteriormente ensamblarla en la plantilla de ensamblaje. En SolidWorks®, los archivos de piezas se guardan en formato *.prt* o *.sldprt*.

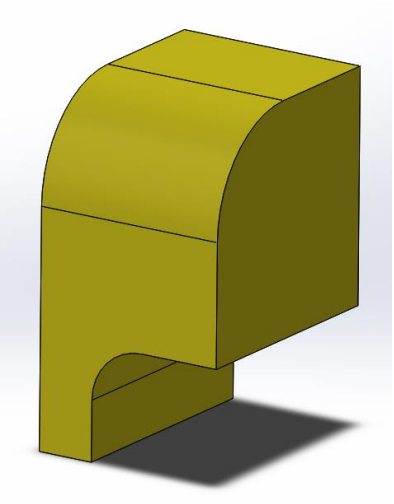


Figura 24. *Pieza nº13 Flotador delantero izquierdo.*

4.2.2. Ensamblaje

En este punto se ha completado ya el modelado de todas las piezas del ROV y se puede comenzar el ensamblaje del mismo, abriendo un nuevo documento de SolidWorks® con la plantilla *Ensamblaje*, tal como se ve en la Figura 25. De tal forma, una vez seleccionada la plantilla *Ensamblaje*, debe aparecer el siguiente entorno gráfico:

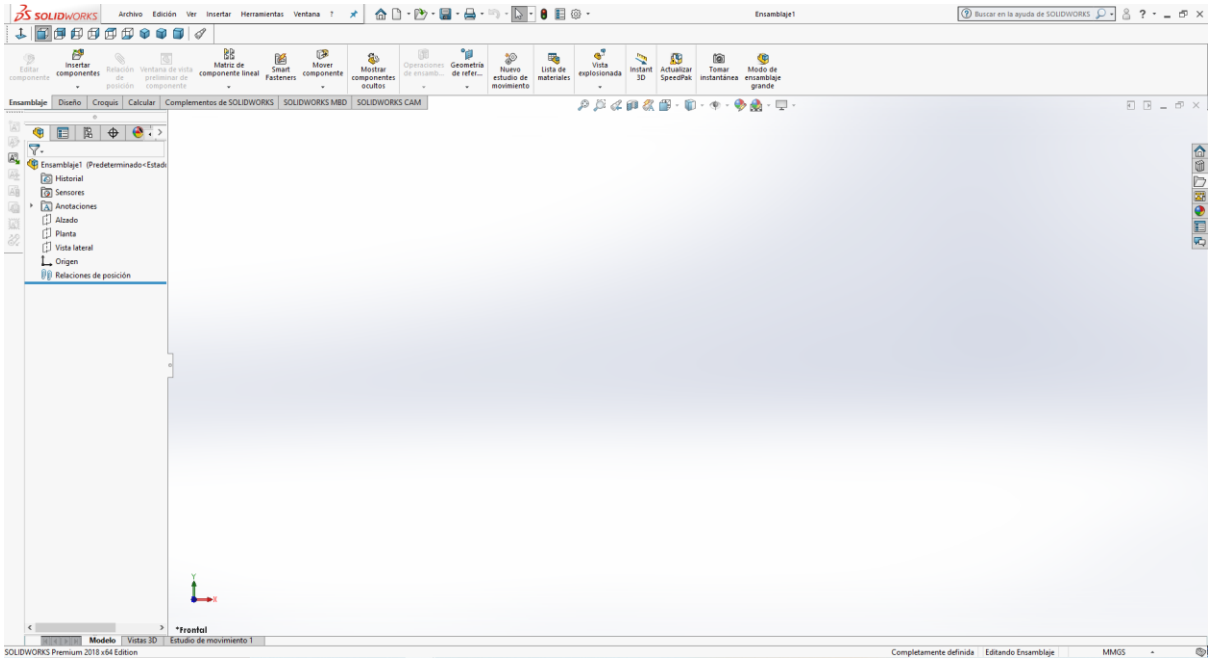


Figura 25. *Interfaz de SolidWorks® de Ensamblaje.*

El proceso de ensamblaje de piezas en SolidWorks®, y en la gran mayoría de herramientas de diseño 3D, consiste en primero insertar un componente o pieza desde la carpeta de piezas mediante el botón *Insertar componentes*. Esta primera pieza será fija automáticamente. Todas las piezas que sucesivamente se vayan insertando "flotarán" en la plantilla de *Ensamblaje*, y es por ello que es necesario fijarlas, es decir, restringirlas de una determinada manera con piezas fijas para formar el conjunto ordenado del ROV.

Se puede comprobar que una pieza no está fija pichando en ella con el botón izquierdo del ratón y dejándolo pulsado mientras lo mueves, arrastrando la pieza con el ratón. Para rotar una pieza (para dejarla en una posición favorable para restringir) se pincha y deja pulsado sobre la pieza con el botón derecho del ratón y se mueve en cualquier dirección para rotar.

Como la primera pieza ya es fija, bastará entonces con aplicar restricciones a la segunda pieza con respecto a la primera. Para ello se pichan primero en la pieza a restringir y

luego en *Relación de Posición*. Añadir restricciones o relaciones de posición consiste en establecer en que posición y orientación se encuentra una pieza con respecto a otra, añadiendo fijaciones de movimiento. Un ejemplo de esto puede verse en la Figura 26.

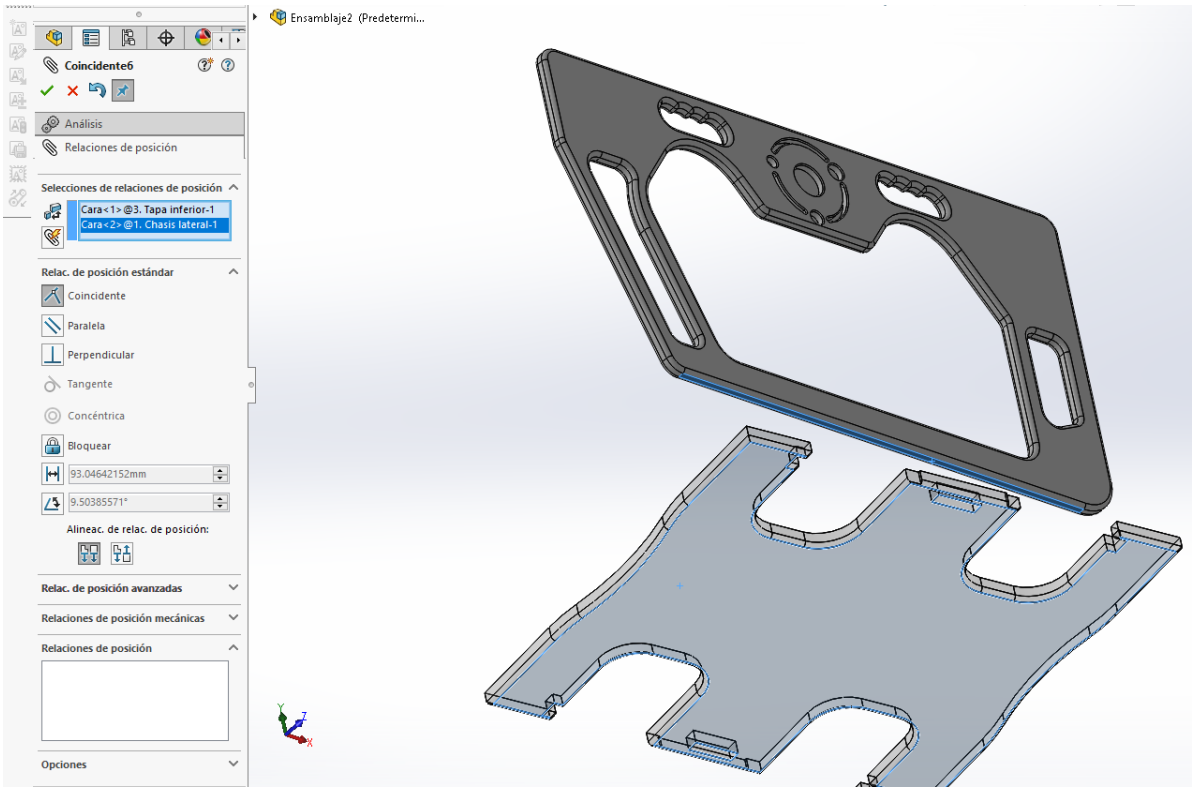


Figura 26. *Ejemplo de relación de coincidencia.*

En la Figura 26 se ha aplicado una relación de coincidencia para fijar la cara inferior de la pieza no fija *nº 2 Chasis inferior* con la cara inferior de la pieza fija *nº 5 Chasis lateral izquierdo* (ver AnexoVII) para que ambas estén el mismo plano definido por la pieza fija. Se puede comprobar que la relación de posición se ha aplicado intentando mover rotar la pieza no fija, y viendo cómo se mantiene la coincidencia entre ambas caras. También se pueden aplicar restricciones mediante los planos de *Alzado*, *Planta* y *Vista lateral* de cada pieza.

Entre los tipos de relaciones de posición se pueden aplicar de tipo Coincidente, Paralela, Perpendicular, Tangente, Concéntrica o Bloquear. Para restringir completamente una pieza con respecto a otra, por una parte se necesitan al menos tres relaciones de posición. De esta forma queda restringido el movimiento de translación y rotación de una pieza con respecto a otra. Es importante que el ensamblaje final del ROV se oriente en algún sentido de la dirección *X*, para que se siga el criterio tomado de que el eje *X* sea el del movimiento del ROV de avance.

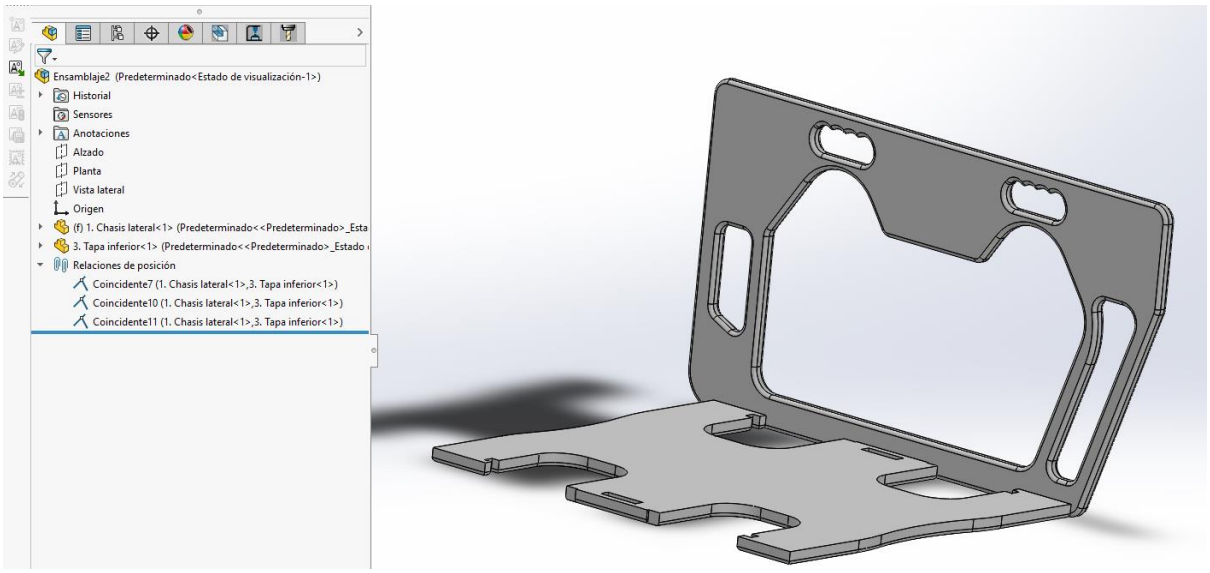


Figura 27. *Relaciones de posición de la pieza nº2 con respecto a la nº5.*

Todas las relaciones de posición que se van aplicando pueden consultarse y editarse en el *Gestor de Diseño del FeatureManager*. Este proceso se de añadir piezas al ensamblaje y restringirlas con respecto a piezas fijas se va repitiendo para formar el conjunto ordenado y fijo de piezas del ROV en el formato de SolidWorks® de ensamblaje *.sldasm* o *.asm*. Para agilizar la inserción y restricción de componentes existe la posibilidad de hacer simetrías o matrices de componentes lineales o circulares como se ve en la Figura 28.

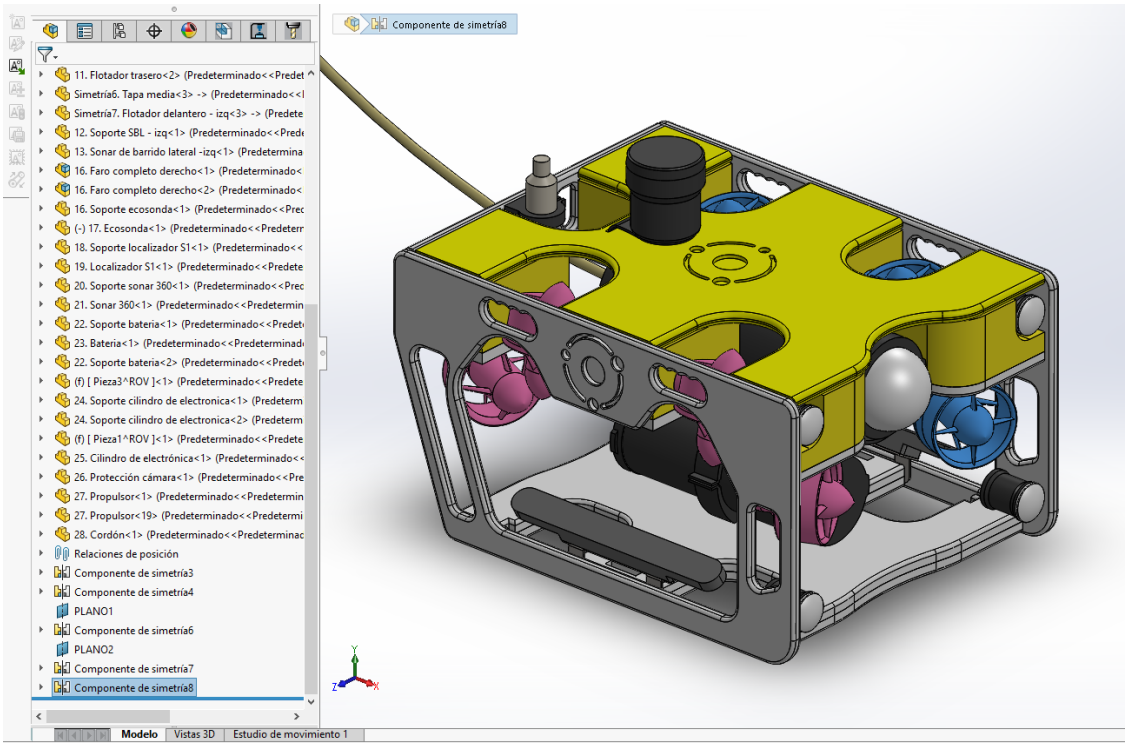


Figura 28. *Simetría de 4 propulsores con respecto a un plano medio.*

4.2.3. Exportación del ensamblaje

La exportación del ROV para su posterior importación en OpenFOAM® se debe realizar en un único archivo en formato *.stl* o *.obj*. Para ello simplemente habría que guardar el ROV de SolidWorks® en cualquiera de estos dos formatos. El problema está en que cuando se quiere realizar este paso sobre algún archivo de ensamblaje *.sldasm* o *.asm*, realmente se realiza la exportación de cada pieza o componente individual en archivos individuales, y eso no es lo que se pretende. Para solucionar este problema, se debe guardar el ensamblaje del ROV como un archivo aparte de tipo pieza (formato *.prt* o *.sldprt.*) y realizar en este algunos ajustes.

El primer ajuste consiste en combinar los cincuenta y cuatro sólidos separados (piezas) en un solo sólido. Para ello se pincha en *"Insertar"* → *"Operaciones"* → *"Combinar"*. En este se debe abrir la carpeta *Sólidos* del *Gestor de Diseño del FeatureManager* y agregar uno a uno los cincuenta y cuatro sólidos.

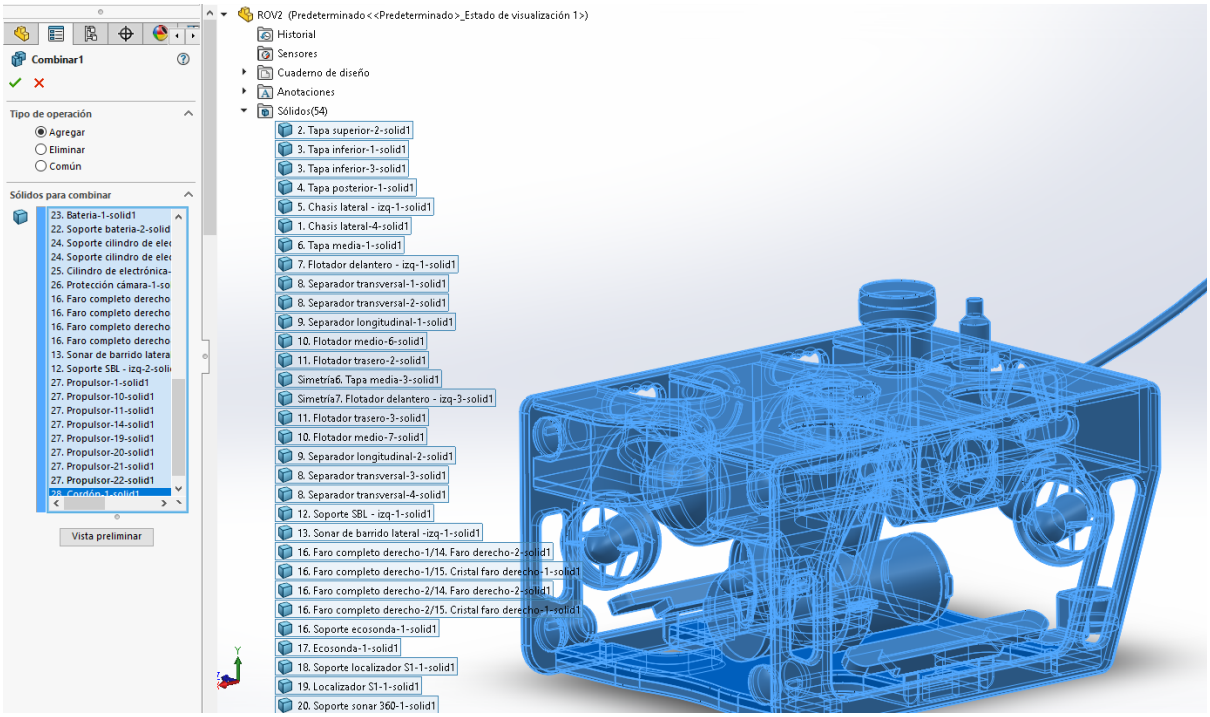


Figura 29. *Combinar sólidos del ROV.*

De esta forma, el ensamblaje pasa a ser un único sólido en formato pieza, por lo que la exportación posterior en formato *.stl* o *.obj* será de un único archivo del ROV completo.

Además, se debe tener en cuenta que la orientación de los ejes del sistema de referencias (SdR a partir de ahora) del ROV en SolidWorks®, va a coincidir con la orientación de los ejes SdR de ParaView®. Esto influye en la orientación del ROV dentro del mallado posteriormente generado en OpenFOAM®. De tal forma, si ambos SdR no coinciden, existen dos opciones: ajustar el mallado y las variables definas (como la velocidad) en OpenFOAM® al SdR establecido en SolidWorks® o ajustar el SdR de SolidWorks® al de ParaView®.

Durante el ensamblaje del ROV se orientó el eje *X* hacia el avance contrario del ROV, el eje *Y* hacia el movimiento vertical contrario, y el eje *Z* hacia el desvío, siguiendo la regla de la mano derecha. Por tanto se debe cambiar el sentido del eje *X*.

Por simplicidad, se opta por reorientar solo el sentido del eje *X* girando 180º grados el SdR, tomando como eje de giro el *Y*, e invirtiéndose así el sentido de los ejes *X* y *Z*. Así quedaría el sistema reorientado igual que en ParaView® y se puede definir la velocidad en OpenFOAM® con una incidencia sobre el ROV en sentido *X* positivo (simulando el avance del mismo sobre el fluido). Para ello, se pincha en *"Insertar"* → *"Geometría de referencia"* → *"Sistema de coordenadas…"*. En este, simplemente habría que invertir la dirección del eje *X*, tal como se ve en la Figura 30.

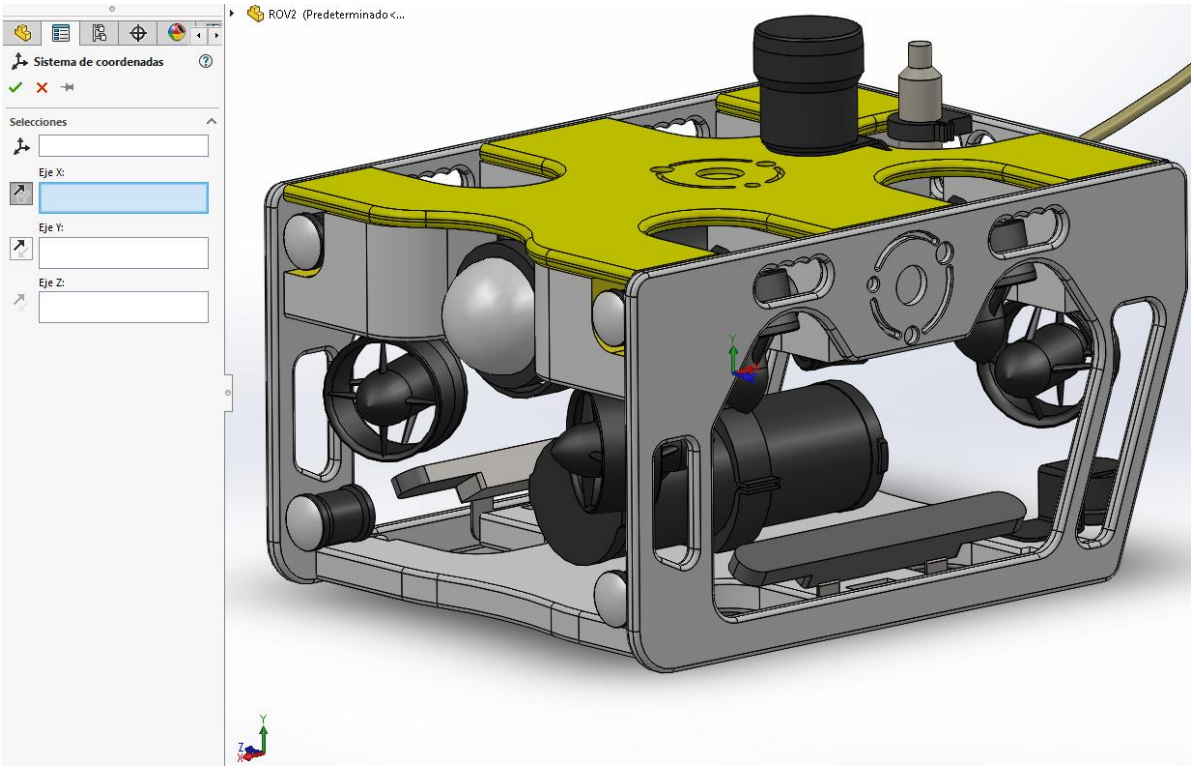


Figura 30. *Creación de un nuevo SdR en SolidWorks®.*

Nótese en la Figura 30 que el SdR de SolidWorks® y el nuevo SdR tienen misma orientación del eje *Y*, y contraria orientación de los ejes *X* y *Z*.

Entonces, si los sólidos han sido combinados y los ejes han sido orientados convenientemente, se puede proceder a la exportación en formato *.stl* por ejemplo. Para ello, se guarda como *STL*, y en la ventana de opciones se deben realizar los siguientes ajustes adicionales. Por una parte, se debe *Crear como ASCII*, ya que es el código de caracteres de alto nivel que puede leer OpenFOAM®, escrito en lenguaje C. La *Unidad* debe ser pasada a *Metros*, ya que cualquier cantidad de distancia, OpenFOAM® lo entiende como metros, por lo que si en el *STL* pone que el ROV mide 390 de ancho, OpenFOAM® lo entenderá como 390 metros en vez de 0,39 metros. La resolución se debe ajustar según la complejidad del *STL* y simulación de datos en OpenFOAM®. Las primeras simulaciones se hicieron con *Resolución Rápida*, pero tras optimizar el mallado y la simulación, se pudo trabajar con una *Resolución* de *Alta calidad*. Por último, se debe cambiar el *Sistema de coordenadas de salida* de predeterminado a *Sistema de coordenadas1*, para elegir el SdR que se creó previamente.

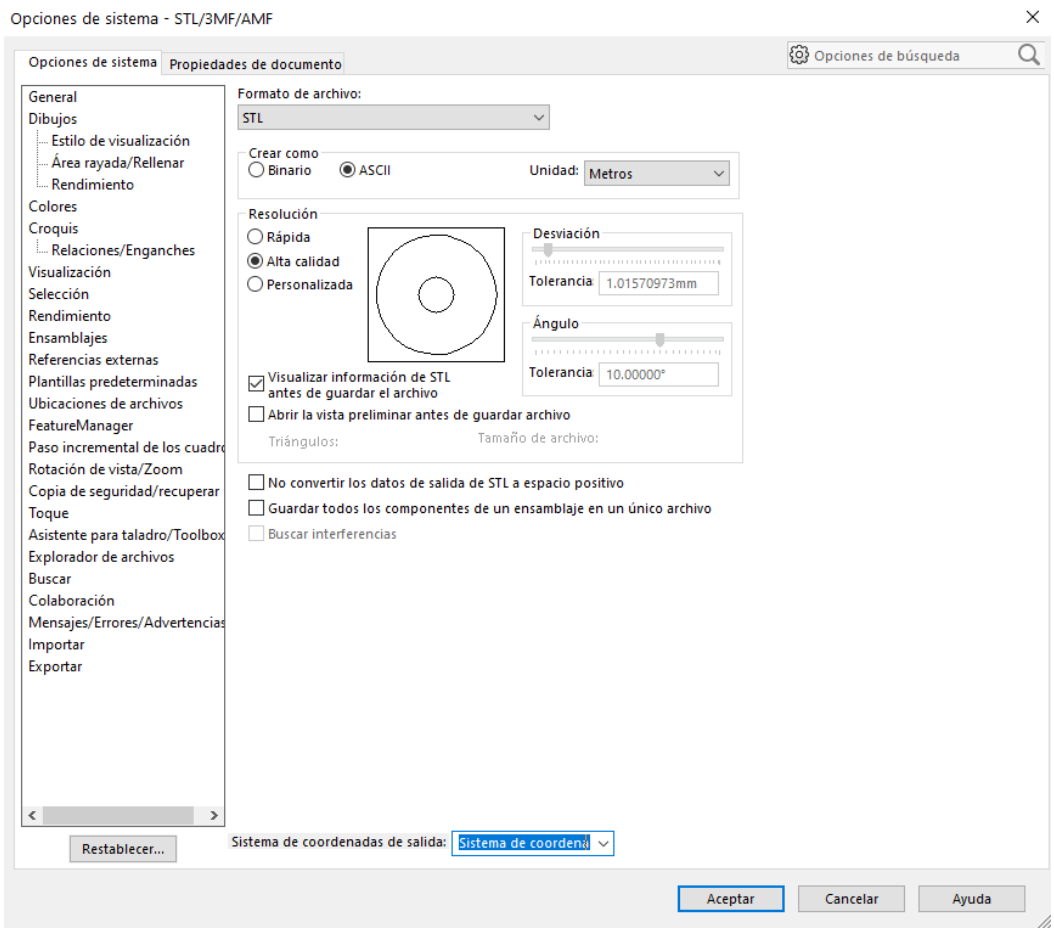


Figura 31. *Ajustes de exportación del ROV en STL.*

4.3. Procedimiento CFD en OpenFOAM® y ParaView®

En este apartado se detalla todo el proceso CFD seguido para el análisis hidrodinámico del caso de estudio SibiuPro. Este procedimiento comprende el preprocesado y simulación en OpenFOAM® y postprocesado en ParaView®, entendiendo que el lector cuenta con conocimientos básicos de ambas herramientas CFD.

4.3.1. Preprocesado

La configuración definida para el caso SibiuPro, tiene la estructura de archivos de la versión 5.0 de OpenFOAM® que se muestra en la Figura 32.

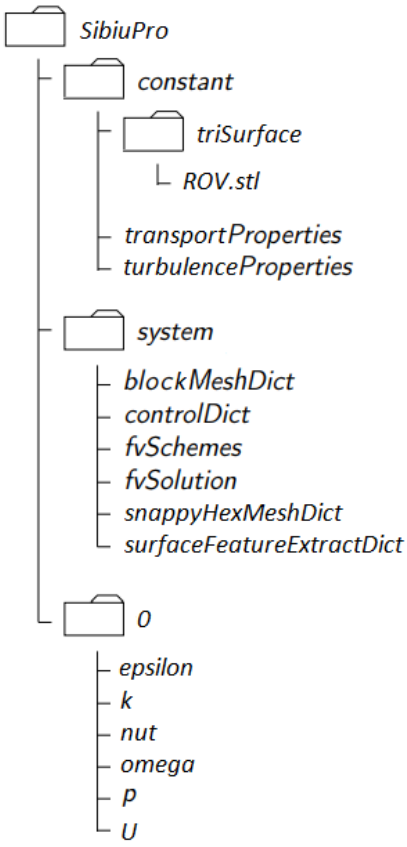


Figura 32. *Estructura del caso SibiuPro.*

El caso se encuentra a disposición del lector en el siguiente enlace de descarga:

SibiuPro.rar

En los siguientes subapartados de preprocesado se muestra la configuración de cada archivo y la ejecución de comandos de generación del mallado seguida.

4.3.1.1. Configuración y generación del mallado

1. Configuración de *blockMeshDict*

El mallado externo se ha configurado en el archivo *blockMeshDict* como un prisma rectangular con unas dimensiones de 4 metros de largo *X*, 1 metro de alto *Y*, y 1,6 metros de ancho *Z*. Este mallado se compone en las tres dimensiones de celdas hexaédricas de lados iguales, con una cantidad de 100 celdas en *X*, 25 celdas en *Y*, y 40 celdas en *Z*, proporcional a las dimensiones del prisma rectangular (100.000 celdas en total).

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
    (0.3 -0.3 -0.6)             // 0
    (4.3 -0.3 -0.6)             // 1
    (4.3 0.7 -0.6)              // 2
    (0.3 0.7 -0.6)              // 3
    (0.3 -0.3 1)                // 4
    (4.3 -0.3 1)                // 5
    (4.3 0.7 1)                 // 6
    (0.3 0.7 1)                 // 7
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (100 25 40) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
    Inlet
    {
        type patch;
        faces
        (
            (3 7 4 0)
        );
    }
    Outlet
    {
        type patch;
        faces
        (
            (2 1 5 6)
        );
    }
```

```
    Walls
    {
        type wall;
        faces
        (
            (0 4 5 1)
            (2 6 7 3)
            (1 2 3 0)
            (7 6 5 4)
        );
    }
);

// ************************************************************************* //
```

Figura 33. *blockMeshDict de SibiuPro.*

Además, se definen los nombres de los tres parches del mallado externo: *Inlet*, *Outlet* y *Walls*. La entrada de fluido *Inlet* se define de tipo parche mediante la cara de vértices del prisma rectangular *(3 7 4 0)* en ese orden. La salida de fluido *Outlet* se define de tipo parche mediante la cara de vértices del prisma rectangular *(2 1 5 6)* en ese orden. Por otra parte, los muros *Walls* se definen de tipo muro en un único parche mediante las cuatro caras restantes y definidas mediante sus vértices.

Para la generación del mallado externo definido se ejecuta el comando:

```
blockMesh
```

2. Visualización del mallado externo en ParaView®

Para comprobar el mallado y la posición y orientación del ROV dentro del mismo, se ejecuta el siguiente comando que abre el mallado en ParaView®, y se carga sobre este, el ROV en formato *.stl*.

```
paraFoam
```

Los vértices definidos previamente sitúan al ROV aproximadamente en el centro del mallado generado para las dimensiones YZ y separado 0,9 metros en la dimensión X del parche *Inlet*. Se define así para apreciar en el postprocesado la estela de la interacción entre el fluido y el ROV en el centro del mallado. La orientación del mismo es la que fue definida en SolidWorks® mediante el nuevo SdR creado y establecido en su exportación en formato *.stl*. Esta posición y orientación del ROV (para el movimiento de avance) dentro del mallado puede apreciarse en las vistas del mismo recopiladas en la Figura 34.

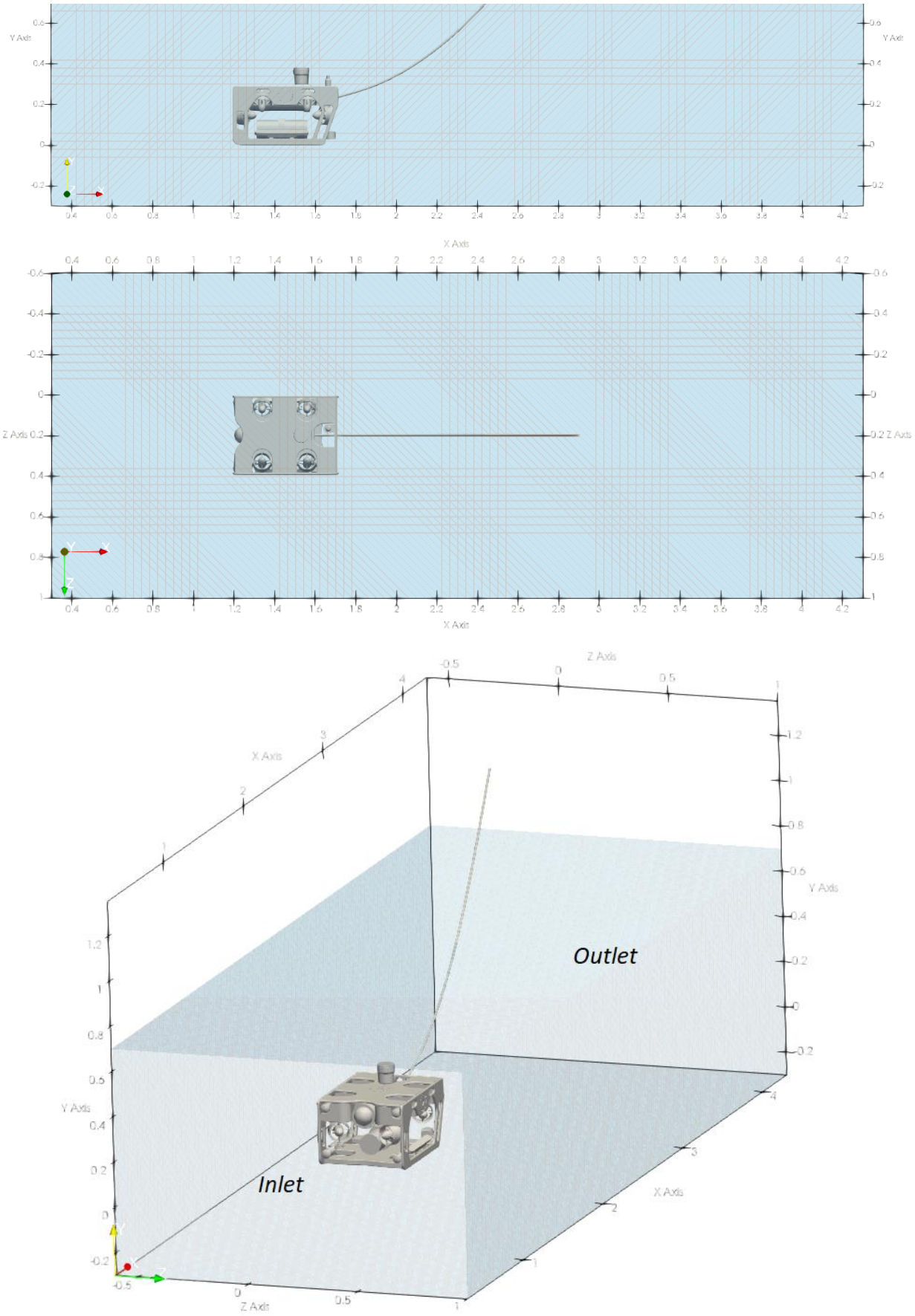


Figura 34. *Vistas de perfil, planta, y perspectiva 3D del ROV dentro del mallado.*

3. Configuración de *surfaceFeatureExtractDict*

La configuración de extracción superficial mediante *surfaceFeatureExtractDict* se ha establecido con *includedAngle* en *180* para que se extraigan todas las aristas del *STL* del ROV. Se ha probado para este caso que establecer *writeObject* a *yes* no es de gran utilidad por la extracción aleatoria de archivos STL que realiza.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ROV.stl
{
    extractionMethod    extractFromSurface;

    extractFromSurfaceCoeffs
    {
        includedAngle   180;
    }

    writeObj                no;
}

// ************************************************************************** //
```

Figura 35. *surfaceFeatureExtract de SibiuPro.*

Para proceder a la extracción superficial del ROV se ejecuta el comando:

```
surfaceFeatureExtract
```

Se recuerda que para poder ejecutar el comando, previamente se debe incluir el archivo de exportación a STL de SolidWorks® *ROV.stl* (extensión en minúsculas) en el directorio *constant/triSurface*.

La ejecución del comando genera el archivo *constant/extendedFeatureEdgeMesh/ ROV.extendedFeatureEdgeMesh*, necesario para la posterior ejecución de *snappyHexMesh*.

4. Configuración de *snappyHexMeshDict*

El mallado interno se ha configurado en el archivo *snappyHexMesh* introduciendo dos geometrías. Por una parte el propio ROV tras su extracción superficial, y por otra parte una caja llamada *box1* de dimensiones de 2,5 metros de largo *X*, 1 metro de alto *Y*, y 1 metro de ancho *Z*. El cometido de *box1* es refinar el volumen de interacción entre el fluido y el ROV, definido mediante el vértice mínimo *(1 -0,3 -0,3)* y vértice máximo *(3,5 0,7 0,7)*. El refinado de este volumen, convierte cada celda del mallado externo en 8 celdas iguales. En cuanto al mallado del ROV se generado de nivel *4* con un *resolveFeatureAngle* de *20* mediante *castellatedMesh* y suavizado mediante *snap*. La función *addLayers* está desactivada porque las pruebas que se realizó con su activación casi no mejoran la calidad del mallado, aumentado considerablemente el tiempo de ejecución del comando.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

castellatedMesh true;
snap            true;
addLayers       false;


geometry
{
    ROV.stl
    {
        type triSurfaceMesh;
        name ROV;
    }

    box1
    {
        type searchableBox;
        min (1 -0.3 -0.3);
        max (3.5 0.7 0.7);
    }
};


castellatedMeshControls
{
    maxLocalCells 100000;
    maxGlobalCells 2000000;
    minRefinementCells 2;
    nCellsBetweenLevels 2;

    features
    (
        {
            file "ROV.eMesh";
            level 4;
        }
    );
```

```
    refinementSurfaces
    {
        ROV
        {
            level (2 2);
        }
    }

    resolveFeatureAngle 20;


    refinementRegions
    {
        ROV
        {
            mode inside;
            levels ((1 2));
        }

        box1
        {
            mode inside;
            levels ((1.0 4));

        }
    }

    locationInMesh (2 0 0);

    allowFreeStandingZoneFaces true;
}

snapControls
{
    nSmoothPatch 4;
    tolerance 4;
    nSolveIter 5;
    nRelaxIter 5;
    nFeatureSnapIter 5;
    implicitFeatureSnap true;
    explicitFeatureSnap false;
    multiRegionFeatureSnap false;
}


addLayersControls
{
    relativeSizes true;

    layers
    {
        "ROV_*"
        {
            nSurfaceLayers 3;
        }
    }
```

```
    expansionRatio 1.1;
    finalLayerThickness 0.9;
    minThickness 0.05;
    nGrow 0;
    featureAngle 30;
    nRelaxIter 5;
    nSmoothSurfaceNormals 1;
    nSmoothNormals 3;
    nSmoothThickness 2;
    maxFaceThicknessRatio 0.5;
    maxThicknessToMedialRatio 0.3;
    minMedianAxisAngle 90;
    nBufferCellsNoExtrude 0;
    nLayerIter 30;
    nRelaxedIter 10;
}

meshQualityControls
{
    maxNonOrtho 65;
    maxBoundarySkewness 20;
    maxInternalSkewness 4;
    maxConcave 80;
    minVol 1e-13;
    minTetQuality 1e-9;
    minArea -1;
    minTwist 0.05;
    minDeterminant 0.001;
    minFaceWeight 0.05;
    minVolRatio 0.01;
    minTriangleTwist -1;
    nSmoothScale 4;
    errorReduction 0.75;

    relaxed
    {
        maxNonOrtho 75;
    }
}

debug 0;
mergeTolerance 1E-6;

// ************************************************************************* //
```

Figura 36. *snappyHexMesh de SibiuPro.*

Para proceder a generar el mallado interno definido se ejecuta el siguiente comando:

```
snappyHexMesh
```

El PC que se ha utilizado para su ejecución, genera el mallado resultante de 1.894.553 celdas en 15 min, frente a los 60 minutos que necesita si se activa la función *addLayers*. Este PC cuenta con un procesador AMD Ryzen 5 3550H y una RAM de 8 GB.

5. Visualización del mallado externo e interno en ParaView®

   Para comprobar el resultado del mallado se abre ParaView®:

```
paraFoam
```

Es de destacar la gran cantidad de pruebas de configuración que se han realizado tanto en *snappyHexMeshDict* como en *blockMeshDict*, para obtener un mallado externo e interno de alta resolución, pero sin que posteriormente se colapse la simulación. La calidad del mallado *snap* resultante puede verse en las siguientes Figura 37, 38, 39 y 40.

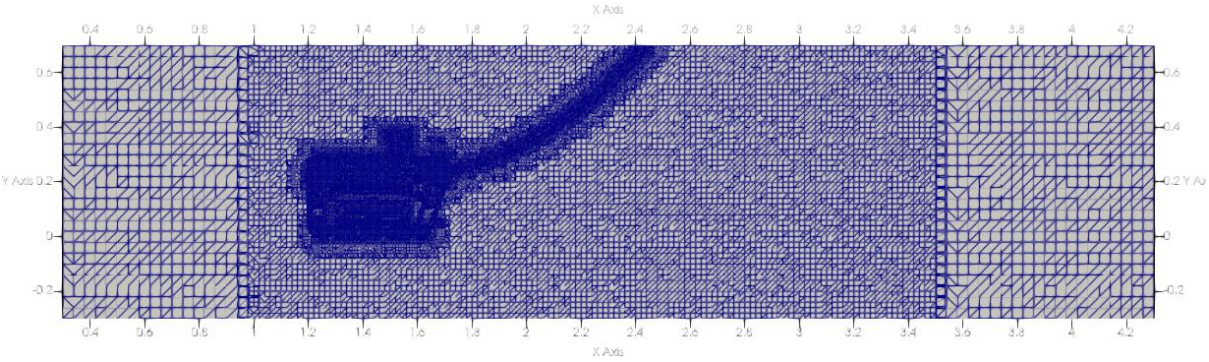


Figura 37. *SurfaceWithEdges del mallado resultante completo.*

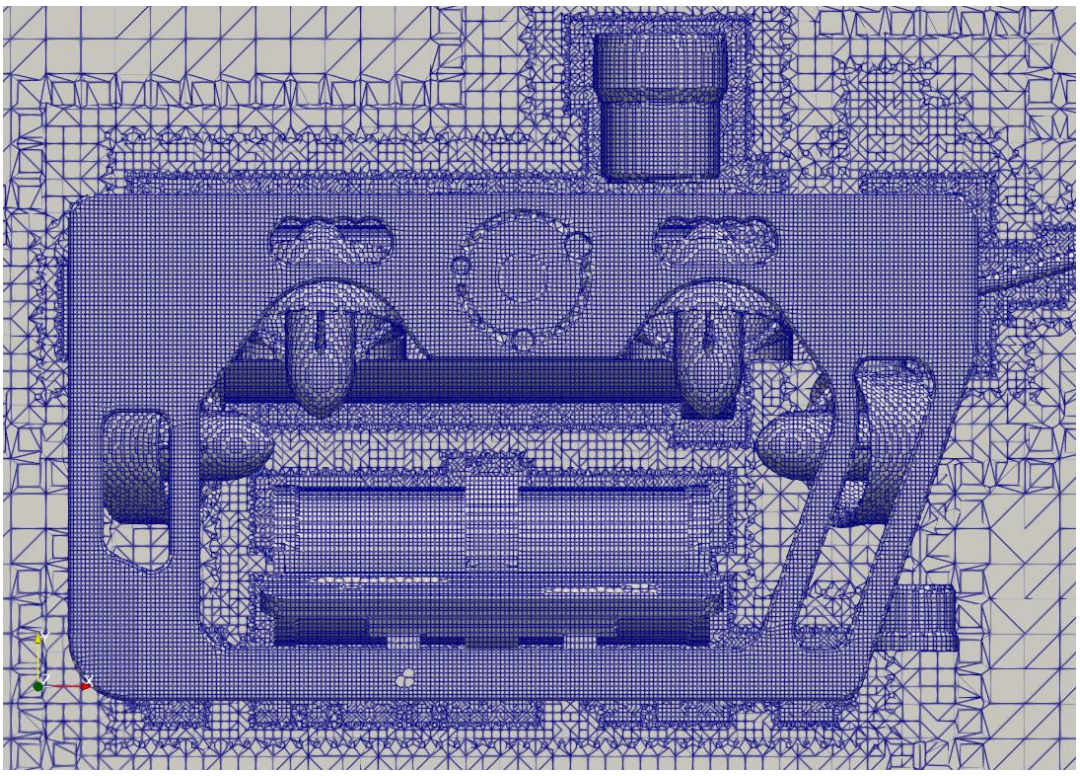


Figura 38. *Zoom al SurfaceWithEdges del mallado resultante completo.*

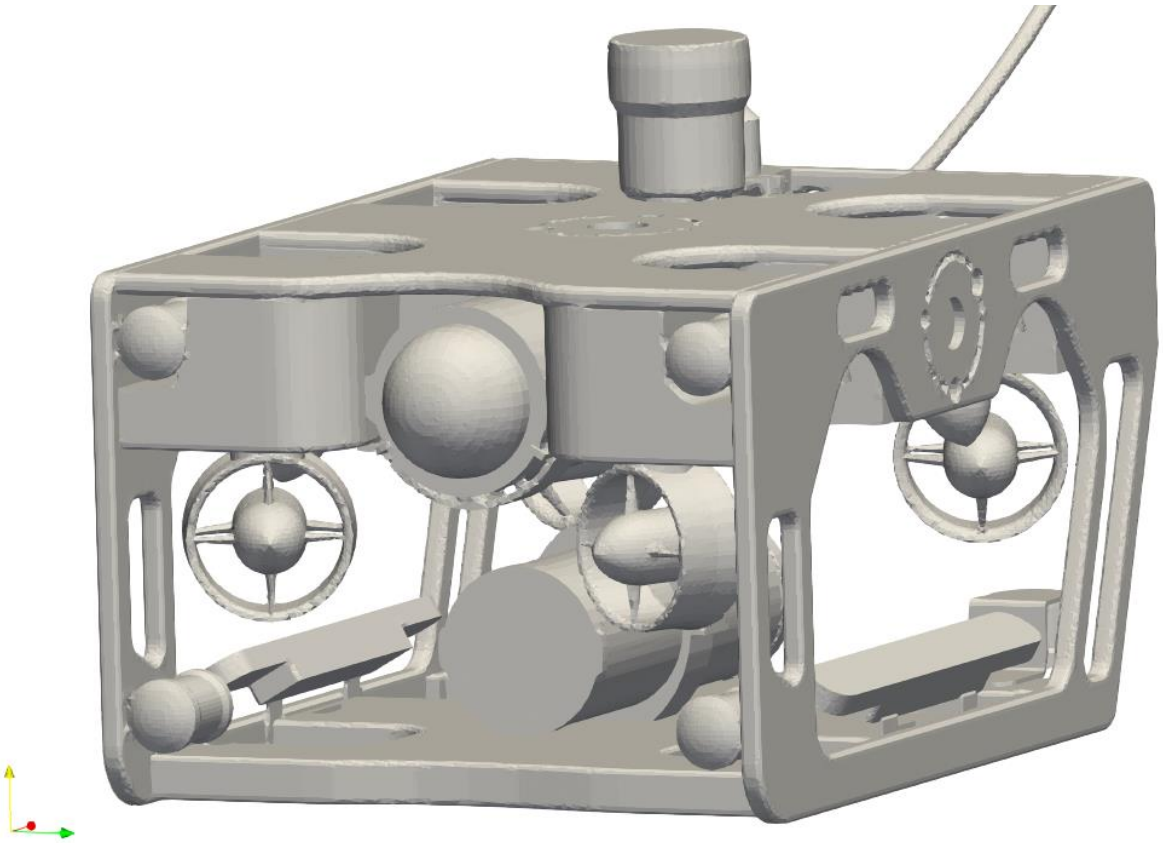


Figura 39. *Surface del mallado resultante del ROV de 275.216 caras.*

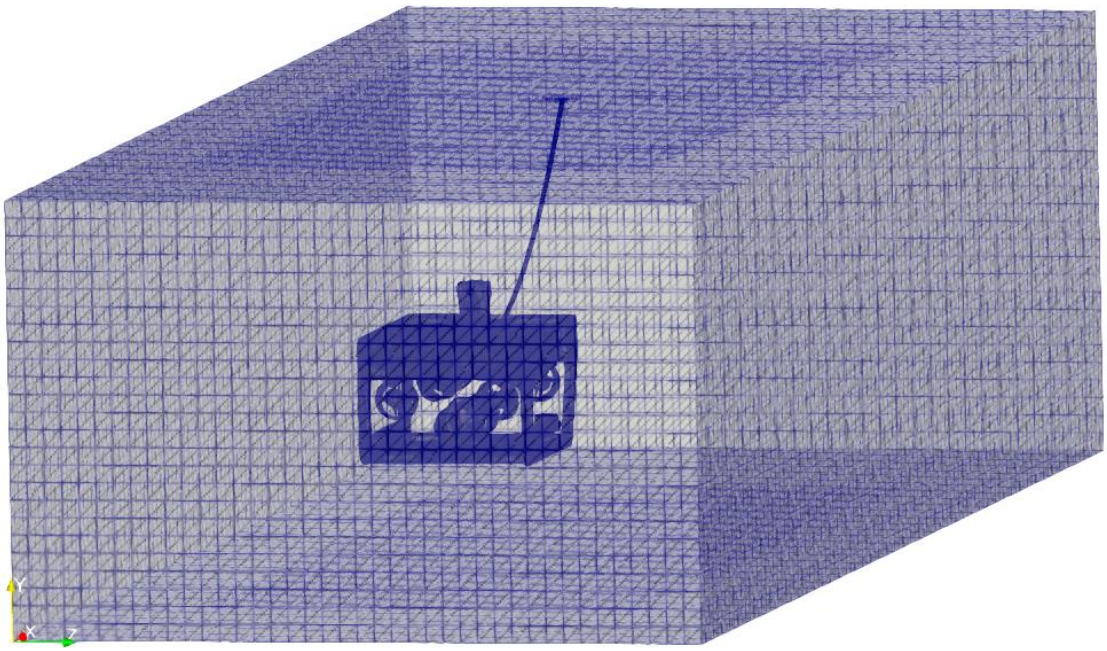


Figura 40. *Vista 3D traslucida del SurfaceWithEdges del mallado resultante completo.*

6. Sustitución del *polyMesh*

Dentro del caso *SibiuPro* se generan dos directorios de tiempo cuyos nombres son los dos primeros saltos de tiempo que se habrían ejecutado al iniciar la simulación, que en este caso son *0.04* y *0.08*. En el directorio *0.04* se encuentra el directorio *polyMesh* que recoge los resultados del mallado generado por la función *castellatedMesh* aplicada al mallado externo. Por otra parte, en el directorio *0.08* se encuentra el directorio *polyMesh* que recoge los resultados del mallado generado por la función *snap* aplicada al mallado de *castellatedMesh*. Este último mallado es el final resultante que se ha mostrado en las Figuras 37, 38, 39 y 40, donde posteriormente se ejecutará la simulación. Para que sea así, habría que sustituir el directorio *constant/polyMesh* generado por *blockMesh*, por el directorio *0.08/polyMesh*. Hecho este paso, los directorios *0.04* y *0.08* pueden borrarse. En la Figura 41 se muestran estos pasos a realizar dentro de la estructura del caso *SibiuPro*, tras ejecutar *snappyHexMesh*.

Figura 41. *Pasos de sustitución del polyMesh.*

4.3.1.2. Configuración de simulación y variables

7. Configuración de transportProperties

En el archivo *transportProperties* se ha configurado las propiedades físicas del fluido. El agua de mar a 20 ºC tiene una viscosidad cinemática *nu* de 10^{-6} m^2/s. El *transportModel* es *Newtonian*, ya que el agua es un fluido newtoniano.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              [0 2 -1 0 0 0 0] 1e-06; // sea water kinematic vistosity (20ºC)

// ************************************************************************* //
```

Figura 42. *transportProperties de SibiuPro.*

8. Configuración de *turbulenceProperties*

La complejidad geométrica del caso y el coste computacional que supone, hace necesario la elección de modelos de turbulencia RANS. En el archivo *turbulenceProperties* se hace la elección del modelo de turbulencia RANS k-omegaSST, por ser un modelo que combina las mejores características de k-épsilon y k-omega, tal como se comentó en el Capítulo anterior. Sin embargo el archivo está configurado de modo que se pueda cambiar al modelo de turbulencia RANS k-ε para comparar los resultados.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType RAS;

RAS
{
//    RASModel        kEpsilon;
    RASModel        kOmegaSST;

    turbulence      on;

    printCoeffs     on;
}

// ************************************************************************* //
```

Figura 43. *turbulenceProperties de SibiuPro.*

Por otra parte, para definir las condiciones iniciales de las variables dependientes del modelo de turbulencia, se deben crear en el directorio *0* los archivos configurados de nombre *k*, *epsilon*, y *nut* para k-ε ó *k*, *omega*, y *nut* para k-ωSST.

La configuración de los valores iniciales de k, ε y ω sigue las siguientes fórmulas aproximadas y definidas por OpenFOAM® (OpenFOAM User Guide, 2012).

- Cálculo de *k*.

$$k = \frac{3}{2}\left(U_{ref}\, I\right)^2 \tag{20}$$

Donde U_{ref} es el módulo de la velocidad en la dirección del flujo e I es la intensidad de turbulencia definida en función del número de Reynolds como:

$$I = 0{,}16\, Re^{-\frac{1}{8}} \tag{21}$$

I es una magnitud adimensional que define la relación entre la magnitud de las fluctuaciones turbulentas y la magnitud de la velocidad media característica. Según su valor se considera una turbulencia de baja intensidad para [0, 1], media intensidad para [1, 5] y alta intensidad para [5, 20].

- Cálculo de ε.

$$\varepsilon = \frac{C_\mu{}^{\frac{3}{4}}\, k^{\frac{3}{2}}}{0{,}07\, L} \tag{22}$$

Donde C_μ es la constante empírica de valor 0,09 comentada en el Capítulo anterior, y L es la longitud de entrada característica, el cual se puede entender con el diámetro hidráulico D_h para el cálculo del número de Reynolds. El valor de L de SibiuPro para el movimiento de avance es el ancho del mismo, igual 0,39 m. El criterio definido por OpenFOAM® permite aproximar ε a una variable de tipo escalar.

- Cálculo de ω.

$$\omega = \frac{k^{\frac{1}{2}}}{C_\mu{}^{\frac{1}{4}}\, L} \tag{23}$$

Donde C_μ es la constante empírica de valor 0,09, y L es la longitud de entrada característica. El criterio definido por OpenFOAM® permite aproximar ω a una variable de tipo escalar. Con los cálculos obtenidos de estas variables en su estado inicial de cada modelado de turbulencia, se procede a la configuración de sus archivos en el directorio 0.

9. Configuración de *k*

El archivo *k* define la configuración de la variable energía cinética turbulenta igual 0,001537 m^2/s^2. Los valores obtenidos del número de Reynolds *Re* de 390.000 e intensidad turbulenta *I* de 0,03201 indican que el fluido se encuentra en régimen turbulento de baja intensidad.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Re = u_ref * L / nu = 1 * 0.39 / 10e-6 = 390000
// I = 0.16 * Re^(-1/8) = 0.16 * 390000^(-1/8) = 0.03201
// k = (3/2)*(u_ref * I)^2 = (3/2)*(1 * 0.03201)^2 = 0.001537 m2/s2

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 1.537e-3;

boundaryField
{
    Inlet
    {
        type            fixedValue;
        value           uniform 1.537e-3;
    }
    Outlet
    {
        type            zeroGradient;
    }
    Walls
    {
        type            kqRWallFunction;
        value           uniform 1.537e-3;
    }
    ROV
    {
        type            kqRWallFunction;
        value           uniform 1.537e-3;
    }
}

// ************************************************************************* //
```

Figura 44. *k de SibiuPro.*

En *dimensions* se configura la dimensión de *k* como m^2/s^2.

En *internalField* se configura *k* como uniforme y escalar con el valor calculado según la aproximación de OpenFOAM®.

En *boundaryField* se dividen todos los parches en subdirectorios con los nombres definidos previamente en *blockMeshDict*, incluyendo el parche *ROV* generado por *snappyHexMeshDict*. El parche de entrada *Inlet* se configura de tipo *fixedValue* con el valor uniforme calculado, y el parche de salida *Outlet* se configura de tipo *zeroGradient*. Por otra parte, los parches de los muros que son *Walls* y *ROV* se parchean de tipo función de pared turbulenta *kqRWallFuction* con el valor uniforme calculado.

10. Configuración de *epsilon*

El archivo *epsilon* define la configuración de la variable tasa de disipación de energía cinética turbulenta igual 3,627*10^{-4} m^2/s^3.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// epsilon = 0.09^3/4 * (k^(3/2)) / (0.07 * L)
// epsilon =  0.09^3/4 * (0.001537 ^(3/2)) / (0.07 * 0.39) = 3.627e-4 m2/s3

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 3.627e-4;

boundaryField
{
    Inlet
    {
        type            fixedValue;
        value           uniform 3.627e-4;
    }
    Outlet
    {
        type            zeroGradient;
    }
    Walls
    {
        type            epsilonWallFunction;
        value           uniform 3.627e-4;
    }
    ROV
    {
        type            epsilonWallFunction;
        value           uniform 3.627e-4;
    }
}

// ************************************************************************* //
```

Figura 45. *epsilon de SibiuPro.*

En *dimensions* se configura la dimensión de ε como m^2/s^3.

En *internalField* se configura ε como uniforme y escalar con el valor calculado según la aproximación de OpenFOAM®.

En *boundaryField* se dividen todos los parches en subdirectorios de igual manera que en el archivo *k*. El parche de entrada *Inlet* se configura de tipo *fixedValue* con el valor uniforme calculado, y el parche de salida *Outlet* se configura de tipo *zeroGradient*. Por otra parte, los parches de los muros *Walls* y *ROV* se parchean de tipo función de pared turbulenta *epsilonWallFuction* con el valor uniforme calculado.

11. <u>Configuración de *omega*</u>

El archivo *omega* define la configuración de la variable tasa de disipación específica turbulenta igual 0,1835 1/s.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// omega = k^(1/2) / (0.09^(1/4) * L)
// omega = 0.001537^(1/2) / (0.09^(1/4) * 0.39) = 0.1835 1/s

dimensions      [0 0 -1 0 0 0 0];

internalField   uniform 0.1835;

boundaryField
{
    Inlet
    {
        type            fixedValue;
        value           uniform 0.1835;
    }
    Outlet
    {
        type            zeroGradient;
    }
    Walls
    {
        type            omegaWallFunction;
        value           uniform 0.1835;
    }
    ROV
    {
        type            omegaWallFunction;
        value           uniform 0.1835;
    }
}

// ************************************************************************* //
```

Figura 46. *omega de SibiuPro.*

En *dimensions* se configura la dimensión de ω como 1/s.

En *internalField* se configura ω como uniforme y escalar con el valor calculado según la aproximación de OpenFOAM®.

En *boundaryField* se dividen todos los parches en subdirectorios de igual manera que en el archivo *k*. El parche de entrada *Inlet* se configura de tipo *fixedValue* con el valor uniforme calculado, y el parche de salida *Outlet* se configura de tipo *zeroGradient*. Por otra parte, los parches de los muros *Walls* y *ROV* se parchean de tipo función de pared turbulenta *omegaWallFuction* con el valor uniforme calculado.

12. Configuración de *nut*

El archivo *nut* define la configuración de la variable viscosidad turbulenta en función de la elección del modelo de turbulencia RANS.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    Inlet
    {
        type            calculated;
        value           uniform 0;
    }
    Outlet
    {
        type            calculated;
        value           uniform 0;
    }
    Walls
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
    ROV
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
}

// ************************************************************************* //
```

Figura 47. *nut de SibiuPro.*

En *dimensions* se configura la dimensión de ν_t como m^2/s.

En *internalField* se configura ν_t como uniforme y escalar de valor *0*.

En *boundaryField* se dividen todos los parches en subdirectorios de igual manera que en el archivo *k*. Los parches de entrada *Inlet* y salida *Outlet* se configuran de tipo *calculated*. Configurar ambos parches como *calculated* hará que se resuelva su resultado en saltos de tiempo sucesivos en todas las celdas según la elección del modelo de turbulencia *RANS*. Por otra parte, los parches de los muros *Walls* y *ROV* se parchean de tipo función de pared turbulenta *nutWallFuction*. Como valor inicial se configura como uniforme de valor *0* para todos los parches, ya que en el instante 0 aún no existe turbulencia todavía.

13. Configuración de *p*

El archivo *p* define la configuración de la variable inicial presión/densidad del fluido.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    Inlet
    {
        type            zeroGradient;
    }
    Outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    Walls
    {
        type            zeroGradient;
    }
    ROV
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //
```

Figura 48. *p de SibiuPro.*

Su valor es igual 0 por conveniencia, aunque este no tiene mucha importancia debido a que esta presión es cinemática para un fluido incompresible. El solucionador calculará la presión real en los instantes sucesivos excepto en *Outlet*, que seguirá un valor fijo igual a 0.

En *Inlet*, *Walls* y *ROV* se ha establecido el tipo de condición frontera para *p* es *zeroGradient*, que significa que el gradiente normal de presión es 0, comportamiento habitual de la presión en muros.

14. Configuración de *U*

El archivo *U* define la configuración de la variable velocidad inicial del fluido igual 0, suponiendo que el fluido se encuentra inicialmente en reposo.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    Inlet
    {
        type            fixedValue;
        value           uniform (1 0 0); // 1.944 knots
    }
    Outlet
    {
        type            zeroGradient;
    }
    Walls
    {
        type            noSlip;
    }
    ROV
    {
        type            noSlip;
    }
}

// ************************************************************************* //
```

Figura 49. *U de SibiuPro.*

Mediante la entrada *Inlet* el fluido se acelera hasta alcanzar una velocidad constante de *1* m/s en sentido *X* positivo. La interacción del fluido con el ROV provocará incremento y decremento de la velocidad en sus proximidades.

La razón de configurar *U* a *1* m/s es aproximar al modelo real en el que el ROV interacciona con el fluido en reposo y no al revés. En este modelo real, el ROV se mueve en su avance en sentido *X* negativo con una velocidad media de 1 m/s. Debido a la alta complejidad que supondría para el análisis CFD establecer el modelo real, se supone que el fluido a esa velocidad interacciona con el ROV en reposo, simulando así el avance del mismo.

Por otra parte, los muros se configuran como *noSlip* para que se comporten como paredes, y la salida *Outlet* de modo inverso a la presión, es decir, *zeroGradient*.

15. Configuración de *fvSchemes* y *fvSolution*

La configuración de discretización de volumen finito y la ecuación del solucionador establecidos en los archivos *fvSchemes* y *fvSolution* respectivamente, proviene de casos de OpenFOAM® de características similares. Esto es flujo externo incompresible en estado estacionario y resuelto mediante el algoritmo SIMPLE en modelos de turbulencia RANS.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         steadyState;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      bounded Gauss linearUpwind grad(U);
    div(phi,k)      bounded Gauss limitedLinear 1;
    div(phi,epsilon) bounded Gauss limitedLinear 1;
    div(phi,omega)  bounded Gauss limitedLinear 1;
    div(phi,v2)     bounded Gauss limitedLinear 1;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
    div(nonlinearStress) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

wallDist
{
    method meshWave;
}


// ************************************************************************* //
```

Figura 50. *fvSchemes de SibiuPro.*

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0.1;
        smoother        GaussSeidel;
    }

    "(U|k|epsilon|omega|f|v2)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-05;
        relTol          0.1;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;
    consistent      yes;

    residualControl
    {
        p               1e-2;
        U               1e-3;
        "(k|epsilon|omega|f|v2)" 1e-3;
    }
}

relaxationFactors
{
    equations
    {
        U               0.9; // 0.9 is more stable but 0.95 more convergent
        ".*"            0.9; // 0.9 is more stable but 0.95 more convergent
    }
}


// ************************************************************************* //
```

Figura 51. *fvSolution de SibiuPro.*

16. Configuración de *controlDict*

En el archivo *controlDict* se ha seleccionado el solucionador *simpleFoam*.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     simpleFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         5;

deltaT          0.02;

writeControl    timeStep;

writeInterval   2;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

functions
{
    wallShearStress1
    {
        // Mandatory entries (unmodifiable)
        type                wallShearStress;
        functionObjectLibs    ("libfieldFunctionObjects.so");

        // Optional entries (runtime modifiable)
        patches         (ROV); // (wall1 "(wall2|wall3)");

        // Optional (inherited) entries
        writePrecision  8;
        writeToFile     true;
        useUserTime     true;
        // region          region0;
        enabled         true;
        log             true;
        // timeStart       0;
        // timeEnd         1000;
        executeControl  timeStep;
        executeInterval 1;
        writeControl    timeStep;
        writeInterval   2;
    }
}

// ************************************************************************* //
```

Figura 52. *controlDict de SibiuPro.*

simpleFoam es un solucionador desarrollado especialmente para flujo turbulento incompresible en estado estacionario, que utiliza el algoritmo SIMPLE. El nombre del algoritmo viene de "*Semi-Implicit Method for Pressure Linked Equations*" (Método semi-implícito para ecuaciones ligadas a presión), el cual es ampliamente utilizado para resolver las ecuaciones RANS en flujos bajo presión y transferencia de calor desde los años 70 (OpenFOAM Wiki, 2020).

La simulación comienza en tiempo 0 y termina en tiempo 5 segundos, con un *deltaT* de *0.02* segundos, obteniendo así un número de Courant *Co* menor o igual a 1 que asegura precisión temporal y estabilidad numérica. Los cálculos para hallar el *deltaT* son los siguientes:

$$\delta x = \frac{d}{n} = \frac{4\ m}{200} = 0{,}02\ m \tag{24}$$

Donde, *d* es la longitud del mallado en la dirección de mayor velocidad y *n* es la cantidad de celdas a lo largo de esa longitud. Según *blockMeshDict* la cantidad de celdas debe tomarse como *100*, pero debido al refinado de la geometría *box1* de *snappyHexMeshDict* sobre la longitud, se ha establecido en *200* para el caso más desfavorable, aunque realmente son menos celdas.

$$\delta t = \frac{Co\ \delta x}{|U|} = \frac{1\ \cdot\ 0{,}02\ m}{1\ {}^{m}/_{s}} = 0{,}02\ s \tag{25}$$

Donde $|U|$ es el valor absoluto de la mayor velocidad que atraviesa las celdas a instante *0*, es decir 1 m/s.

Un *deltaT* tan bajo durante 5 segundos implica resolver 250 saltos de tiempo. Sabiendo que el caso cuenta con cinco variables y un mallado de 1.894.553 celdas, tantos saltos de tiempo pueden implicar generar gran cantidad de datos, tiempo de simulación y tiempo de carga de esos datos. Es por ello que se ha establecido un *writeInterval* de *2*, para guardar la mitad de instantes de tiempo. Esto significa que solo se guardan los instantes múltiplos de 0,04 segundos, 125 instantes de tiempo.

Además se ha añadido la función objeto *wallShearStress1* para que calcule en cada instante, una sexta variable adicional de esfuerzo cortante de pared sobre la superficie del ROV. Las entradas para introducir esta variable se han extradido del siguiente enlace del manual de usuario de OpenFOAM®:

https://www.openfoam.com/documentation/guides/latest/doc/guide-fos-field-wallShearStress.html

De tal forma, se han copiado y pegado las entradas y palabras clave de la función objeto *wallShearStress1* sobre la función *function*. En este se han realizado una serie de modificaciones para que se adapte al caso SibiuPro en la versión 5.0 de OpenFOAM®.

Entre estas modificaciones se encuentra comentar las líneas de *región*, *timeStart* y *timeEnd*, y cambiar *writeInterval* a *2* para que la lectura y escritura de datos de *wallShearStress1* coincida con la de las otras variables. Además, como único parche se configura *ROV* para que solo se calcule el esfuerzo cortante de pared sobre la superficie del mismo. Por último se cambia la siguiente línea por la línea que le sigue:

```
libs            (fieldFunctionObjects);
functionObjectLibs    ("libfieldFunctionObjects.so");
```

Con este último archivo quedan todos los archivos configurados, y el preprocesado del caso *SibiuPro* completado, tras la ejecución de comandos de presprocesado.

4.3.2. Simulación

Para ejecutar la simulación se introduce el comando del solucionador:

```
simpleFoam
```

El PC que se está utilizando, con un procesador AMD Ryzen 5 3550H y una RAM de 8 GB, resuelve el caso *SibiuPro* tal y como está configurado sin errores en 3 horas. Los datos generados por los 125 instantes de tiempo de las seis variables suponen un espacio en el disco duro de 27 GB. Estos datos quedan guardados en 125 directorios de tiempo dentro del caso *SibiuPro*.

4.3.3. Postprocesado

Una vez obtenidos los resultados, el procedimiento seguido de postprocesado en ParaView® es el habitual que se sigue para casos CFD, es decir, aplicar convenientemente combinaciones de filtros (slice, contour, glyph, tube, stream tracer, etc.) que permiten visualizar la interacción entre el fluido y la geometría para su interpretación. Este procedimiento de filtrado de datos en casos CFD viene explicado en el apartado de postprocesado en ParaView® del Anexo V, y los resultados obtenidos con su análisis se detalla en el Capítulo siguiente.

Sin embargo, en este apartado si se procede a explicar el procedimiento de cálculo de las fuerzas y coeficientes hidrodinámicos.

Para empezar, se debe cargar los resultados obtenidos de OpenFOAM® en ParaView®, aplicando los parches *ROV* e *internalMesh*, y omitiendo las variables del modelo de turbulencia k-ωSST de *k*, *nut* y *omega*.

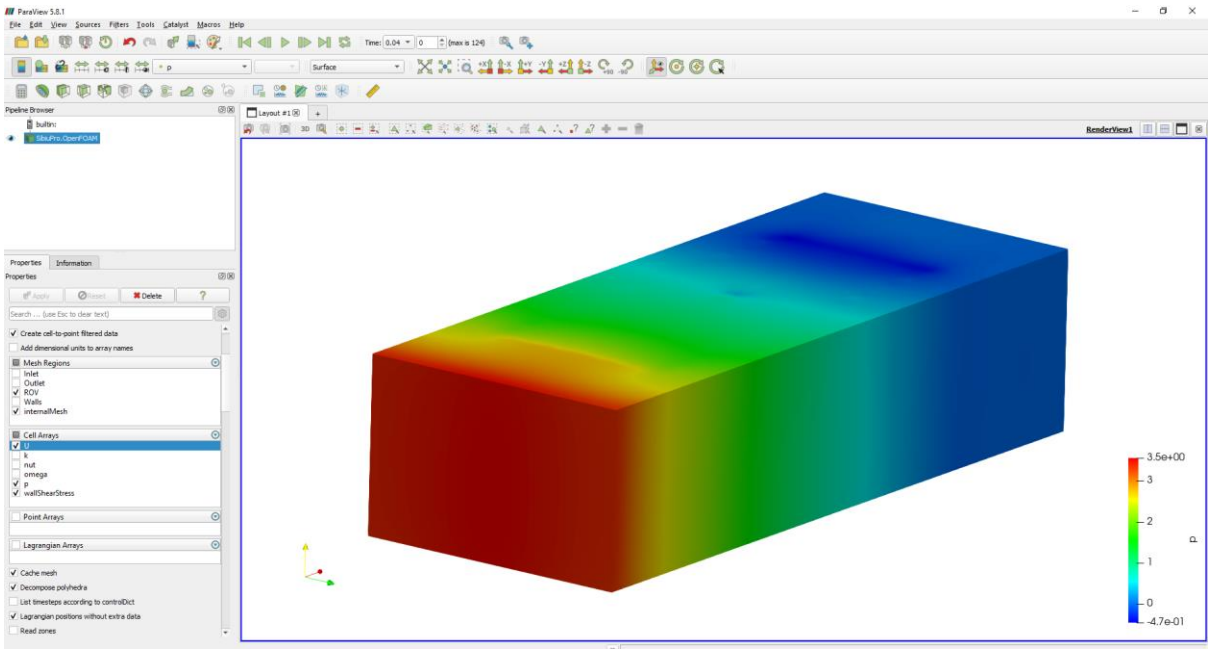


Figura 53. *Carga de SibiuPro.OpenFOAM.*

Sobre el módulo *SibiuPro.OpenFOAM* se aplica el filtro *Extract Block* seleccionando el parche *ROV*, que es el parche del que se quiere calcular los coeficientes hidrodinámicos. Entonces se genera el módulo *ExtractBlock1*.

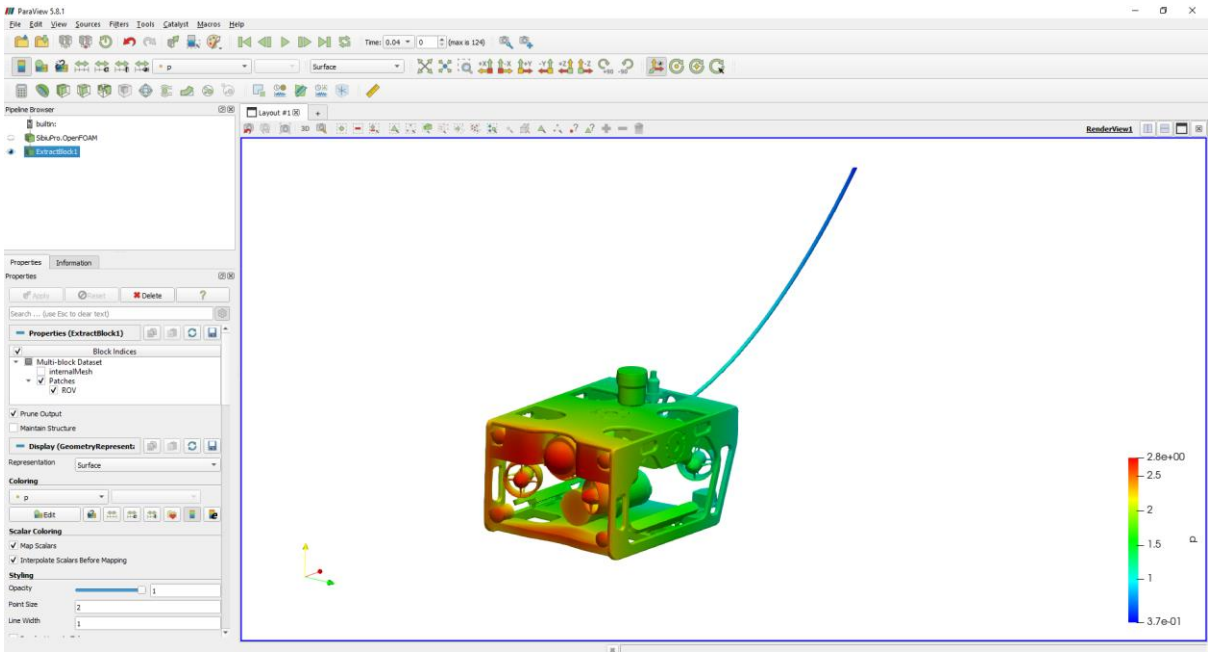


Figura 54. *ExtractBlock1*.

Como se ha explicado en el Capítulo anterior, la fuerza de arrastre actúa en el sentido del flujo, mientras que la fuerza de sustentación actúa en el sentido del flujo ortogonal vertical hacia arriba. Tal como se ve en la Figura 54, en el caso SibiuPro, el sentido del flujo es el del eje X positivo (eje rojo), mientras que el sentido ortogonal vertical hacia arriba es el del eje Y positivo (eje amarillo).

La definición analítica de fuerza total aerodinámica o hidrodinámica vista en el Capítulo anterior se corresponde con la ecuación (25).

$$F = \int_A (P - \tau_w)\, dA \tag{26}$$

Previamente a su cálculo, se debe proyectar la presión ejercida sobre la superficie del ROV para obtener la variable vectorial de presión superficial P, y para ello se necesita la dirección normal de cada cara de celda en toda la superficie del ROV.

Esto se puede conseguir aplicando el *Generate Surface Normals* sobre el módulo *ExtractBlock1*. De tal forma, queda generado el módulo *GenerateSurfaceNormals1*. Entonces se permite seleccionar *Normals* en la lista desplegable "*Coloring*" y la componente X, Y o Z en la lista desplegable "*Magnitude*", tal como se muestra en la Figura 55.

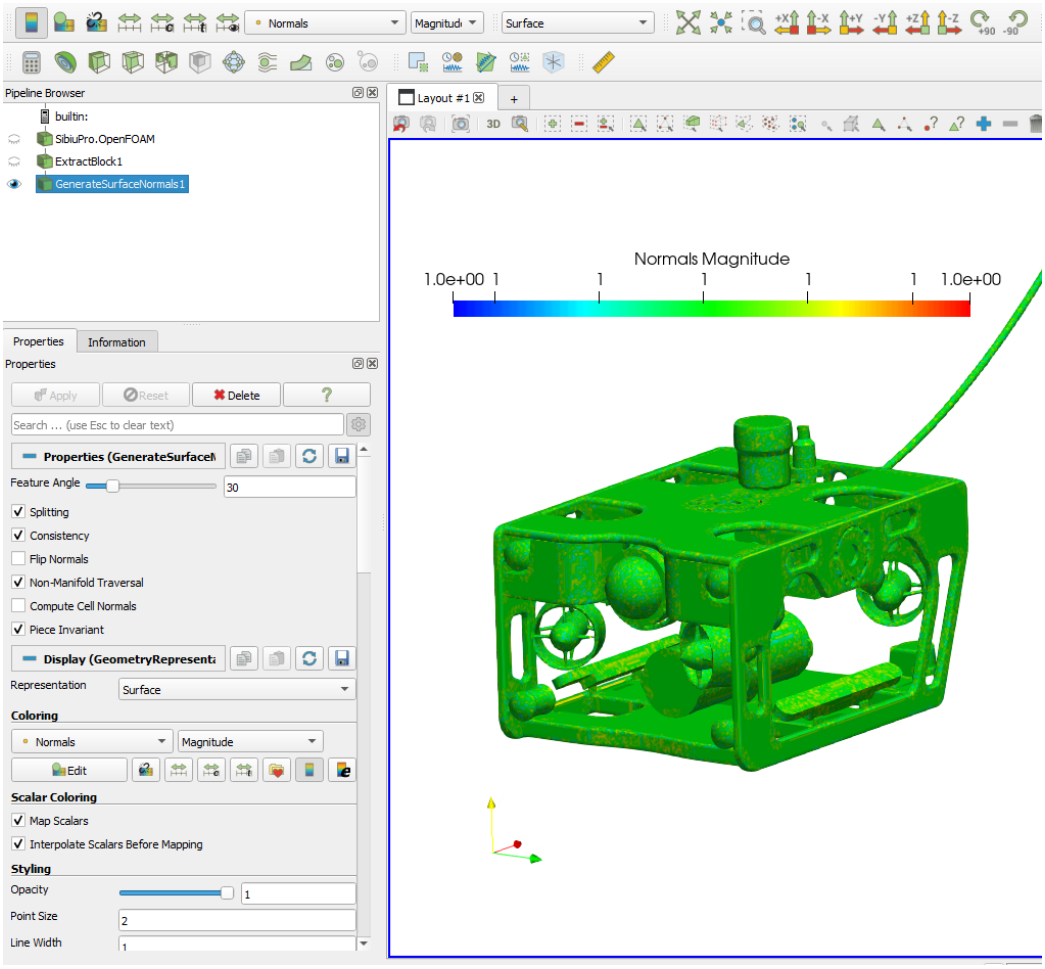


Figura 55. *GenerateSurfaceNormals1. Visualización de Normals.*

Conocida la variable *Normals* de cada celda, se puede calcular la presión superficial que se ejerce sobre cada celda mediante el filtro *Calculator*. Dentro de este filtro se selecciona se escribe el *Result Array Name* como *P* (también se puede llamar así al módulo *Calculator1*), se escribe la ecuación *p*1000*Normals* y se aplica el módulo. Se recuerda que *p* no es presión, es realmente presión/densidad del fluido. De tal forma, la presión que ejerce el fluido se calcula como p*densidad del agua = 1.000 kg/m^3.

Lo mismo que ocurre con la variable *p*, ocurre con la variable calculada por simulación y nombrada en OpenFOAM® como *wallShearStress*. Esta variable se corresponde con una tensión dividida por la densidad del fluido, igual que *p*. De tal forma, se aplica otro filtro *Calculator* para obtener la tensión en unidades de presión. Dentro del filtro se selecciona se escribe el *Result Array Name* como *WSS* y se escribe la ecuación *wallShearStress*1000* sin multiplicar por ningún vector ya que *wallShearStress* ya es una variable vectorial tangente a la superficie. Así τ_w queda calculado y de nombre *WSS*.

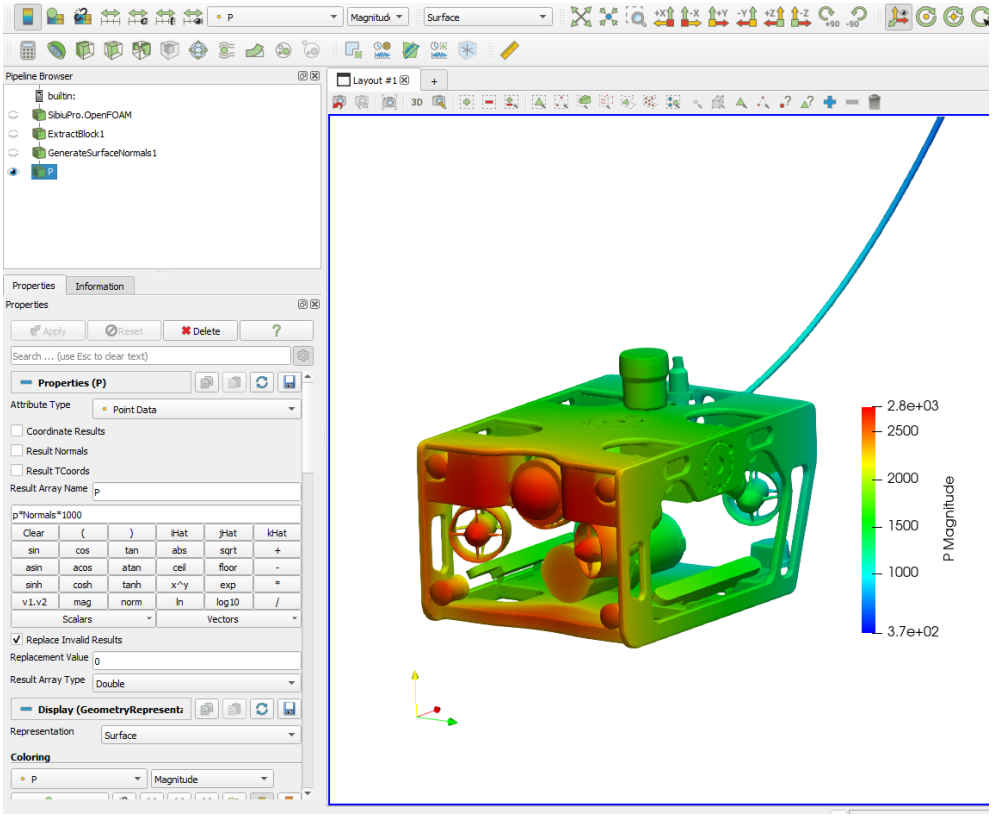


Figura 56. *Calculator para P.*

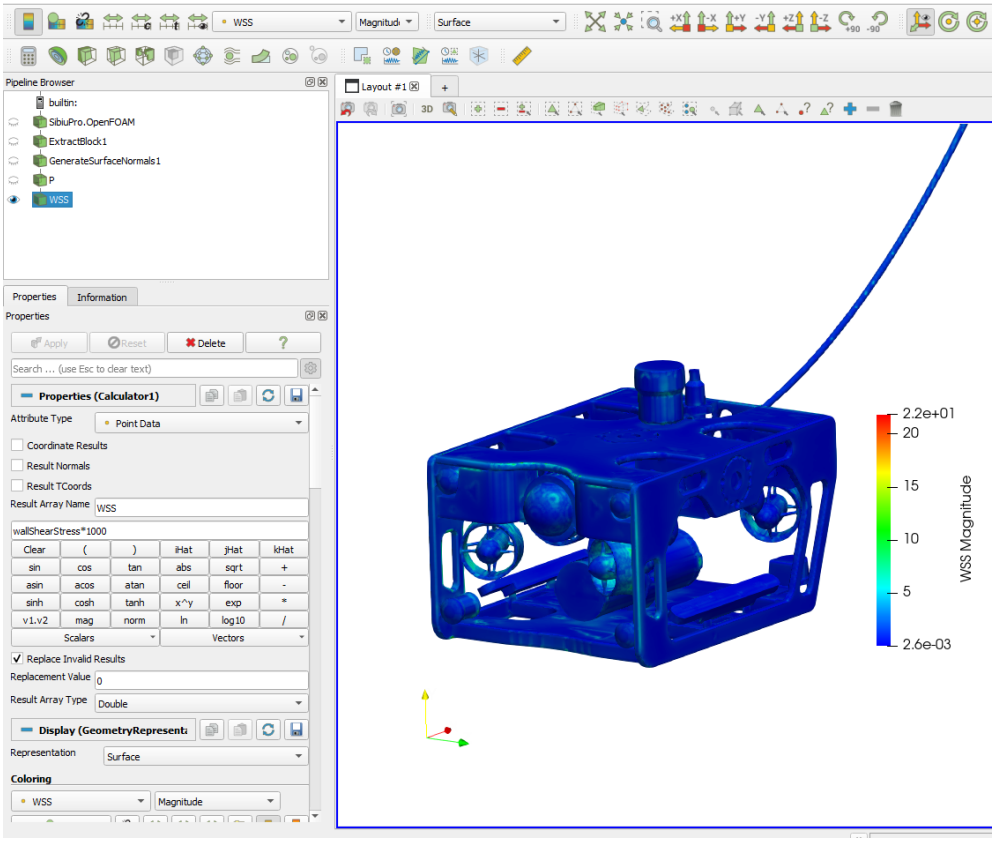


Figura 57. *Calculator para WSS.*

A partir de ambas variables se puede calcular la presión hidrodinámica total que se ejerce sobre cada cara de la celda del ROV. Para ello se crea un nuevo filtro *Calculator* donde se escribe el *Result Array Name* como *P-WSS* y la ecuación *P-WSS*, de igual forma que para la expresión analítica del mismo.

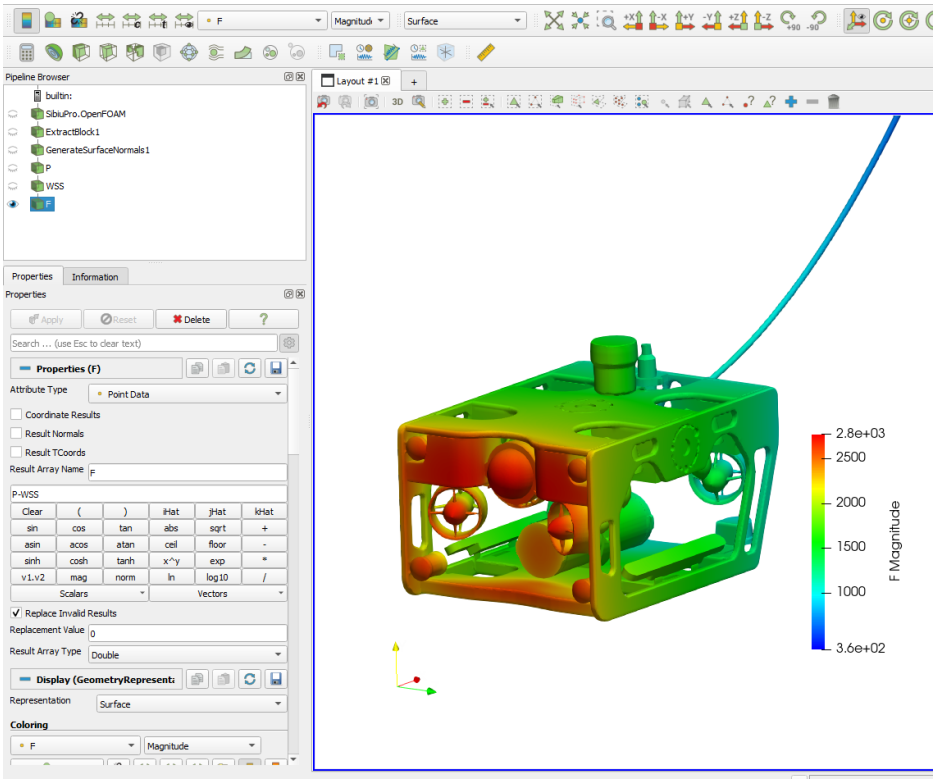


Figura 58. *Calculator para P-WSS.*

Para visualizar cada componente de la presión hidrodinámica total sobre cada cara del ROV, se puede seleccionar cada dirección mediante la lista desplegable de *Magnitude*.

El último paso consiste en integrar la presión hidrodinámica total *P-WSS* a lo largo de toda la superficie del ROV de 275.216 caras de celdas, de igual manera que en la definición analítica del mismo. Para ello se aplica el filtro *Integrate Variables*, generando el módulo *IntegrateVariables1*. Entonces aparece la siguiente ventana de resultados de ParaView® donde se puede filtrar los valores de las variables mediante el botón .

En la Figura 59 se muestra la integral a lo largo de toda la superficie del ROV de *P-WSS*, *P* y *WSS*, es decir, la fuerza hidrodinámica total *F*, la fuerza de presión *PF* y la fuerza de fricción *FF* respectivamente con las tres componentes de cada una. Cada una de estas componentes se analizarán en el Capítulo siguiente.

Showing IntegrateVariables1 Attribute: Point Data Precision: 6

	Point ID	P-WSS			P			WSS		
0	0	96.5683	-9.23439	-1.12868	95.584	-9.15957	-1.11907	-0.98428	0.0748166	0.00960756

Figura 59. *IntegrateVariables1 en el instante 0.04 segundos.*

La fuerza hidrodinámica total F da un valor de 96,5683 N para su componente X y de -9,23439 N para su componente Y. Estas componentes son la fuerza de arrastre y sustentación respectivamente. La componente Z se corresponde con la fuerza de desvío sobre el ROV, cuyo valor es despreciable con respecto a las otras componentes. Este cálculo es el correspondiente para el instante 0,04 s. En el instante 5 s, los resultados obtenidos son los siguientes:

Showing IntegrateVariables1 Attribute: Point Data Precision: 6

	Point ID	P-WSS			P			WSS		
0	0	50.5828	-7.02683	1.97871	49.2621	-7.06628	1.99603	-1.32069	-0.0394555	0.0173215

Figura 60. *IntegrateVariables1 en el instante 5 segundos.*

De la tabla de la Figura 60 se saca un valor de 50,5828 N para fuerza de arrastre y de -7,02683 N para la fuerza de sustentación en el instante 5 segundos. A partir del valor medio en el tiempo de ambas fuerzas, se puede obtener el valor de los coeficientes hidrodinámicos despejando las ecuaciones de fuerzas hidrodinámicas.

Coeficiente de arrastre: $$C_D = \frac{2\,|F_D|}{\rho\, A_D\, U^2} \quad (27)$$

Coeficiente de sustentación: $$C_L = \frac{2\,|F_L|}{\rho\, A_L\, U^2} \quad (28)$$

De ambas fórmulas se conocen todos los parámetros menos el área transversal del vehículo que interactúa con el fluido en el plano ortogonal a la velocidad A_D y el área transversal del vehículo que interactúa con el fluido en el plano horizontal paralelo a la velocidad A_L (geometría no afilada).

Ambas áreas se pueden calcular volviendo al modelado del ROV en formato *.prt* de SolidWorks® y aplicando vistas de sección en las secciones de mayor área A_D y A_L, tal como se muestra en las Figuras 61 y 62.

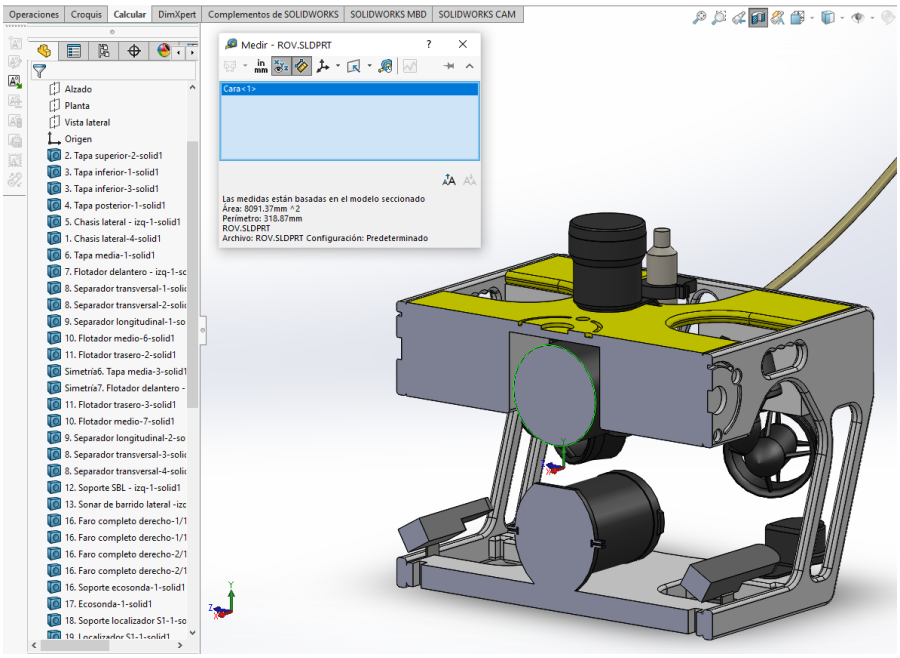


Figura 61. *Mayor sección de arrastre.*

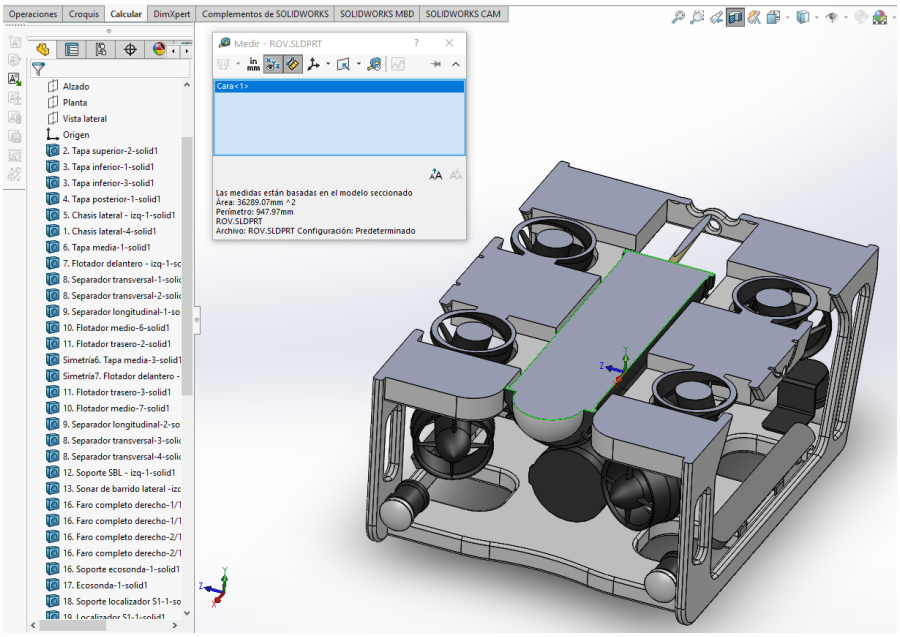


Figura 62. *Mayor sección de sustentación.*

Mediante la herramienta medir de SolidWorks® se recopila cada pequeña área generada por la vista de sección. Sumando estas áreas se concluye un área transversal de arrastre de 0,05479 m², y un área transversal de sustentación de 0,014497 m².

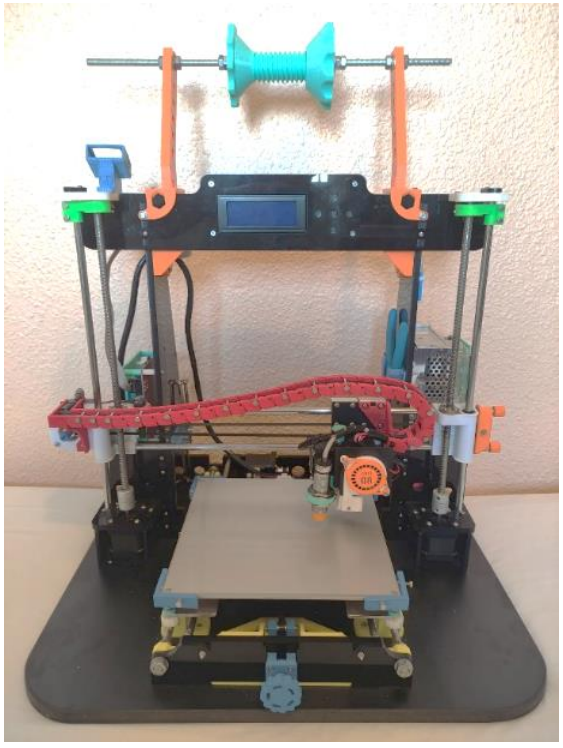
Con estos datos se puede entonces calcular los coeficientes hidrodinámicos, cuyo resultado y análisis se muestra en el siguiente Capítulo.

4.4. Procedimiento experimental

En este apartado se describen los materiales y la metodología experimental seguida para generar las líneas de flujo verticales centradas de interacción entre el fluido y el ROV Sibiu Pro de manera empírica. El alcance del mismo no recoge el cálculo de magnitudes cuantitativas como las fuerzas hidrodinámicas, que podrían plantearse en futuros trabajos.

4.4.1. Materiales

Tabla 2. *Lista de materiales para el procedimiento experimental.*

ID	NOMBRE	FOTO	UDS
1.	Canal de flujo (Laboratorio de Mecánica de Fluidos, ETSI, Universidad de Huelva)		1
2.	Impresora 3D (personal; modelo Anet A8 mejorada)		1

3.	1 kg de filamento PLA		1
4.	Agujas hipodérmicas		13
5.	Jeringuilla de 20 ml		1
6.	Permanganato potásico		1
7.	Tornillo de rosca madera 3.5x16		1
8.	Tornillo de rosca madera 2.5x12		3
9.	Tornillo de rosca madera 2.5x10		11
10.	Pegamento instantáneo		1
11.	Cutter		1

12.	Pinzas de impresora 3D		1
13.	Pistola de silicona		1
14.	Barra de silicona		1
15.	Probeta		1
16.	Suero de 1 l		1
17.	Gotero		1
18.	Papel de lija		1
19.	Destornillador de estrella		1

4.4.2. Diseño e impresión 3D del colector

Para la generación de las líneas de corriente verticales se usará una disolución de permanganato potásico ($KMnO_4$). La idea consiste básicamente en que este fluido colorante se divida uniformemente mediante un colector descrito a lo largo de este apartado, diseñado en SolidWorks® e impreso en 3D. La interacción del flujo colorante con el flujo de agua del canal permitirá visualizar las líneas de corriente.

Tal como se muestra en la Figura 63, el diseño del colector consta de dos piezas pegadas mediante pegamento instantáneo: colector frontal y colector trasero. Las medidas generales de ambas piezas se muestran en el plano de piezas del colector del Anexo VII.

Interiormente, el colector trasero cuenta con un conducto cilíndrico superior por donde se introduce el fluido colorante, pasando posteriormente a un conducto de sección cuadrada que divide el flujo hacia el colector frontal. La función del colector frontal consiste en dividir el fluido colorante en 12 flujos separados en altura cada 1 cm mediante huecos donde se introducen 12 agujas hipodérmicas para uniformizar la separación. Para generar los huecos se ha modelado en SolidWorks® el sólido de este colector restado con moldes de aguja de cierta expansión radial para que las agujas posteriormente entren adecuadamente pero sin holguras, y pegadas de igual forma.

Figura 63. *Colector trasero (izq) y colector frontal (drch) en SolidWorks®.*

El diseño exterior del conjunto colector está pensando hidrodinámicamente para que produzca la mínima interacción con el flujo del agua del canal, con una sección horizontal triangular aguda doble. Cuenta con una base circular para que se mantenga de pie en el canal y se garantice un flujo horizontal recto. Más detalles del mismo se muestran en el plano de piezas del colector del Anexo VII.

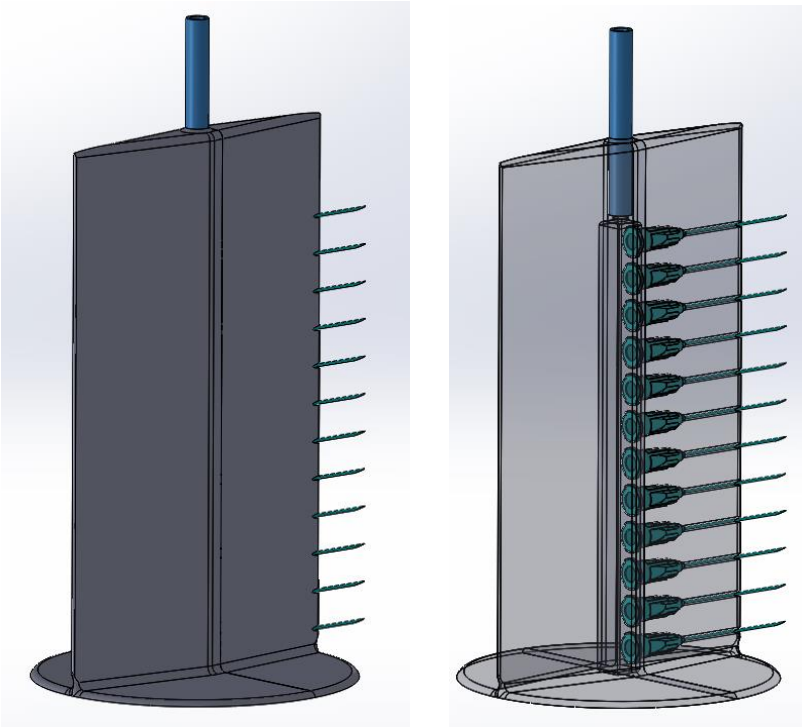


Figura 64. *Conjunto colector en SolidWorks®.*

Para la impresión en 3D del mismo se utiliza el software de generación del archivo *.gcode* llamado Ultimaker Cura®. En este programa, la orientación de ambas piezas se dispone para que la primera capa se corresponda con la cara de pegado de las mismas. Esta orientación no necesita soportes para el colector frontal, aunque si para el colector trasero.

Entre las configuraciones generales que se deben introducir en el programa, se establece una altura de capa de 0,2 mm y un relleno del 100% para que no se formen huecos por donde se acumulen bolsas de aire y se filtre el colorante. Para evitar el *warping* (despegue por retracción de las capas al enfriar) que posiblemente se produzca, es importante especificar una temperatura de la cama de al menos 60 ºC y un borde de al menos 8 mm. Mediante el borde se consigue aumentar la superficie de adhesión de la primera capa, especialmente en zonas conflictivas como en la base del colector. Se puede reforzar aún más la correcta adhesión de esta capa si se aplica laca previamente sobre la superficie caliente de la cama.

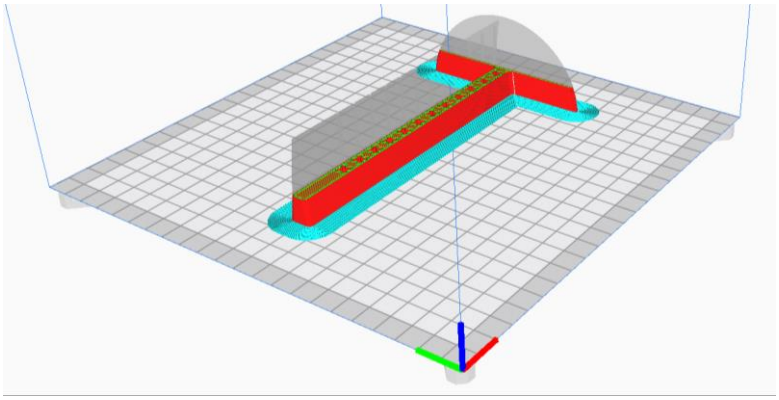


Figura 65. *Colector frontal en Ultimaker Cura®.*

Una vez se impriman ambas piezas, se deben retirar los bordes, soportes y posibles hilos que se hayan generado mediante cutter y pinzas. Además es recomendable realizar un pequeño lijado sobre las caras a pegar y sus bordes. Posteriormente, se introducen y pegan cada una de las agujas, con cuidado de no dañar las mismas.

El proceso de pegado de ambas piezas del colector se debe realizar con el máximo cuidado para que posteriormente no se produzcan filtraciones del colorante. Una vez aplicado el pegamento sobre las caras a pegar, se ejerce una ligera presión con los dedos tratando de no desplazar las piezas mientras el pegamento se seca durante al menos 2 minutos. Por último, es recomendable reforzar los bordes con más pegamento. El resultado se muestra en la Figura 66.

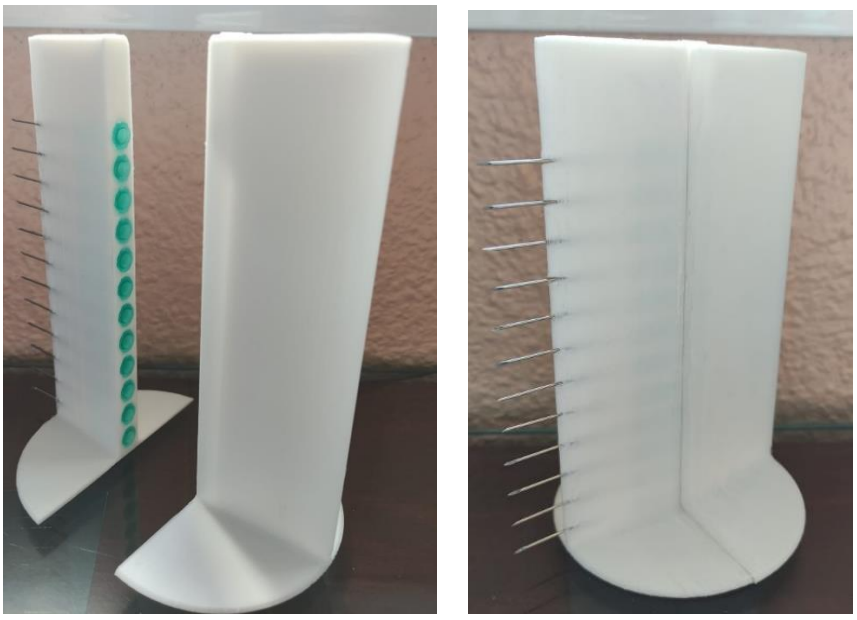


Figura 66. *Conjunto colector impreso en 3D con filamento PLA.*

4.4.3. Empaquetado, modificación e impresión 3D del ROV

La impresión 3D del ROV a escala cuenta con la limitación de que el ancho del canal de flujo del laboratorio es de 6 cm. Es por ello que la escala a la cual se imprima el ROV debe contar con un ancho total de no mucho más de 5,5 cm. Teniendo en cuenta que el ROV a escala real cuenta con un ancho de 39 cm, el ROV impreso debe ser un 14,1% el real, es decir, una escala aproximada de 7/50.

Para la impresión de 3D se realizaron diversas pruebas. La primera consiste en imprimir el ROV tal cual fue modelado y ensamblado en SolidWorks®, utilizando como primera capa la cara frontal de mismo, y generando los soportes que fuesen necesarios para asegurar la calidad de impresión de los detalles. La problemática está en la complejidad de retirar los soportes sin romper las piezas desfavorables del ROV.

La idea de la segunda prueba plantea el pegado de grupos de piezas mediante pegamento rápido. Los conjuntos de piezas se establecen para que se imprima la primera capa sobre caras favorables que generen el mínimo o ningún soporte, y así no se comprometan los detalles. El problema surge en el pegado de ciertos grupos de piezas que cuentan con una superficie muy pequeña y de difícil agarre para ejercer la correcta presión de pegado. Esto es debido a la complejidad geométrica del propio ROV y el tamaño tan reducido de su escala.

La última idea y final consiste nuevamente en imprimir grupos de piezas atornillados mediante tornillos pequeños de rosca de madera. Ello implica la necesidad de modelar en SolidWorks® huecos cilíndricos donde se introduzcan los tornillos. Para no modificar las piezas ya modeladas del ensamblaje del ROV real lo que se hace es empaquetar este ensamblaje con sus piezas.

Empaquetar es una herramienta que tiene SolidWorks® que copia el directorio del ensamblaje con sus piezas y modifica sus nombres para que el nuevo ensamblaje este referenciado a las piezas nuevas de distinto nombre. De esta forma se tienen dos ensamblajes del ROV iguales pero referenciados a distintas piezas, permitiéndose modificar las nuevas piezas sin afectar al ensamblaje original.

En este punto se especifican los grupos de piezas mediante estados de visualización y se modifican las piezas para el atornillado entre grupos, teniendo en cuenta la cara de primera capa de impresión y minimizando la generación de soportes.

Debido además al nivel de detalle de las piezas del ensamblaje y la dificultad para su impresión a una escala tan reducida (por ejemplo el propulsor), otras piezas también se han modificado ligeramente para que se imprima con buena calidad y se atornille correctamente.

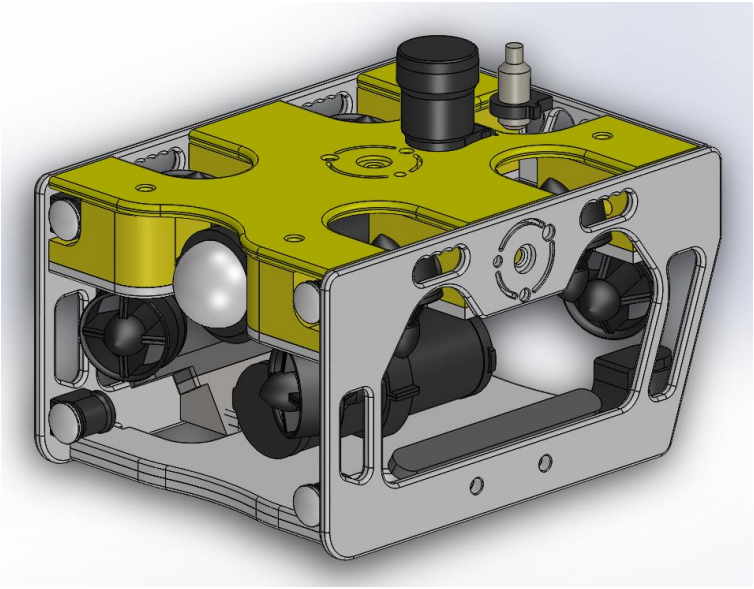


Figura 67. *Ensamblaje del ROV a imprimir.*

En la Figura 68 se muestran los 8 grupos de piezas pensados, donde los tres grupos superiores se deben imprimir tal cual y simétricos.

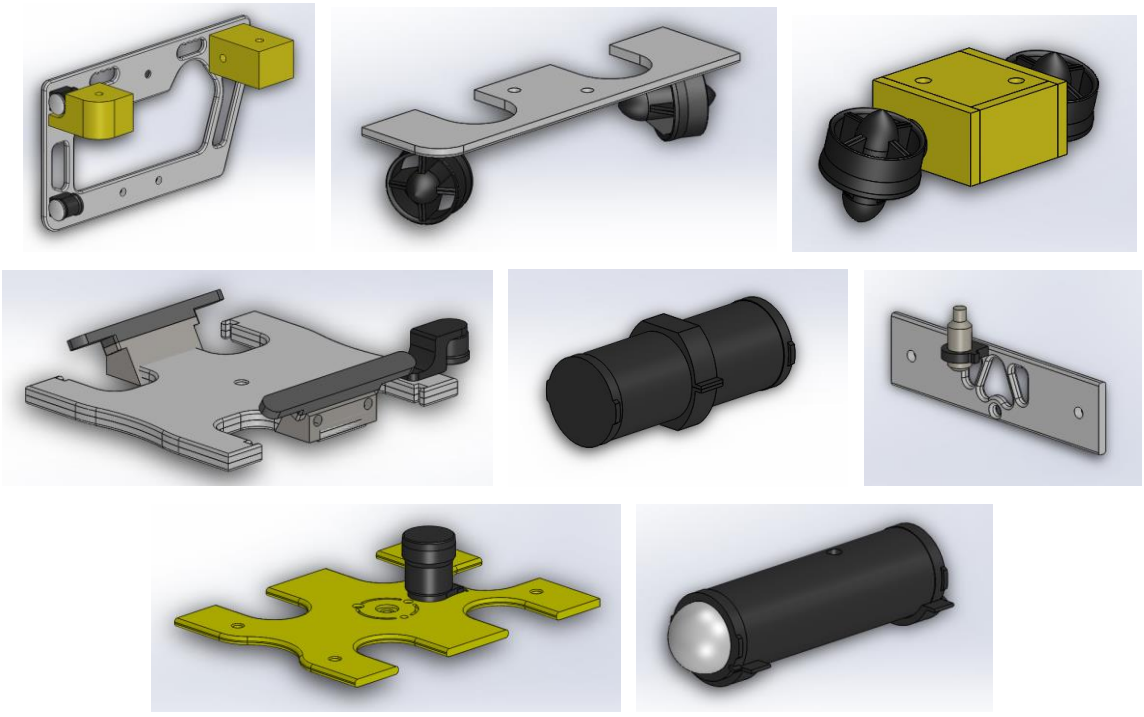


Figura 68. *Ocho grupos de piezas del ROV a imprimir.*

Para generar cada archivo *.stl* se guardan los estados de visualización de cada grupo de piezas como *.sldprt*. En cada uno de estos archivos guardados se combina el sólido y se guarda el *.stl*, tal como se mostró en apartados anteriores.

En Ultimaker Cura® primeramente se escalan las piezas un 14,1%, se establece una cara favorable de primera capa y se genera la pieza simétrica según el caso.

Entre las configuraciones generales, igual que para el colector, se establece una altura de capa de 0,2 mm y un relleno de 100% para que no se existan huecos por donde se acumulen bolsas de aire y se filtre el colorante. Dependiendo del grupo de piezas, será necesario generar soportes o introducir bordes.

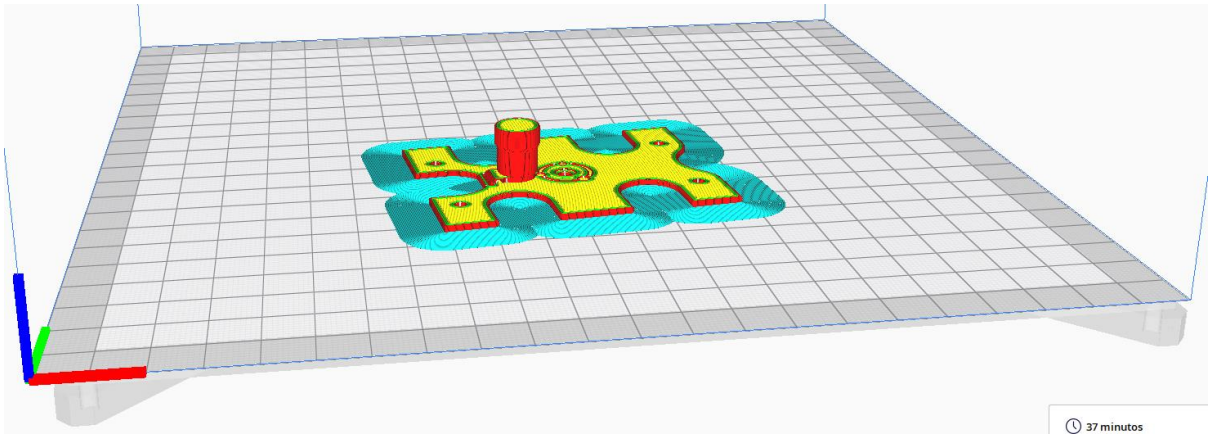


Figura 69. *Grupo superior en Ultimaker Cura®.*

De igual manera que para el colector, se deben retirar los bordes, soportes y posibles hilos que se hayan generado mediante cutter, pinzas y lija.

Por último solo faltaría atornillar cada pieza en su lugar. Para este caso se ha utilizado un único tornillo de 3.5 para agarrar el cilindro de electrónica con la tapa superior. El resto de uniones se realizan con tornillos de 2.5x10 o 2.5x12 dependiendo de cuanta profundidad de agarre sea necesaria. Es posible que ciertos tornillos entren con cierta dificultad. Bastaría en estos casos con abrir ligeramente los huecos mediante punzón o taladro de broca fina para posteriormente atornillar adecuadamente. El resultado final se muestra en la Figura 70.

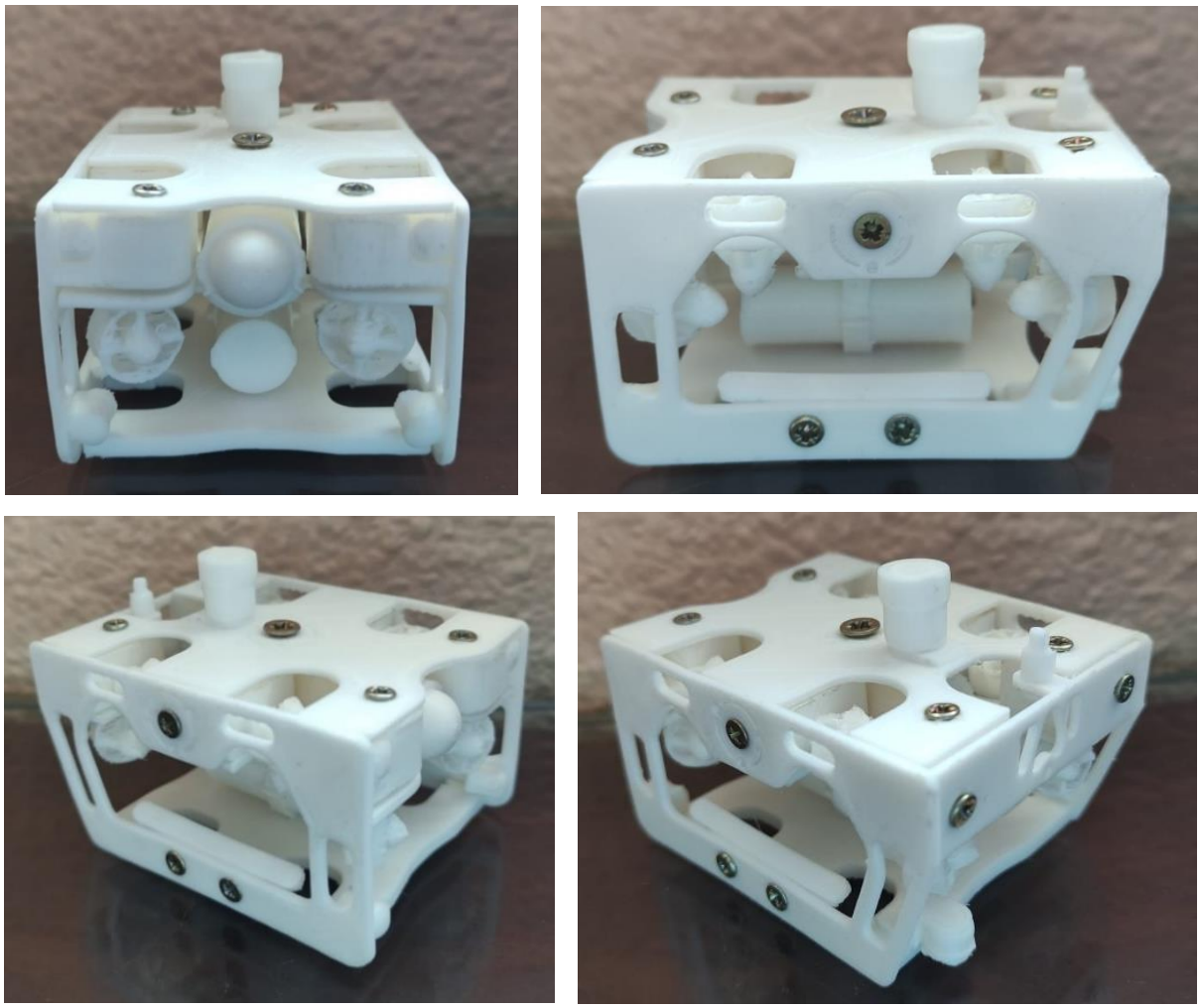


Figura 70. *ROV a escala impreso en 3D con filamento PLA.*

4.4.4. Descripción del ensayo experimental

El procedimiento experimental comienza una vez se dispone de todos los materiales y se ha fabricado el colector y el ROV a escala. En este apartado se describen cada uno de los pasos a seguir en el laboratorio de Mecánica de Fluidos de la ETSI de la Universidad de Huelva.

1. Se llena el tanque del canal y se comienza a recircular el flujo.

Es importante asegurarse de que el ciclo del canal este correctamente conectado y cebar la bomba antes de encender el motor.

El agua del canal debe alcanzar una altura de al menos 15 cm.

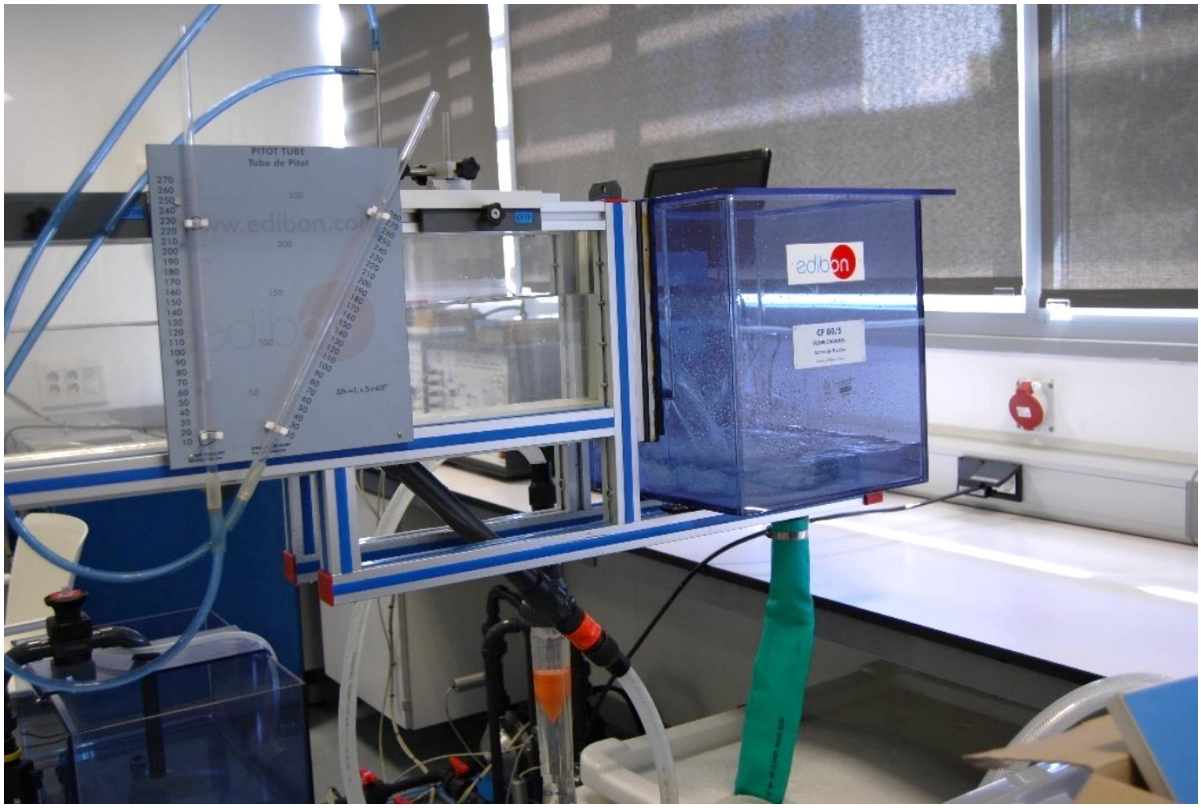


Figura 71. *Llenado del canal de flujo.*

2. Preparación del colorante.

Se llena un matraz de un litro con agua.

A continuación se vierte media cucharada de permanganato potásico.

Al instante el agua adquiere un color violeta intenso.

Este líquido se debe manejar con cuidado, ya que mancha la piel y la ropa.

Figura 72. *Probeta de $KMnO_4$ diluido.*

3. Llenado de colorante.

Se llena la jeringuilla de colorante.

Se coloca la aguja hipodérmica en la jeringuilla.

Se pincha la aguja en la bolsa de suero y se llena de colorante.

Se repite el proceso hasta que la bolsa adquiera un color intenso.

Figura 73. *Llenado de la bolsa suero con colorante.*

4. Conexionado del gotero.

Se conecta el extremo superior del gotero con el regulador cerrado a la bolsa de colorante.

Se conecta el externo inferior del gotero al colector.

Para asegurar que en la conexión inferior no se producen fugas se pega el mismo mediante silicona caliente.

Figura 74. *Pegado de la conexión del gotero al colector.*

5. Comprobación de las conexiones y el colector.

Se introduce el colector en un lavabo.

Se abre el regulador del gotero para dejar pasar el colorante

Se comprueba las conexiones, el colector y el flujo a través de las agujas.

Si existen fugas, se sellan mediante silicona caliente.

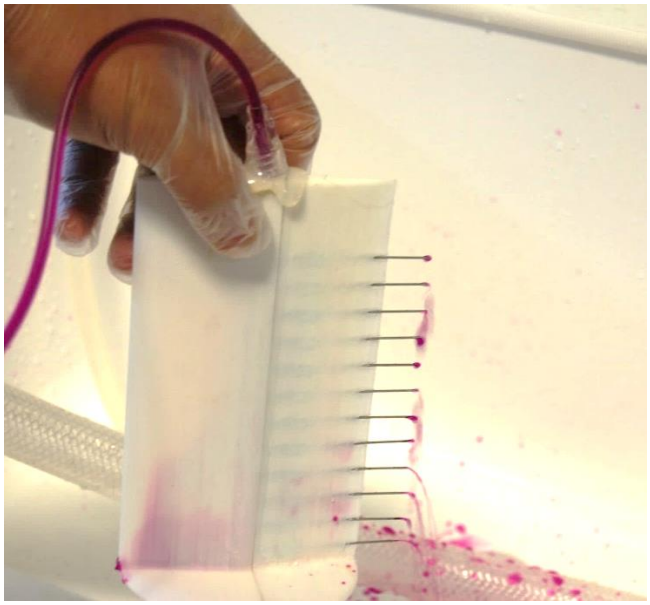


Figura 75. *Comprobación de las conexiones y el colector.*

6. Colocación del ROV a escala.

El ROV debe quedar fijo a una altura media dentro del canal de flujo.

Para ello, rudimentariamente se ha utilizado una pinza y un pequeño trozo de goma espuma que la abre. De tal forma, el ROV queda comprimido por uno de los extremos de esta pinza y el cristal del canal a una altura media.

Además se colocado una lámina de goma espuma en la cara exterior del cristal para que posteriormente se produzca contraste entre el blanco de la goma espuma y el colorante violeta.

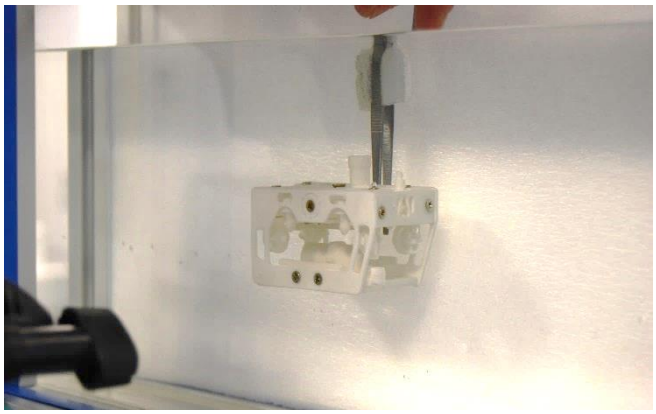


Figura 76. *Colocación del ROV a escala.*

7. Colocación del depósito de abastecimiento de colorante y el colector.

La bolsa se coloca enganchada al canal a una altura superior al colector para que el colorante caiga por gravedad

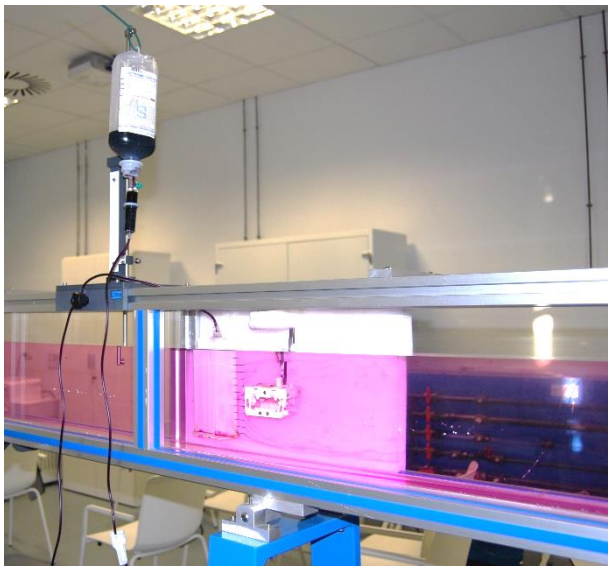


Figura 77. *Colocación del depósito de abastecimiento de colorante.*

Para colocar el colector, surge el problema de que cuando se diseñó la base del mismo no se tuvo en cuenta que el diámetro no puede ser superior a 6 cm (ancho del canal). Como la base diseñada tiene un diámetro de 8,6 cm, se optó por usar las pinzas para cortar 1,3 cm cada lateral de la base. Además se sella la parte cortada mediante silicona caliente para evitar posibles fugas.

De tal forma se introduce el colector en el canal de flujo quedando fijo y vertical a una altura media. Nótese el corte de la base en la Figura 78.

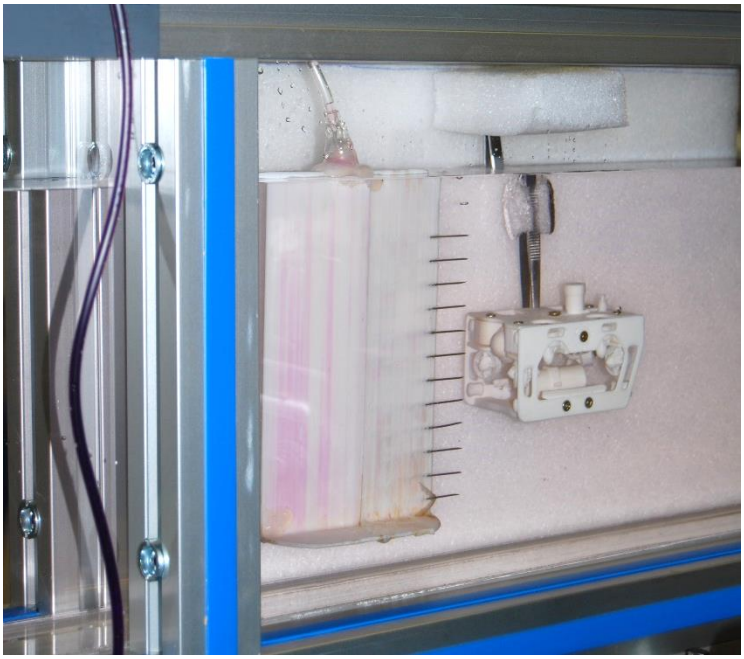


Figura 78. *Colocación del colector.*

8. Ajuste de velocidad del flujo y regulación del colorante

En este punto se prueban las líneas de flujo.

Las diversas pruebas realizadas confirman que se debe trabajar con una velocidad del flujo mínima para que se garantice la uniformidad de las líneas. Para ello se debe minimizar la diferencia de altura del canal y las revoluciones del motor. También se coloca una compuerta que ralentiza el flujo a continuación del ROV.

Adicionalmente se debe regular la cantidad de colorante entrante mediante el gotero. En este existe además la posibilidad de introducir el colorante a presión por inyección mediante jeringuilla y aguja.

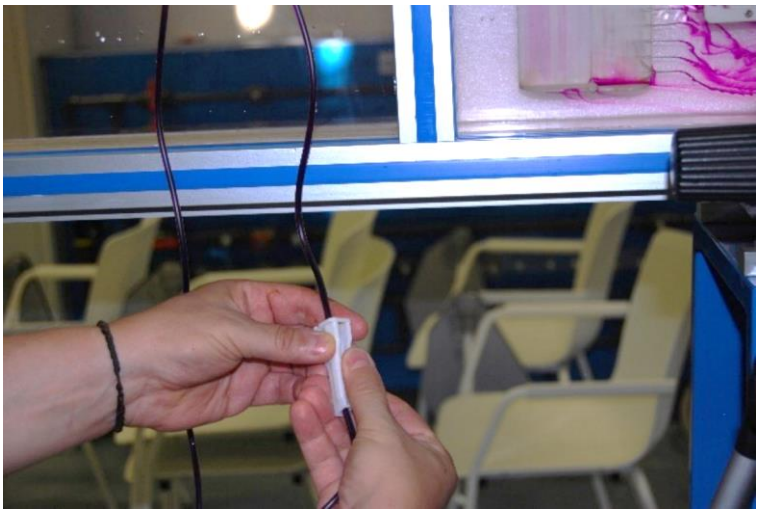


Figura 79. *Regulación del colorante.*

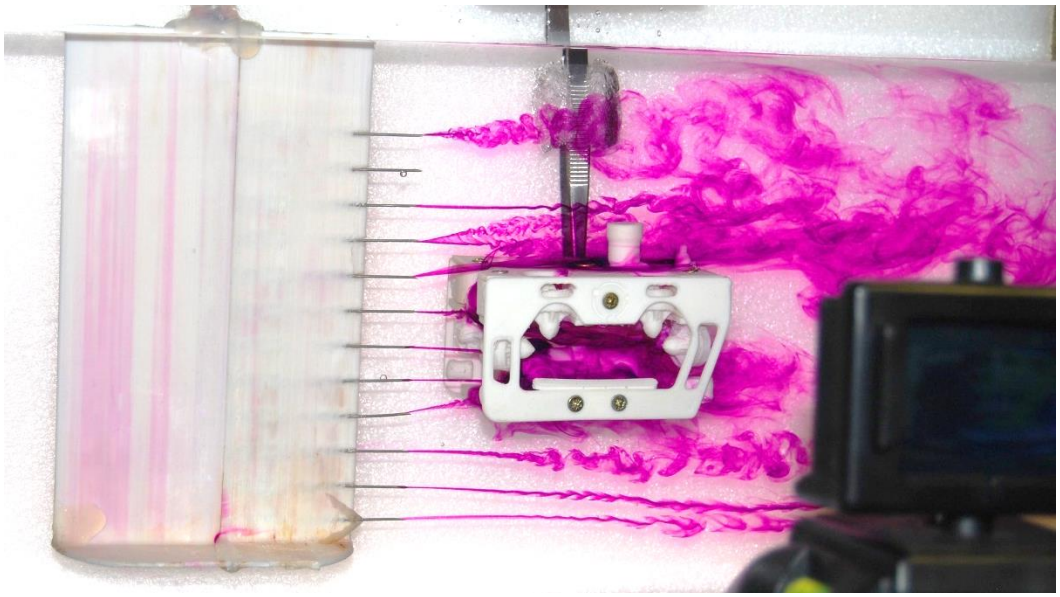


Figura 80. *Prueba de líneas de flujo.*

La interpretación de los resultados empíricos se realiza en el siguiente Capítulo.

Capítulo 5
Resultados y discusión

En este Capítulo se analizan los resultados hidrodinámicos obtenidos para el ROV Sibiu Pro de simulación CFD mediante OpenFOAM® y ParaView®, y experimentales en el laboratorio de Mecánica de Fluidos de la ETSI de la Universidad de Huelva.

Por una parte se analizan las componentes de las fuerzas hidrodinámicas que el fluido a 1 m/s ejerce sobre el ROV, obteniendo posteriormente los coeficientes hidrodinámicos. Para contrastar los resultados se comparan con otro trabajo similar.

A continuación se examina mediante capturas de imágenes y animaciones los resultados postprocesados de la interacción del movimiento de avance del ROV con respecto al flujo y viceversa.

Por último se valoran y comparan los resultados experimentales con los resultados simulados, comprobando la semejanza de interacción flujo-ROV entre ambos procedimientos.

5.1. Fuerzas y coeficientes hidrodinámicos obtenidos

A partir del procedimiento de postprocesado seguido en el Capítulo anterior se obtiene la Tabla 3 de resultados de los vectores de fuerza de presión, fuerza de fricción y fuerza total hidrodinámica, con cada una de sus componentes, a lo largo de los 125 instantes de tiempo de simulación.

Tabla 3. *Fuerza de presión PF, Fuerza de fricción FF y Fuerza hidrodinámica F.*

Step (s)	Pressure Force (N)			Friction Force (N)			Hydrodynamic Force (N)		
	PFx	PFy	PFz	FFx	FFy	FFz	Drag	Lift	Side
0,04	95,5840	-9,15957	-1,119070	-0,9843	0,07482	0,009610	**96,57**	**-9,23**	-1,13
0,08	45,3344	-4,12641	-0,731488	-0,1561	0,01408	0,003157	**45,49**	**-4,14**	-0,73
0,12	42,6497	2,37853	1,379390	-0,3090	-0,01709	-0,010840	**42,96**	**2,40**	1,39
0,16	46,4886	3,22067	1,817380	-0,4312	-0,04048	-0,019070	**46,92**	**3,26**	1,84
0,2	47,6029	1,83276	0,967791	-0,5159	-0,04256	-0,017571	**48,12**	**1,88**	0,99
0,24	47,4789	-0,74737	0,061520	-0,5837	-0,03442	-0,007524	**48,06**	**-0,71**	0,07
0,28	47,4706	-3,50253	-0,057283	-0,6318	-0,01924	-0,002205	**48,10**	**-3,48**	-0,06
0,32	47,7384	-6,62834	0,327015	-0,6868	0,00076	-0,003905	**48,43**	**-6,63**	0,33
0,36	47,6146	-8,49215	0,450633	-0,7469	0,00791	-0,004202	**48,36**	**-8,50**	0,45
0,4	46,9993	-8,51277	0,355590	-0,8019	0,01179	-0,003485	**47,80**	**-8,52**	0,36
0,44	46,6943	-7,62288	0,415747	-0,8491	0,01554	-0,005984	**47,54**	**-7,64**	0,42
0,48	46,9472	-7,07458	0,702598	-0,8903	0,01712	-0,007639	**47,84**	**-7,09**	0,71
0,52	47,3995	-7,65479	0,802980	-0,9297	0,01741	-0,003058	**48,33**	**-7,67**	0,81
0,56	47,8762	-9,27982	0,567936	-0,9707	0,01867	0,006455	**48,85**	**-9,30**	0,56
0,6	48,2610	-11,0995	0,373964	-1,0126	0,01550	0,014735	**49,27**	**-11,12**	0,36
0,64	48,4799	-11,7890	0,497980	-1,0519	0,00990	0,018278	**49,53**	**-11,80**	0,48
0,68	48,5162	-10,5010	0,727252	-1,0874	0,00480	0,018857	**49,60**	**-10,51**	0,71
0,72	48,4571	-8,24472	0,911234	-1,1153	0,00357	0,018872	**49,57**	**-8,25**	0,89
0,76	48,3336	-6,47580	1,077440	-1,1351	0,00962	0,018020	**49,47**	**-6,49**	1,06
0,8	48,1898	-5,81507	1,070030	-1,1539	0,01623	0,014970	**49,34**	**-5,83**	1,06
0,84	48,1564	-6,00102	0,914028	-1,1761	0,01681	0,009335	**49,33**	**-6,02**	0,90
0,88	48,2143	-6,33130	0,804884	-1,1943	0,01113	0,007354	**49,41**	**-6,34**	0,80
0,92	48,2568	-6,66643	0,774037	-1,2078	0,00326	0,009322	**49,46**	**-6,67**	0,76
0,96	48,2024	-7,19700	0,816269	-1,2208	-0,00748	0,013159	**49,42**	**-7,19**	0,80
1	48,0911	-7,67419	0,797871	-1,2377	-0,01955	0,017594	**49,33**	**-7,65**	0,78
1,04	48,0684	-7,68095	0,630120	-1,2559	-0,02795	0,019167	**49,32**	**-7,65**	0,61
1,08	48,1006	-7,15272	0,526601	-1,2650	-0,02956	0,018021	**49,37**	**-7,12**	0,51
1,12	48,0012	-6,53453	0,648443	-1,2630	-0,02656	0,017267	**49,26**	**-6,51**	0,63
1,16	47,7265	-6,23169	0,917052	-1,2546	-0,02213	0,019018	**48,98**	**-6,21**	0,90
1,2	47,4506	-6,35367	1,125490	-1,2492	-0,01687	0,021090	**48,70**	**-6,34**	1,10
1,24	47,3018	-6,72155	1,214210	-1,2498	-0,01480	0,022740	**48,55**	**-6,71**	1,19
1,28	47,3035	-7,16957	1,281120	-1,2543	-0,01802	0,023660	**48,56**	**-7,15**	1,26
1,32	47,4069	-7,54463	1,389080	-1,2624	-0,02402	0,024330	**48,67**	**-7,52**	1,36
1,36	47,5280	-7,56402	1,564230	-1,2729	-0,03099	0,024680	**48,80**	**-7,53**	1,54

1,4	47,6503	-7,09284	1,770230	-1,2841	-0,03623	0,024060	**48,93**	**-7,06**	1,75
1,44	47,7455	-6,26721	1,933730	-1,2937	-0,03866	0,023630	**49,04**	**-6,23**	1,91
1,48	47,7353	-5,53828	2,026730	-1,2983	-0,03911	0,024050	**49,03**	**-5,50**	2,00
1,52	47,5980	-5,28404	2,040860	-1,2995	-0,03934	0,025510	**48,90**	**-5,24**	2,02
1,56	47,4589	-5,56038	1,955870	-1,2996	-0,04043	0,027790	**48,76**	**-5,52**	1,93
1,6	47,4836	-6,14495	1,809990	-1,2988	-0,04295	0,029380	**48,78**	**-6,10**	1,78
1,64	47,6637	-6,73213	1,678100	-1,2984	-0,04699	0,029350	**48,96**	**-6,69**	1,65
1,68	47,9047	-7,17454	1,614670	-1,3006	-0,05044	0,027690	**49,21**	**-7,12**	1,59
1,72	48,1422	-7,39188	1,631900	-1,3066	-0,04992	0,025080	**49,45**	**-7,34**	1,61
1,76	48,3130	-7,23625	1,686830	-1,3145	-0,04449	0,022580	**49,63**	**-7,19**	1,66
1,8	48,3141	-6,70168	1,745130	-1,3195	-0,03734	0,020940	**49,63**	**-6,66**	1,72
1,84	48,1247	-6,05839	1,791400	-1,3176	-0,03091	0,020860	**49,44**	**-6,03**	1,77
1,88	47,8591	-5,67067	1,818990	-1,3102	-0,02763	0,021990	**49,17**	**-5,64**	1,80
1,92	47,6717	-5,64888	1,789570	-1,3012	-0,02792	0,023180	**48,97**	**-5,62**	1,77
1,96	47,6567	-5,87476	1,695760	-1,2939	-0,03086	0,024100	**48,95**	**-5,84**	1,67
2	47,8110	-6,29989	1,589510	-1,2899	-0,03521	0,024330	**49,10**	**-6,26**	1,57
2,04	48,0766	-6,93281	1,536670	-1,2889	-0,03958	0,023930	**49,37**	**-6,89**	1,51
2,08	48,3625	-7,51919	1,577930	-1,2933	-0,04287	0,022870	**49,66**	**-7,48**	1,56
2,12	48,5843	-7,62905	1,691220	-1,3024	-0,04276	0,021900	**49,89**	**-7,59**	1,67
2,16	48,6375	-7,11378	1,801520	-1,3082	-0,03977	0,022080	**49,95**	**-7,07**	1,78
2,2	48,4501	-6,26565	1,836940	-1,3052	-0,03604	0,023470	**49,76**	**-6,23**	1,81
2,24	48,1156	-5,60513	1,823500	-1,2985	-0,03295	0,026050	**49,41**	**-5,57**	1,80
2,28	47,8514	-5,42859	1,841230	-1,2926	-0,03079	0,028090	**49,14**	**-5,40**	1,81
2,32	47,8357	-5,70326	1,896410	-1,2904	-0,03054	0,028600	**49,13**	**-5,67**	1,87
2,36	48,0799	-6,24805	1,958230	-1,2931	-0,03225	0,028010	**49,37**	**-6,22**	1,93
2,4	48,4710	-6,91031	2,029470	-1,2995	-0,03441	0,027160	**49,77**	**-6,88**	2,00
2,44	48,8796	-7,45739	2,055950	-1,3086	-0,03550	0,025640	**50,19**	**-7,42**	2,03
2,48	49,1143	-7,59267	2,027610	-1,3171	-0,03485	0,024080	**50,43**	**-7,56**	2,00
2,52	48,9797	-7,09674	1,996040	-1,3200	-0,03284	0,024080	**50,30**	**-7,06**	1,97
2,56	48,4762	-6,17913	1,952150	-1,3160	-0,03010	0,024860	**49,79**	**-6,15**	1,93
2,6	47,9169	-5,40232	1,945010	-1,3075	-0,02678	0,026100	**49,22**	**-5,38**	1,92
2,64	47,6348	-5,15037	1,974310	-1,2998	-0,02419	0,026120	**48,93**	**-5,13**	1,95
2,68	47,7342	-5,34336	2,019400	-1,2986	-0,02407	0,025210	**49,03**	**-5,32**	1,99
2,72	48,0948	-5,76293	2,091690	-1,3033	-0,02656	0,022990	**49,40**	**-5,74**	2,07
2,76	48,5586	-6,33454	2,162930	-1,3115	-0,03044	0,020270	**49,87**	**-6,30**	2,14
2,8	49,0477	-6,84825	2,153390	-1,3218	-0,03414	0,017490	**50,37**	**-6,81**	2,14
2,84	49,4177	-6,99764	2,033150	-1,3310	-0,03650	0,015970	**50,75**	**-6,96**	2,02
2,88	49,4115	-6,59965	1,893460	-1,3338	-0,03655	0,016120	**50,75**	**-6,56**	1,88
2,92	48,9156	-5,77288	1,788790	-1,3269	-0,03339	0,016560	**50,24**	**-5,74**	1,77
2,96	48,2645	-5,04053	1,751760	-1,3164	-0,02889	0,016750	**49,58**	**-5,01**	1,74
3	47,8387	-4,75081	1,832540	-1,3075	-0,02478	0,016390	**49,15**	**-4,73**	1,82
3,04	47,8158	-4,83842	1,971890	-1,3039	-0,02308	0,014880	**49,12**	**-4,82**	1,96
3,08	48,1467	-5,27215	2,091560	-1,3071	-0,02486	0,013000	**49,45**	**-5,25**	2,08
3,12	48,6811	-6,08025	2,132830	-1,3161	-0,02791	0,011710	**50,00**	**-6,05**	2,12
3,16	49,2370	-6,94320	2,036460	-1,3279	-0,03021	0,010380	**50,56**	**-6,91**	2,03
3,2	49,6127	-7,39823	1,794500	-1,3373	-0,03154	0,011670	**50,95**	**-7,37**	1,78

3,24	49,5982	-7,02741	1,588020	-1,3401	-0,03143	0,015670	**50,94**	**-7,00**	1,57
3,28	49,1150	-6,00785	1,573400	-1,3317	-0,02833	0,018810	**50,45**	**-5,98**	1,55
3,32	48,4317	-4,96031	1,652470	-1,3175	-0,02514	0,019300	**49,75**	**-4,94**	1,63
3,36	47,8601	-4,25926	1,732540	-1,3048	-0,02354	0,017120	**49,16**	**-4,24**	1,72
3,4	47,7259	-3,98667	1,849980	-1,2985	-0,02529	0,015410	**49,02**	**-3,96**	1,83
3,44	47,9877	-4,11206	1,968280	-1,2996	-0,03030	0,015370	**49,29**	**-4,08**	1,95
3,48	48,4294	-4,71708	2,045500	-1,3075	-0,03631	0,015360	**49,74**	**-4,68**	2,03
3,52	48,9427	-5,59835	2,107380	-1,3213	-0,04094	0,014230	**50,26**	**-5,56**	2,09
3,56	49,3916	-6,23763	2,133470	-1,3357	-0,04254	0,013740	**50,73**	**-6,20**	2,12
3,6	49,5034	-6,14737	2,108630	-1,3453	-0,04155	0,015120	**50,85**	**-6,11**	2,09
3,64	49,1680	-5,41570	2,043150	-1,3465	-0,03825	0,016940	**50,51**	**-5,38**	2,03
3,68	48,5987	-4,58372	1,998480	-1,3382	-0,03348	0,018560	**49,94**	**-4,55**	1,98
3,72	48,0874	-4,07495	2,054510	-1,3258	-0,02990	0,018700	**49,41**	**-4,05**	2,04
3,76	47,9106	-4,00786	2,167080	-1,3160	-0,02865	0,016820	**49,23**	**-3,98**	2,15
3,8	48,1364	-4,33750	2,211220	-1,3117	-0,03079	0,014180	**49,45**	**-4,31**	2,20
3,84	48,6206	-5,13329	2,178730	-1,3130	-0,03552	0,011380	**49,93**	**-5,10**	2,17
3,88	49,2378	-6,19381	2,067470	-1,3199	-0,03967	0,009080	**50,56**	**-6,15**	2,06
3,92	49,7745	-7,00383	1,861430	-1,3289	-0,04088	0,008140	**51,10**	**-6,96**	1,85
3,96	49,9243	-6,92897	1,677720	-1,3345	-0,04003	0,009620	**51,26**	**-6,89**	1,67
4	49,5740	-6,09414	1,613660	-1,3321	-0,03640	0,013100	**50,91**	**-6,06**	1,60
4,04	48,9667	-5,12121	1,634390	-1,3234	-0,03180	0,016510	**50,29**	**-5,09**	1,62
4,08	48,3583	-4,37344	1,749210	-1,3138	-0,02845	0,017420	**49,67**	**-4,34**	1,73
4,12	48,0162	-3,99457	1,931870	-1,3060	-0,02815	0,016150	**49,32**	**-3,97**	1,92
4,16	48,0746	-3,92552	2,053160	-1,3021	-0,03114	0,013330	**49,38**	**-3,89**	2,04
4,2	48,3469	-4,26389	2,079520	-1,3023	-0,03625	0,010020	**49,65**	**-4,23**	2,07
4,24	48,6721	-4,96136	2,038760	-1,3087	-0,04128	0,007750	**49,98**	**-4,92**	2,03
4,28	48,9757	-5,64347	1,942610	-1,3202	-0,04462	0,007780	**50,30**	**-5,60**	1,93
4,32	49,1325	-5,82672	1,826870	-1,3308	-0,04550	0,009710	**50,46**	**-5,78**	1,82
4,36	49,0036	-5,43472	1,746180	-1,3348	-0,04440	0,012810	**50,34**	**-5,39**	1,73
4,4	48,6461	-4,75998	1,764980	-1,3316	-0,04265	0,015750	**49,98**	**-4,72**	1,75
4,44	48,2312	-4,23812	1,858920	-1,3240	-0,04064	0,017630	**49,56**	**-4,20**	1,84
4,48	48,0242	-4,20963	1,953230	-1,3172	-0,03908	0,017920	**49,34**	**-4,17**	1,94
4,52	48,1577	-4,68120	2,024630	-1,3141	-0,03816	0,016830	**49,47**	**-4,64**	2,01
4,56	48,5324	-5,47533	2,083480	-1,3147	-0,03826	0,015510	**49,85**	**-5,44**	2,07
4,6	49,0016	-6,35539	2,107390	-1,3183	-0,03952	0,015510	**50,32**	**-6,32**	2,09
4,64	49,3942	-6,91457	2,023970	-1,3238	-0,04041	0,016960	**50,72**	**-6,87**	2,01
4,68	49,5001	-6,79165	1,858980	-1,3280	-0,03970	0,019580	**50,83**	**-6,75**	1,84
4,72	49,2282	-6,10402	1,739350	-1,3283	-0,03764	0,022530	**50,56**	**-6,07**	1,72
4,76	48,7164	-5,32292	1,726880	-1,3239	-0,03556	0,024220	**50,04**	**-5,29**	1,70
4,8	48,2119	-4,76974	1,779360	-1,3171	-0,03472	0,022970	**49,53**	**-4,74**	1,76
4,84	47,9901	-4,57395	1,845360	-1,3115	-0,03488	0,019640	**49,30**	**-4,54**	1,83
4,88	48,1015	-4,78840	1,893160	-1,3085	-0,03545	0,016060	**49,41**	**-4,75**	1,88
4,92	48,4053	-5,42714	1,945450	-1,3082	-0,03670	0,013980	**49,71**	**-5,39**	1,93
4,96	48,8221	-6,32086	2,005330	-1,3123	-0,03825	0,014540	**50,13**	**-6,28**	1,99
5	49,2621	-7,06628	1,996030	-1,3207	-0,03945	0,017320	**50,58**	**-7,03**	1,98

De la Tabla 3 se pueden extraer las gráficas mostradas en las Figuras 81, 82 y 83 que relacionan la influencia de cada componente de la fuerza de presión y fuerza de fricción sobre cada componente de la fuerza total hidrodinámica. Se muestra en estas gráficas una alta oscilación inicial debida a la transición de transitoria a estacionaria del fluido en la que se adapta a la forma del ROV.

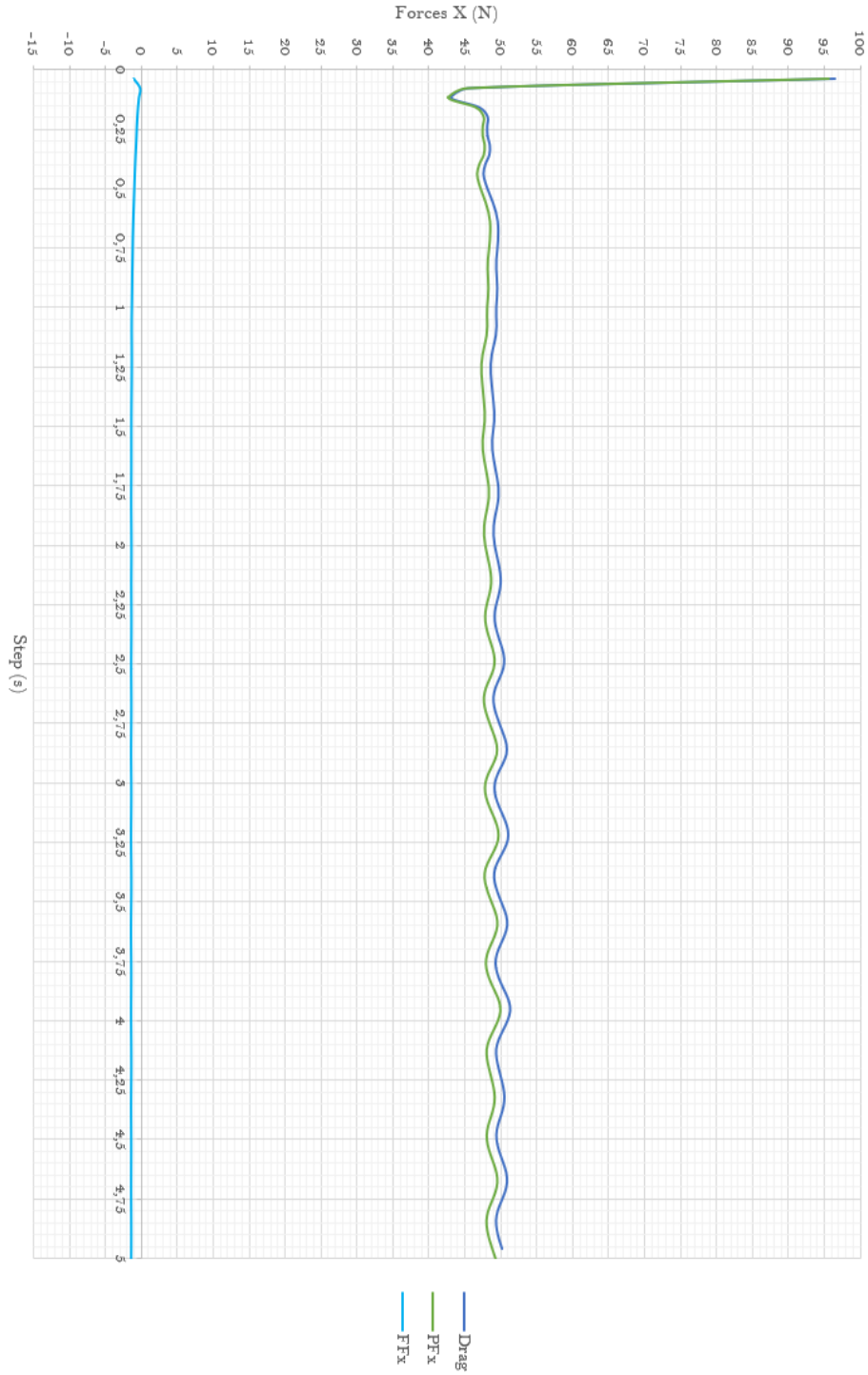


Figura 81. *Fuerza de arrastre (Drag) con sus componentes PFx y FFx.*

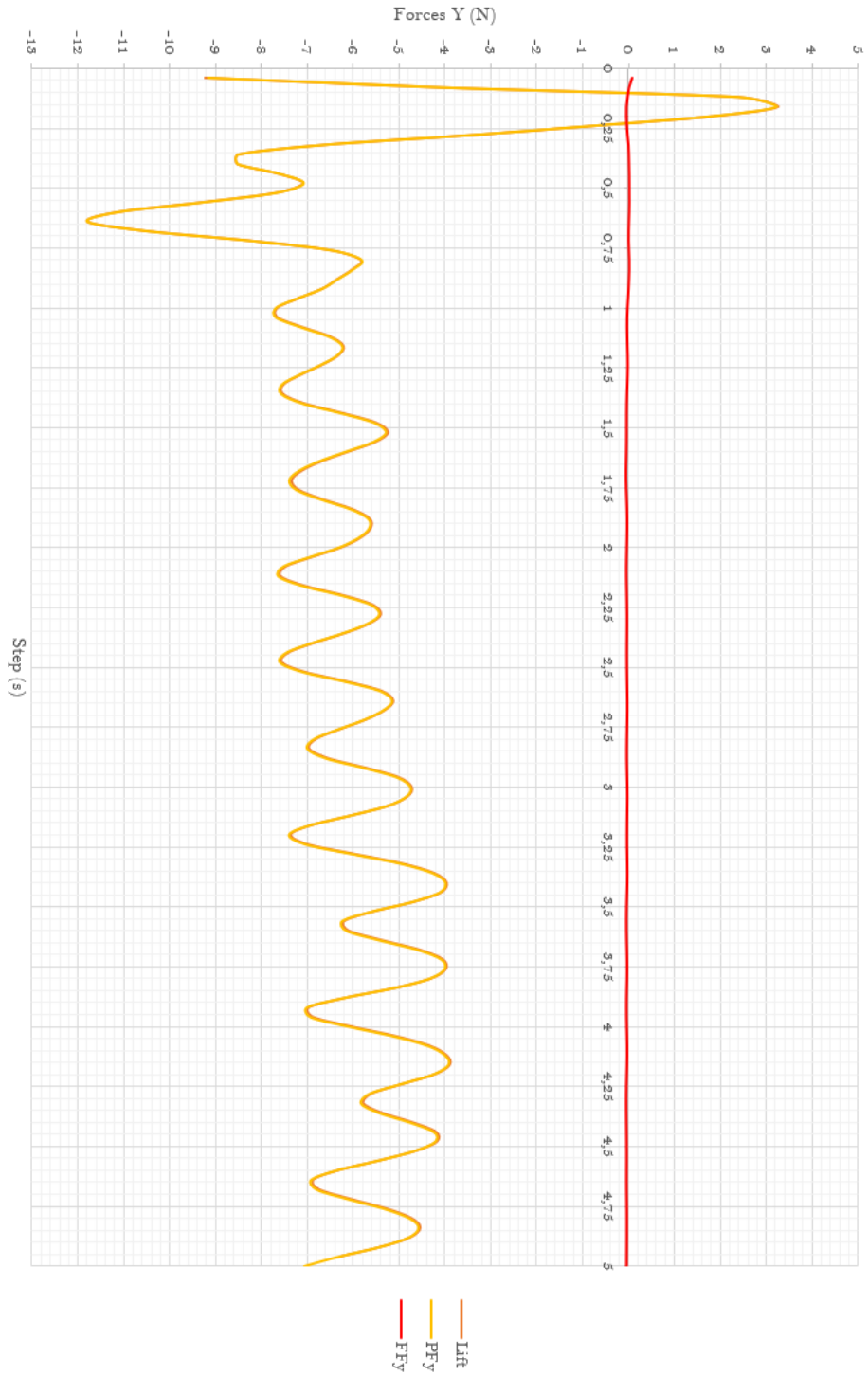


Figura 82. *Fuerza de sustentación (Lift) con sus componentes PFy y FFy.*

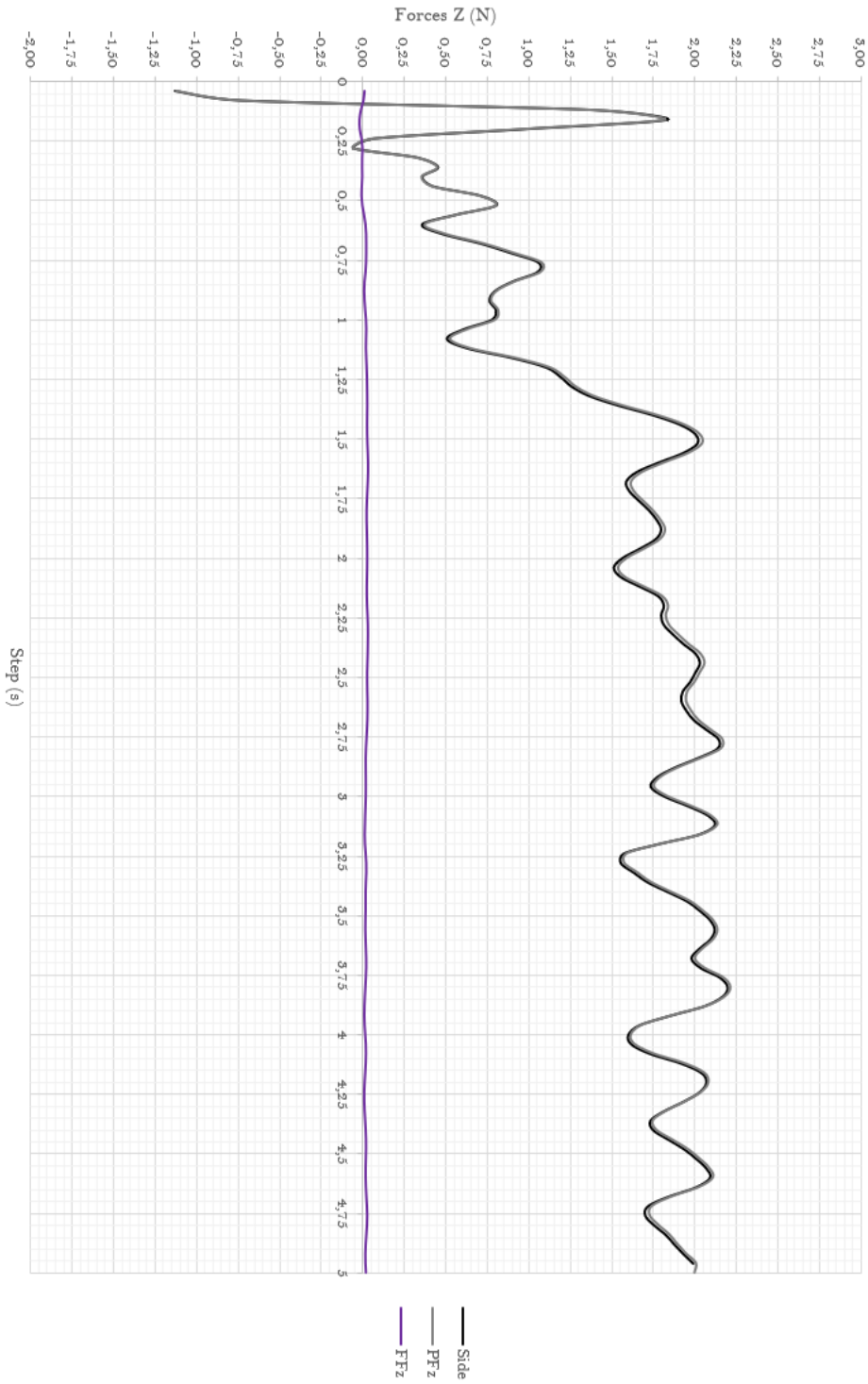


Figura 83. *Fuerza de desvío (Side) con sus componentes PFz y FFz.*

De las Figuras 81, 82 y 83 se constata la gran dominancia en el ROV que produce la fuerza de presión sobre la fuerza de fricción, siendo esta última más relevante en su componente *X*, aunque con una influencia mínima. Para las componentes *Y* y *Z* la influencia de la fuerza de fricción es prácticamente cero.

Una fuerza de fricción tan despreciable indica una mínima adherencia del fluido sobre la superficie del ROV. Este fenómeno se debe principalmente a su diseño poco afilado (poco hidrodinámico) que genera una separación del flujo tan grande que impide la correcta adherencia del mismo, y se traduce en la dominancia fuerza de presión. La visualización extraída de ParaView® del fenómeno comentado se muestra posteriormente en este Capítulo.

En la Figura 84 se muestra una comparación gráfica de cada componente de la fuerza hidrodinámica. En ella se puede apreciar la dominancia de la fuerza de arrastre sobre las otras componentes, con un valor medio de convergencia de 50 N. Esta dominancia es común en la mayoría de los cuerpos, pero es mucho más relevante para el ROV por la razón comentada previamente.

A partir de la fuerza de arrastre se puede realizar una estimación de la potencia hidrodinámica necesaria para mantener constante la velocidad del ROV.

$$P = F_D \cdot v = 50\ N \cdot 1\ \frac{m}{s} = 50\ W \tag{29}$$

Aumentar la velocidad del ROV supone aumentar la potencia y consumo de combustible exponencialmente, debido a que la fuerza de arrastre es proporcional a la velocidad al cuadrado. Esto ocurre para cualquier vehículo ya sea subacuático, terrestre o aéreo. La potencia hidrodinámica o aerodinámica no se debe confundir con la potencia mecánica, cuyo valor será mayor debido a las pérdidas mecánicas.

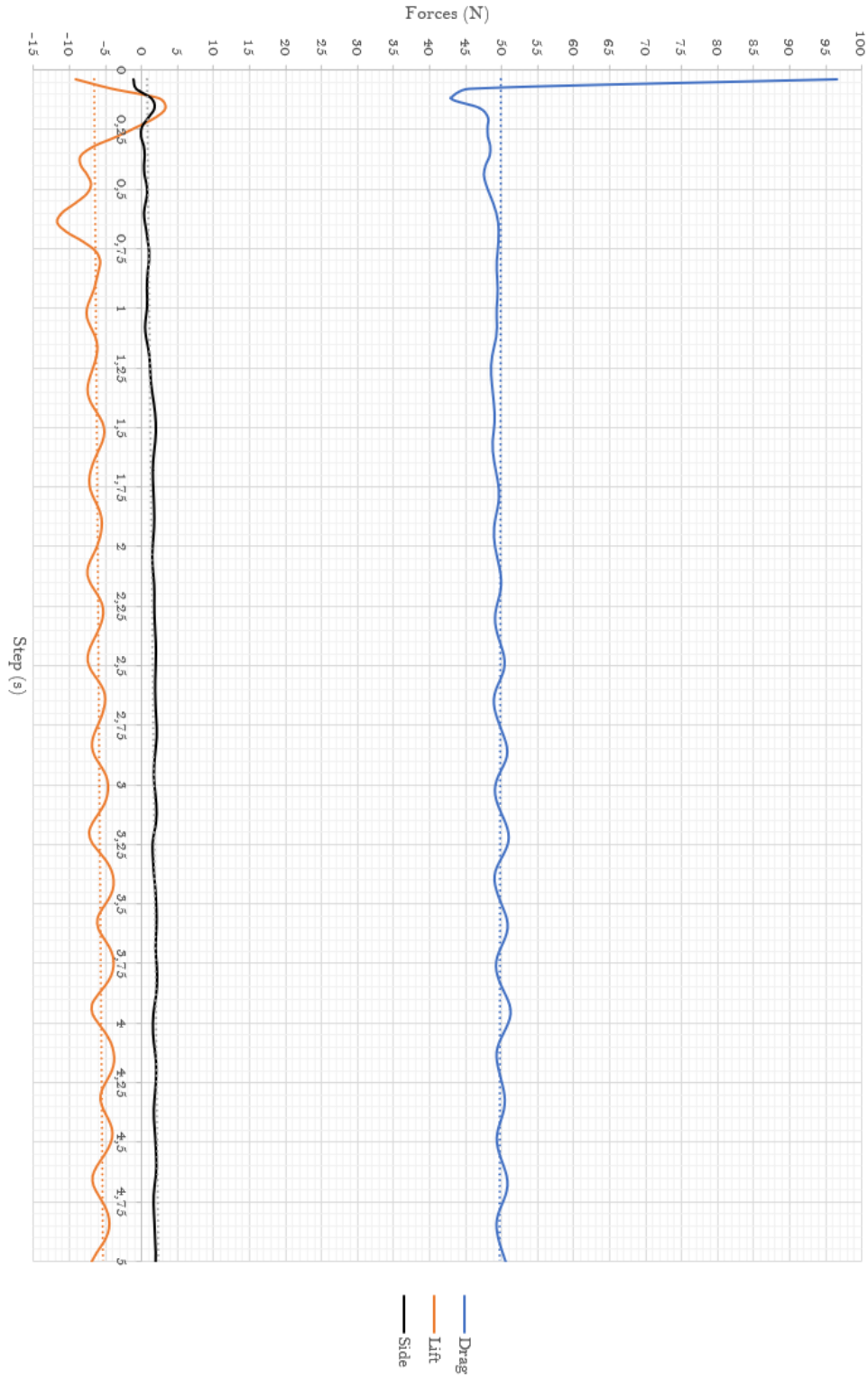


Figura 84. *Fuerza total hidrodinámica con sus componentes Drag, Lift y Side.*

Con respecto a la fuerza de sustentación se obtiene un valor medio de -6,5 N, con una oscilación en el tiempo más acusada que para la fuerza de arrastre. El valor negativo del mismo indica que el sentido de esta fuerza es opuesto al que se supuso, siendo entonces vertical hacia abajo.

Por otra parte, se puede apreciar como la fuerza de desvío es despreciable en comparación con las componentes comentadas, con un valor medio de 2 N. Es este el motivo por el que su estudio es irrelevante tanto para este caso como para la mayoría de los casos.

En Li *et al.* (2020) se realiza un estudio hidrodinámico de la fuerza de arrastre para los movimientos avance, desvío y moviento vertical del ROV BlueROV2. Sus resultados simulados y expermientales arrojan un valor aproximado de 38 y 45 N respectivamente, para una velocidad de avance de 1 m/s.

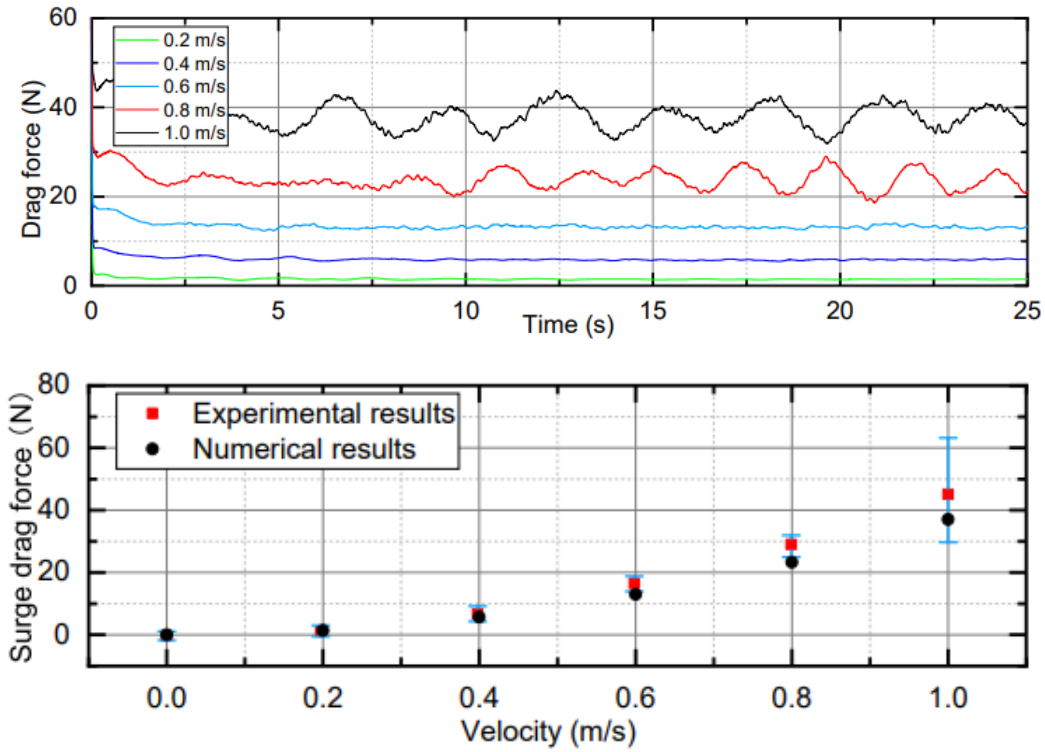


Figura 85. *Fuerza de arrastre para el avance del BluROV2 (Li et al., 2020).*

Comparando estos resultados con los obtenidos mediante la simulación del ROV Sibiu Pro, tiene sentido que BlueROV2 genere una resistencia al movimiento de avance menor que Sibiu Pro (50 N). Esto es debido al diseño más compacto y afilado de BlueROV2 que genera una separación de flujo menos acusada, y por tanto menor fuerza de presión.

Figura 86. *Vista frontal de BlueROV2 (https://www.turbosquid.com/es/3d-models/3d-underwater-robot-bluerov2-rov-model-1521263).*

A partir de los valores medios de fuerzas de arrastre y sustentación, y sus áreas transversales obtenidas de SolidWorks®, se puede entonces calcular los coeficientes de arrastre y sustentación.

$$C_D = \frac{2\,|F_D|}{\rho\, A_D\, U^2} = \frac{2 \cdot |50\ \mathrm{N}|}{1.000\ \frac{kg}{m^3} \cdot 0{,}05479\ m^2 \left(1\ \frac{m}{s}\right)^2} = 1{,}825 \tag{30}$$

$$C_L = \frac{2\,|F_L|}{\rho\, A_L\, U^2} = \frac{2 \cdot |-6{,}5\ \mathrm{N}|}{1.000\ \frac{kg}{m^3} \cdot 0{,}14467\ m^2 \left(1\ \frac{m}{s}\right)^2} = 0{,}09 \tag{31}$$

Con respecto al coeficiente de sustentación, un valor tan bajo del mismo indica un mínimo desvío vertical debido al flujo sobre el ROV en su movimiento de avance, siendo en todo caso vertical hacia abajo por el signo negativo de la fuerza de sustentación.

En cuanto al valor tan alto del coeficiente de arrastre, se refleja numéricamente la carencia hidrodinámica del ROV Sibiu Pro. Sin embargo, no se puede comparar este valor con coeficientes de cuerpos interaccionados con flujos de aire, ya que la densidad del aire es 1,27 kg/m^3, la cual es 833 veces más pequeña que la densidad del agua.

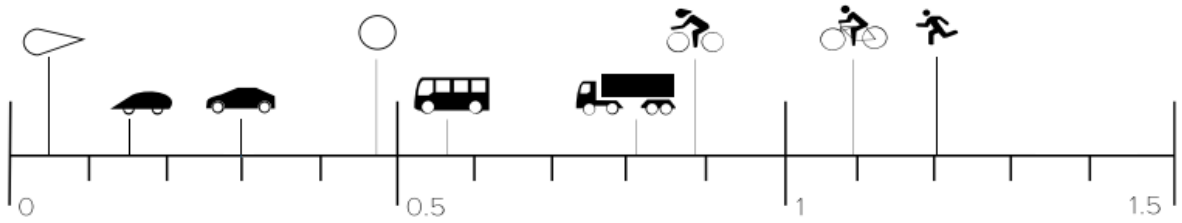


Figura 87. *Comparación del coeficiente de arrastre para fluido aire (AirShaper).*

Para obtener las fuerzas y coeficientes hidrodinámicos de Sibiu Pro para los movimientos de desvío y m. vertical a 1 m/s se puede volver a resolver el caso SibiuPro cambiando la orientación del ROV en SolidWorks®. Esta nueva orientación implica modificar los archivos *blockMeshDict* y *snappyHexMesh* para que el mallado se adapte al ROV.

Además se debe volver a calcular los valores de *k* y *omega* (o *epsilon* para el modelo de turbulencia k-ε) con una longitud característica de 520 mm correspondiente con el largo de Sibiu Pro. En las Figuras 88 y 89 se muestran las orientaciones del ROV dentro del mallado para ambos movimientos.

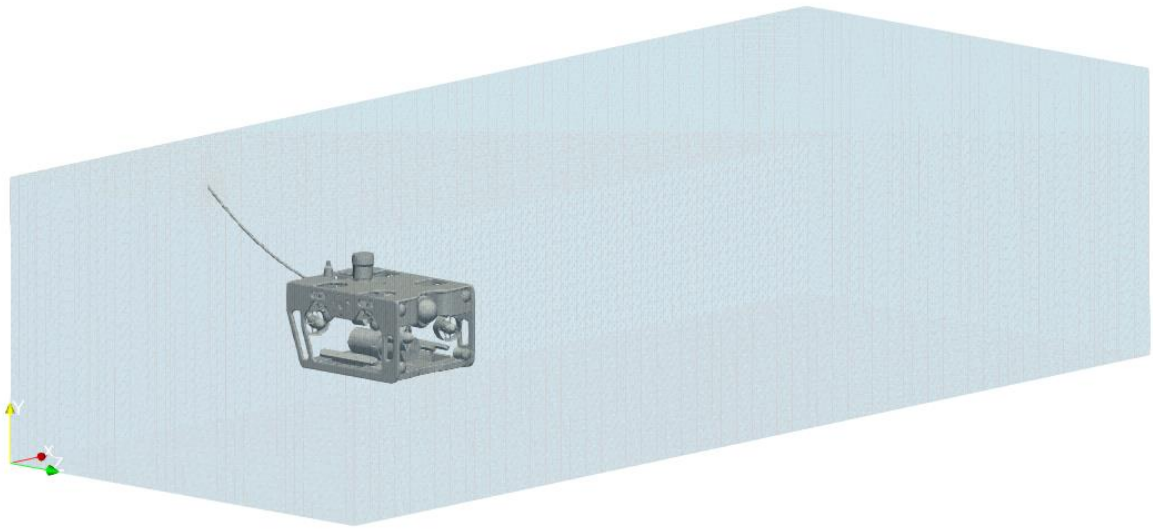


Figura 88. *Mallado para el movimiento de desvío a 1 m/s de SibiuPro.*

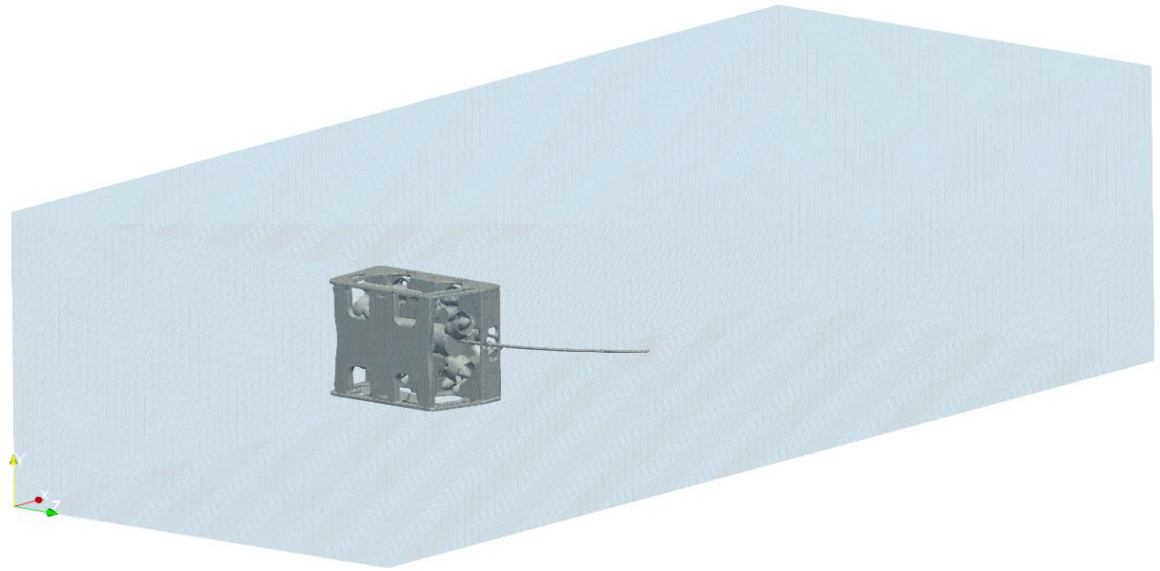


Figura 89. *Mallado para el movimiento vertical a 1 m/s de SibiuPro.*

En SolidWorks® se obtiene un área transversal de desvío de 0,09728 m2, necesaria para calcular el coeficiente de arrastre para el desvío y el coeficiente de sustentación del movimiento vertical.

En la Tabla 4 se recopilan los valores obtenidos de fuerzas y coeficientes de arrastre y sustentación para los tres movimientos principales, con las áreas arrastre y sustentación utilizada en cada caso.

Tabla 4. *Áreas, Fuerzas y Coeficientes en los 3 movimientos de Sibiu Pro.*

			A_D (m^2)	A_L (m^2)	F_D (N)	F_L (N)	C_D	C_L
X	Avance	**Surge**	0,05479	0,14467	**50**	-6,5	**1,825**	0,090
Y	Desvío	**Sway**	0,09728	0,14467	**89**	-6	**1,830**	0,083
Z	M. Vertical	**Heave**	0,14467	0,09728	**109,6**	0,7	**1,515**	0,014

Es previsible que ambos movimientos nuevos suponen un arrastre bastante superior al movimiento de avance, ya que el fluido incide perpendicularmente sobre áreas planas más grandes, generando así una separación de flujo mayor. Sin embargo, la variación de fuerzas de sustentación es mínima.

En Li *et al.* (2020) se realiza el mismo cálculo de fuerzas y coeficientes de arrastre para los mismos tres movimientos a 1 m/s del ROV BlueROV2.

Surge		Sway		Heave	
KL	KQ	KL	KQ	KL	KQ
1.3125	38.169	9.1435	129.6607	2.015	243.25

Figura 90. *Coeficientes y fuerzas de arrastre (KL y KQ) a 1 m/s (Li et al., 2020).*

A partir de dichos cálculos se obtiene la Tabla 5 comparativa de ambos ROVs.

Tabla 5. *Fuerzas y coeficientes de arrastre de Sibiu Pro y BlueROV2.*

	SURGE		SWAY		HEAVE	
	F_D (N)	C_D	F_D (N)	C_D	F_D (N)	C_D
Sibiu Pro	50	1,825	89	1,830	109,6	1,515
BlueROV2	38,169	1,3125	129,6607	9,1439	243,25	2,015

De la Tabla 5 se puede verificar un mejor comportamiento hidrodinámico del ROV BlueROV2 en su movimiento de avance (el más común e importante), generando un arrastre un 23,7 % menor que Sibiu Pro. Esto se debe al diseño más afilado de BlueROV2 que se aprecia en sus flotadores, generando así una separación de flujo menos acusada que los flotadores de Sibiu Pro que son perpendiculares al flujo.

Sin embargo esto no ocurre para los movimientos de desvío y m. vertical, para los cuales BlueROV2 genera una mayor fuerza de arrastre. El motivo de este fenómeno se debe principalmente a que su diseño de chasis es más compacto (menos abierto) que SibiuPro, generando así estancamientos de flujo que generan una mayor fuerza de presión.

5.2. Análisis CFD del ROV

En este apartado se muestran imágenes y videos para interpretar los resultados postprocesados en ParaView® de la interacción que el fluido ejerce sobre el ROV Sibiu Pro en su movimiento de avance a una velocidad constante de 1 m/s.

5.2.1. Nubes de presión

Las nubes de presión (*pressure clouds*) permiten visualizar la estela donde se genera la fuerza de arrastre por la separación de flujo que bordea dichas nubes. Se puede obtener mediante un contorno espacial donde el coeficiente de presión vale 0. El volumen interno del contorno se corresponde con las zonas donde este coeficiente es negativo, que es donde ocurre la generación y desprendimiento de vórtices. Cuanto menor sea dicho volumen, menor será la fuerza de arrastre, indicando una mejor hidrodinámica del cuerpo.

En las Figuras 91 y 92 se muestra como la geometría poco afilada de Sibiu Pro produce enormes nubes de presión debido a las caras planas sobre las que el fluido incide perpendicularmente durante el movimiento de avance. Son especialmente acusadas las nubes de presión que generan los flotadores delanteros, el cilindro de electrónica, la batería y el sonar 360. También se generan nubes en la parte trasera del vehículo debido a la forma rectangular del chasis posterior que no favorece la reagrupación del flujo superior con el flujo que bordea exteriormente al ROV. Se recuerda que en el Anexo VII se encuentra el plano de explosionado del ROV con la lista de cada uno de sus componentes.

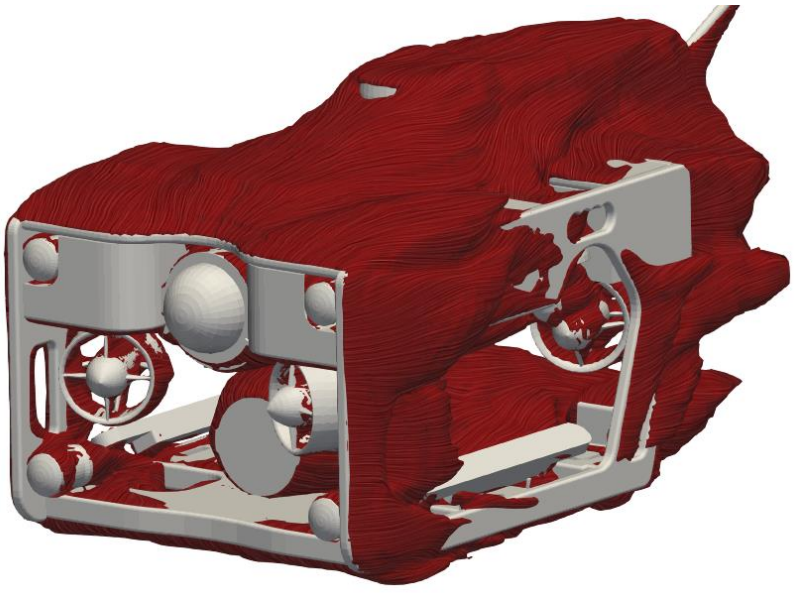


Figura 91. *Vista 3D de las nubes de presión sobre Sibiu Pro.*

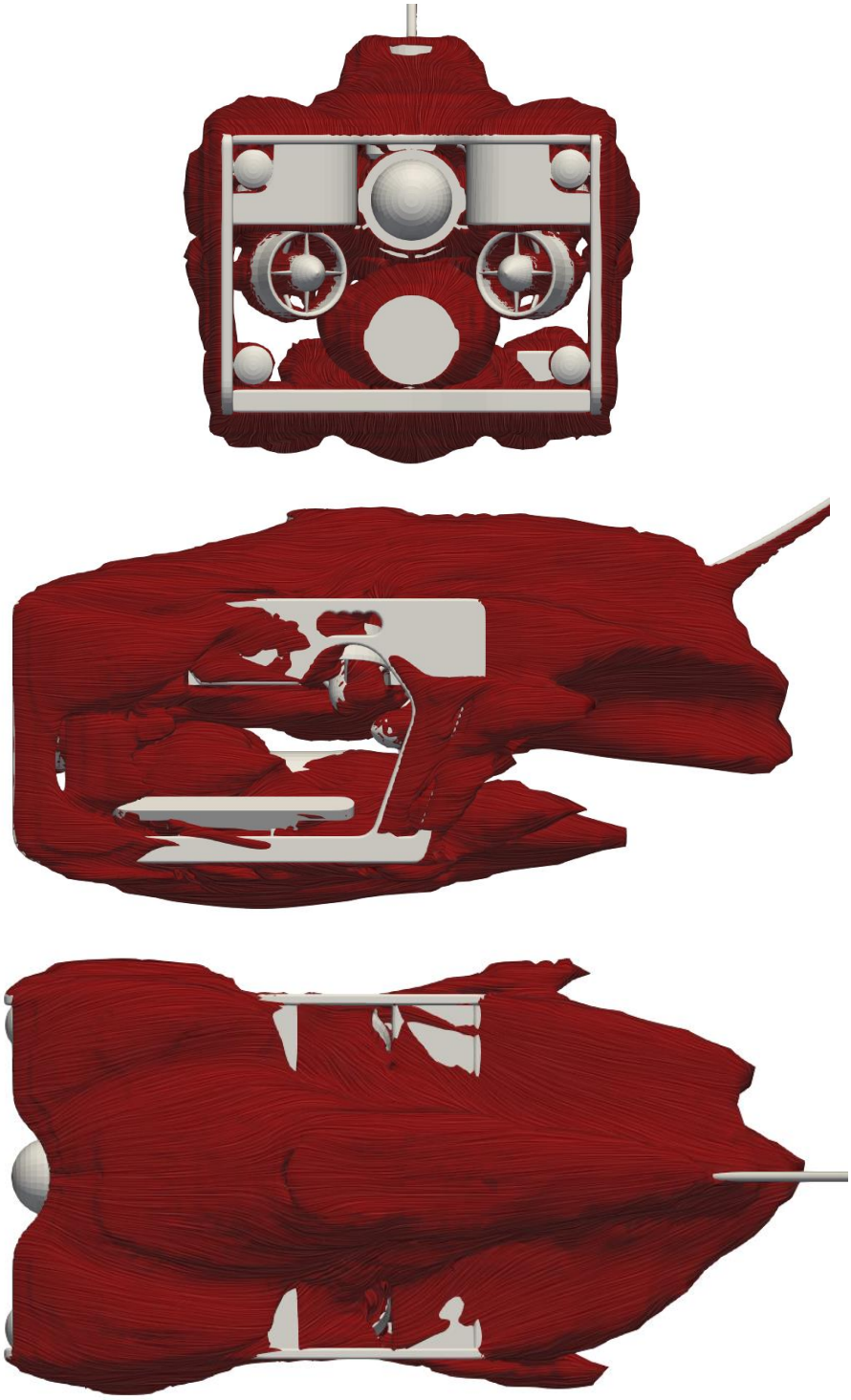


Figura 92. *Vistas de las nubes de presión sobre Sibiu Pro.*

En el siguiente enlace de video del canal de YouTube del Proyecto KTTSeaDrones se aprecia la evolución las nubes de presión mostradas en este apartado a lo largo del tiempo (minuto 0:20).

https://www.youtube.com/watch?v=xHGl1xOGoZA&ab_channel=kttseadrones

5.2.2. Presión superficial. Surface Pressure

La presión superficial es la presión que el fluido ejerce sobre el cuerpo perpendicular a la superficie, y que empuja o tira del mismo dependiendo de su sentido.

Las presiones más altas se alcanzan donde el fluido incide sobre caras frontales planas y perpendiculares al sentido del flujo, provocando una parada instantánea de las partículas por estancamiento. Cuanto más perpendicular al flujo sea una superficie, más probable es que se cree una sobrepresión de arrastre. Durante el movimiento de avance de Sibiu Pro se alcanzan presiones de estancamiento significativas de hasta 640 Pa tanto en los flotadores delanteros como en la punta del vidrio de protección de la cámara y sobre la batería.

Del mismo modo, una superficie orientada hacia atrás genera una depresión tira del cuerpo en sentido contrario, generando arrastre. Este fenómeno ocurre por el principio de Bernoulli, que establece que cuando la velocidad aumenta por separación de flujo, la presión debe disminuir, conservándose así la energía mecánica. Sobre Sibiu Pro se alcanza valores negativos muy puntuales de hasta -1.200 Pa sobre las superficies donde comienza la separación del flujo. Estas son las transiciones redondeadas de las caras frontales y laterales del chasis y la tapa superior, además de las superficies interiores de los rotores delanteros.

En la fase de diseño es importante reducir los picos de presión y depresión suavizando la superficie en transiciones críticas para así reducir el arrastre.

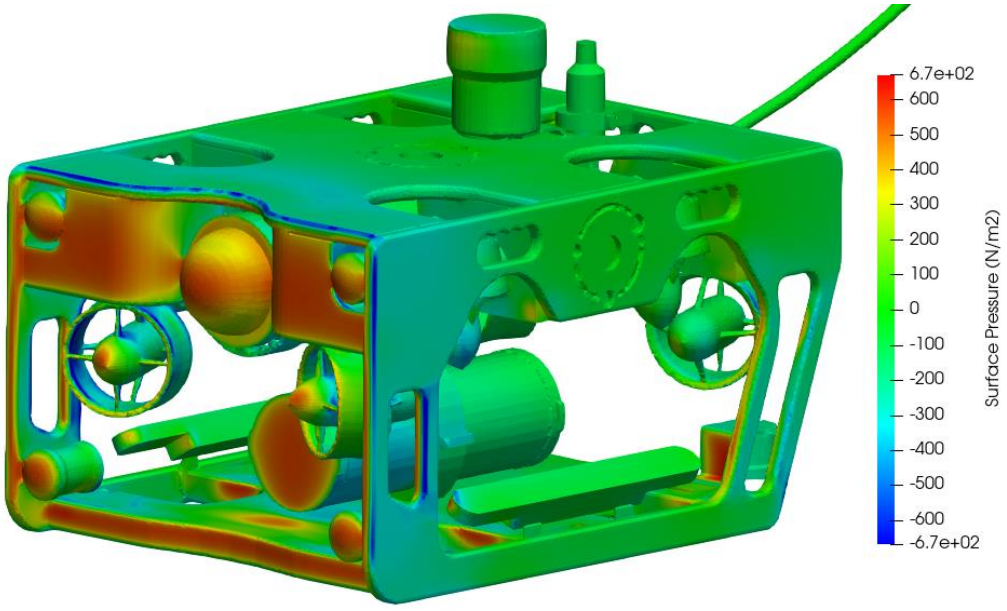


Figura 93. *Presión superficial sobre Sibiu Pro.*

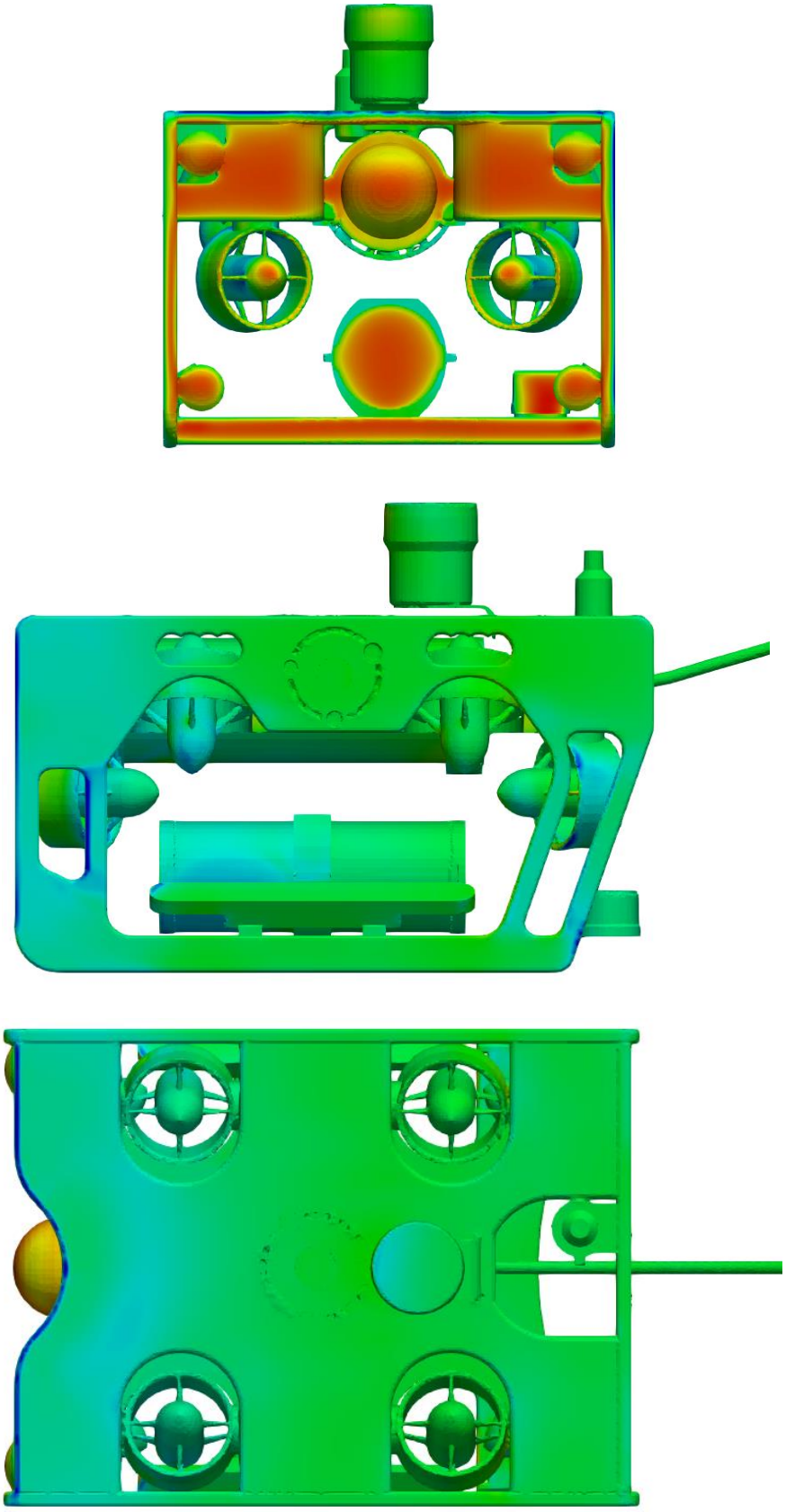


Figura 94. *Vistas de la presión superficial sobre Sibiu Pro.*

5.2.3. Fricción superficial. Wall Shear Stress

La fricción superficial es la tensión que el fluido ejerce sobre el cuerpo paralela a la superficie, en el sentido de las líneas de corriente superficiales.

Tal como puede compararse con los valores vistos para la presión superficial, la fricción superficial sobre Sibiu Pro es mínima debido a su geometría poco afilada. Las zonas donde la fricción se concentra son muy localizas en superficies donde el flujo incide a alta velocidad tras la separación del mismo. Algunas de estas superficies son la cara superior del chasis inferior y la cara inferior del cilindro de electrónica.

Las superficies completamente en azul se corresponden con zonas donde el fluido no experimenta adherencia con Sibiu Pro, ya sea por la separación de flujo previa o por la presión de estancamiento que frena completamente al flujo.

En las Figuras 95 y 96 se muestra la fricción superficial con un estilo de representación *Surface LIC*, que permite apreciar la dirección de las líneas de corriente sobre la superficie del ROV.

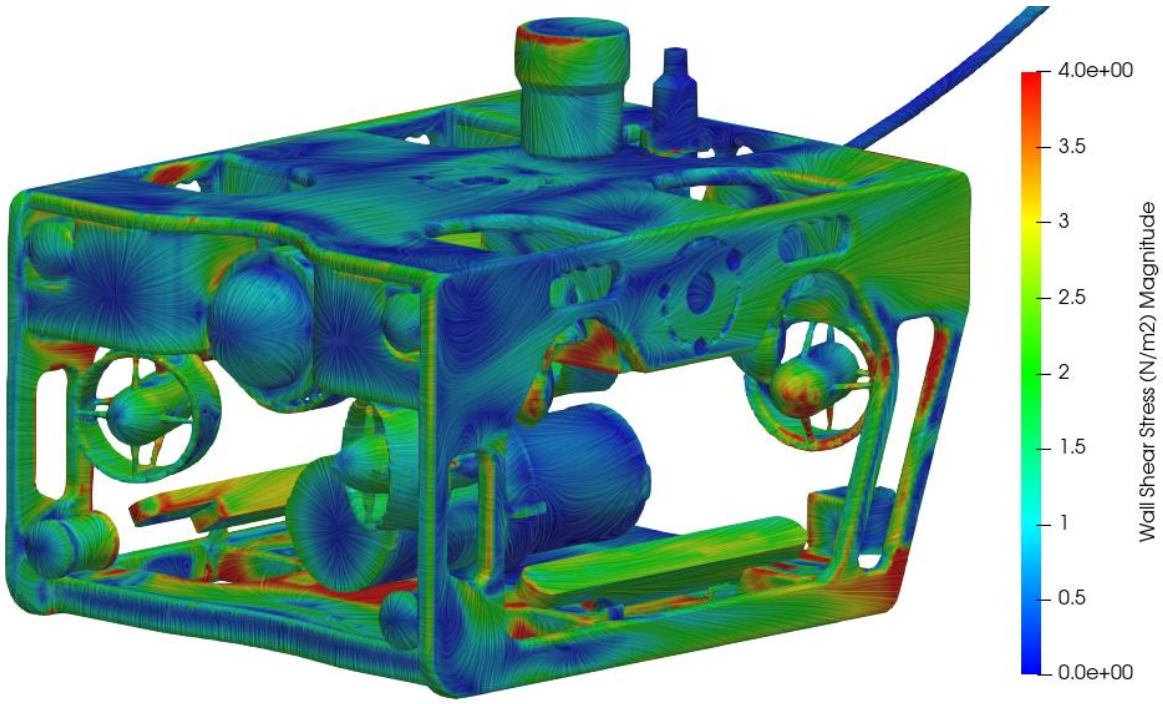


Figura 95. *Fricción superficial sobre Sibiu Pro.*

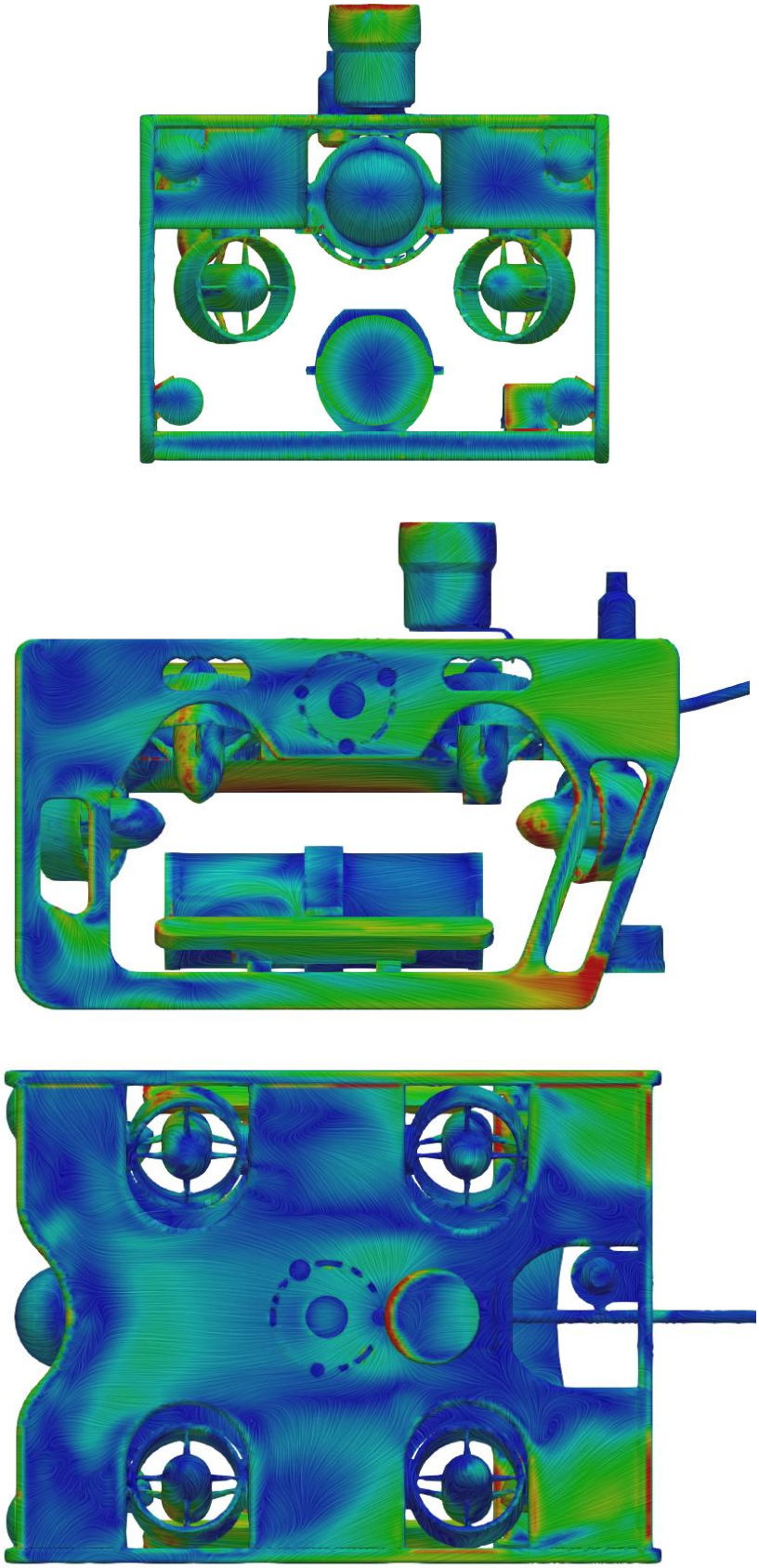


Figura 96. *Vistas de la fricción superficial sobre Sibiu Pro.*

5.3. Análisis CFD del flujo

En este apartado se muestran imágenes y videos para interpretar los resultados postprocesados en ParaView® de la interacción que el ROV Sibiu Pro ejerce sobre el fluido en su movimiento de avance a una velocidad constante de 1 m/s.

5.3.1. Líneas de corriente. Streamlines

Las líneas de flujo o de corriente son un recurso común en túneles de viento que consiste en liberar humo sobre la entrada del flujo de aire para conocer la trayectoria que sigue durante la interacción con cuerpos. La correcta compresión de la mencionada trayectoria del flujo a través de un cuerpo, es la base para optimizar su diseño.

En las Figuras 97 y 98 se muestra la trayectoria de las líneas de corriente verticales y centradas sobre Sibiu Pro, coloreadas según la magnitud de la velocidad del flujo. Se puede observar como la trayectoria del flujo sigue tres caminos distintos en su interacción con Sibiu Pro: superior, inferior y central.

Con respecto a los flujos superior e inferior se aprecian separaciones de flujo muy pronunciadas debidas a las caras planas frontales de la tapa superior y el chasis inferior respectivamente. Estas desviaciones provocan aceleraciones del flujo por decremento de presión, alcanzando velocidades de hasta 1,6 m/s (color rojo intenso), lo cual supone un incremento del 60%. El flujo superior se acelera nuevamente en su interacción con el sonar 360 y con el cordón umbilical posteriormente.

Por otra parte, el flujo central que atraviesa al ROV tiende a agruparse y recorrer superficialmente la zona inferior del cilindro de electrónica, generando una aceleración central y deceleración superficial del mismo. Esta interacción provoca la fricción superficial sobre el cilindro de electrónica que se muestra en la vista lateral del anterior apartado.

Sobre la batería se puede apreciar un gran estancamiento del flujo debido a la interacción con su cara frontal plana perpendicular al sentido flujo, perdiéndose toda su velocidad (color azul intenso) en su conversión a energía de presión. Este fenómeno no ocurre de manera tan acentuada sobre el cristal de protección de la cámara debido a su forma semiesférica. Esta geometría es mucho más hidrodinámica que las caras planas perpendiculares al flujo, ya que generan una separación de flujo inferior.

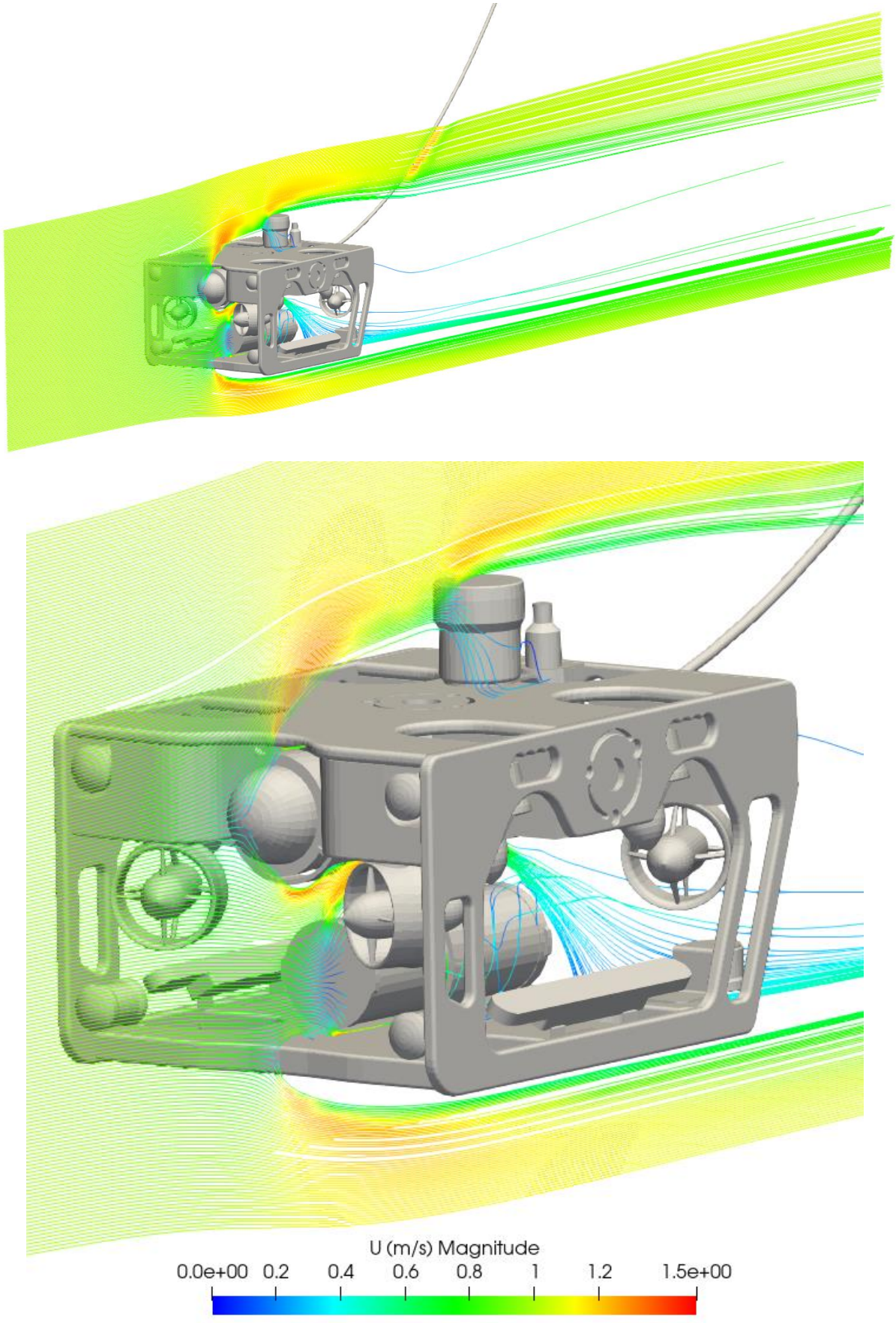


Figura 97. *Velocidad de las líneas de corriente verticales centradas.*

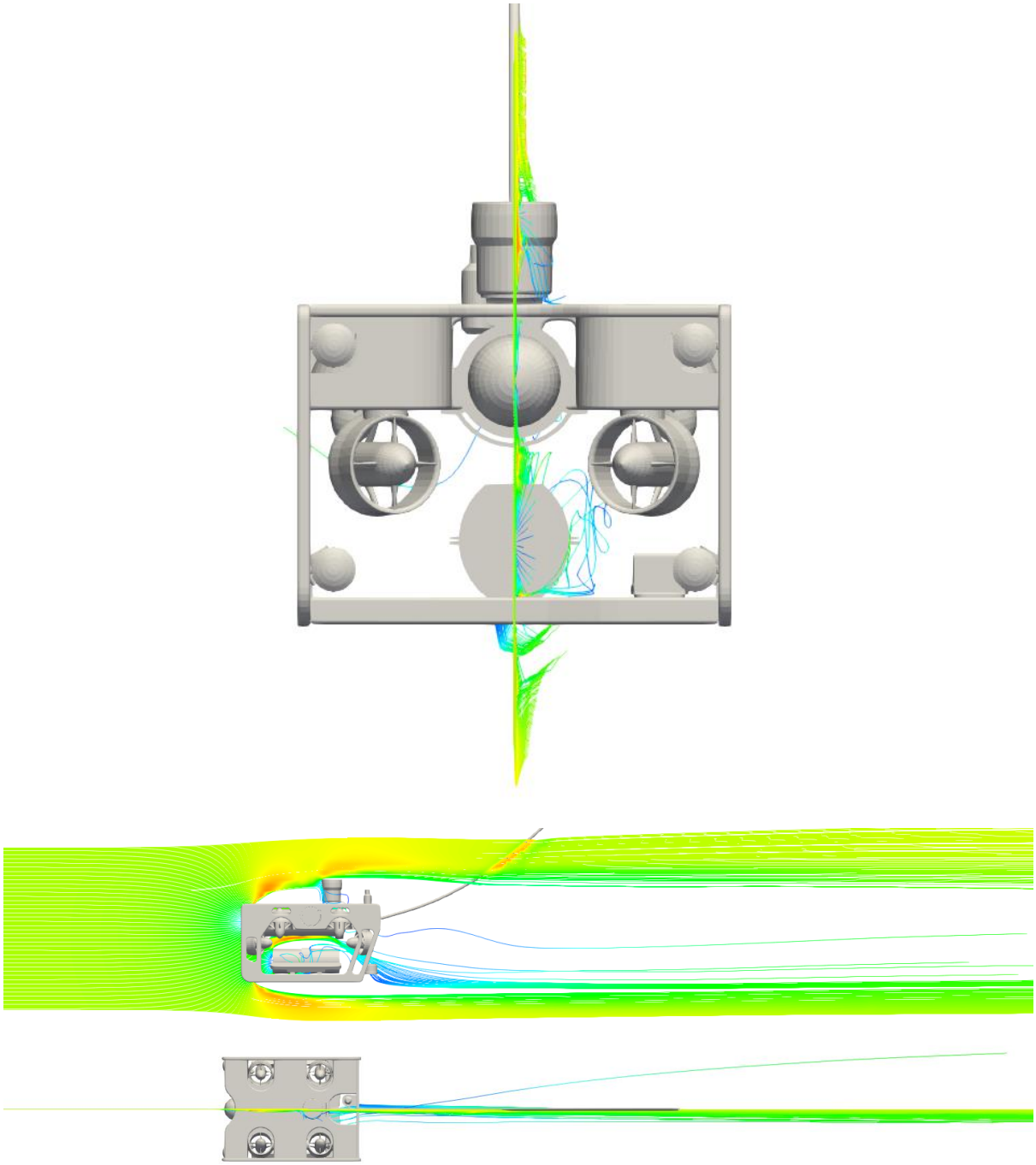


Figura 98. *Vistas de las líneas de corriente verticales centradas.*

En las Figuras 99 y 100 se muestran las mismas líneas de corriente verticales pero proyectadas sobre la cara frontal del chasis lateral izquierdo. Se aprecia una pronunciada separación de flujo, alcanzándose velocidades ligeramente menores que las generadas por las separaciones de flujo superior e inferior mostradas anteriormente.

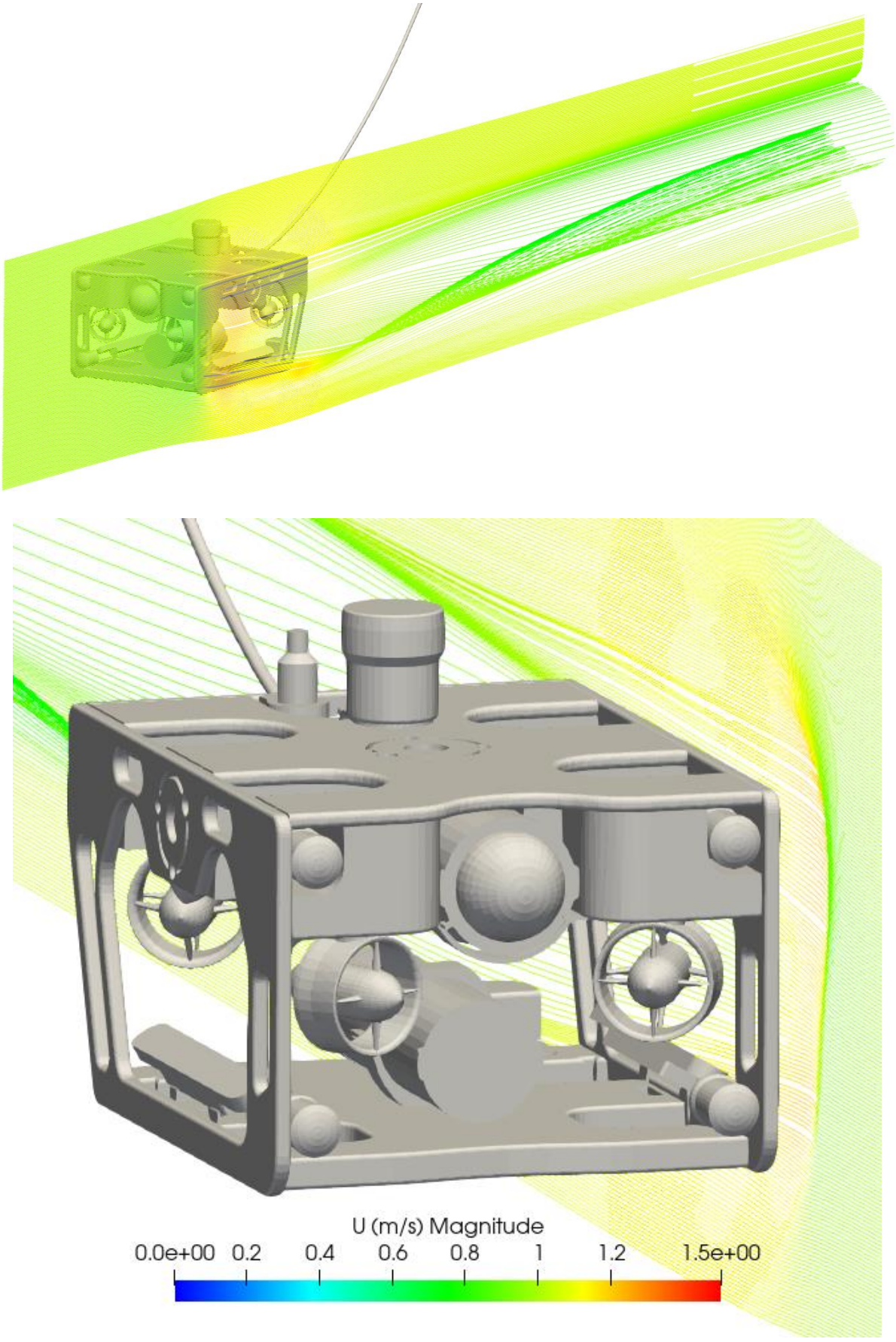


Figura 99. *Velocidad de las líneas de corriente verticales sobre el chasis lateral derecho.*

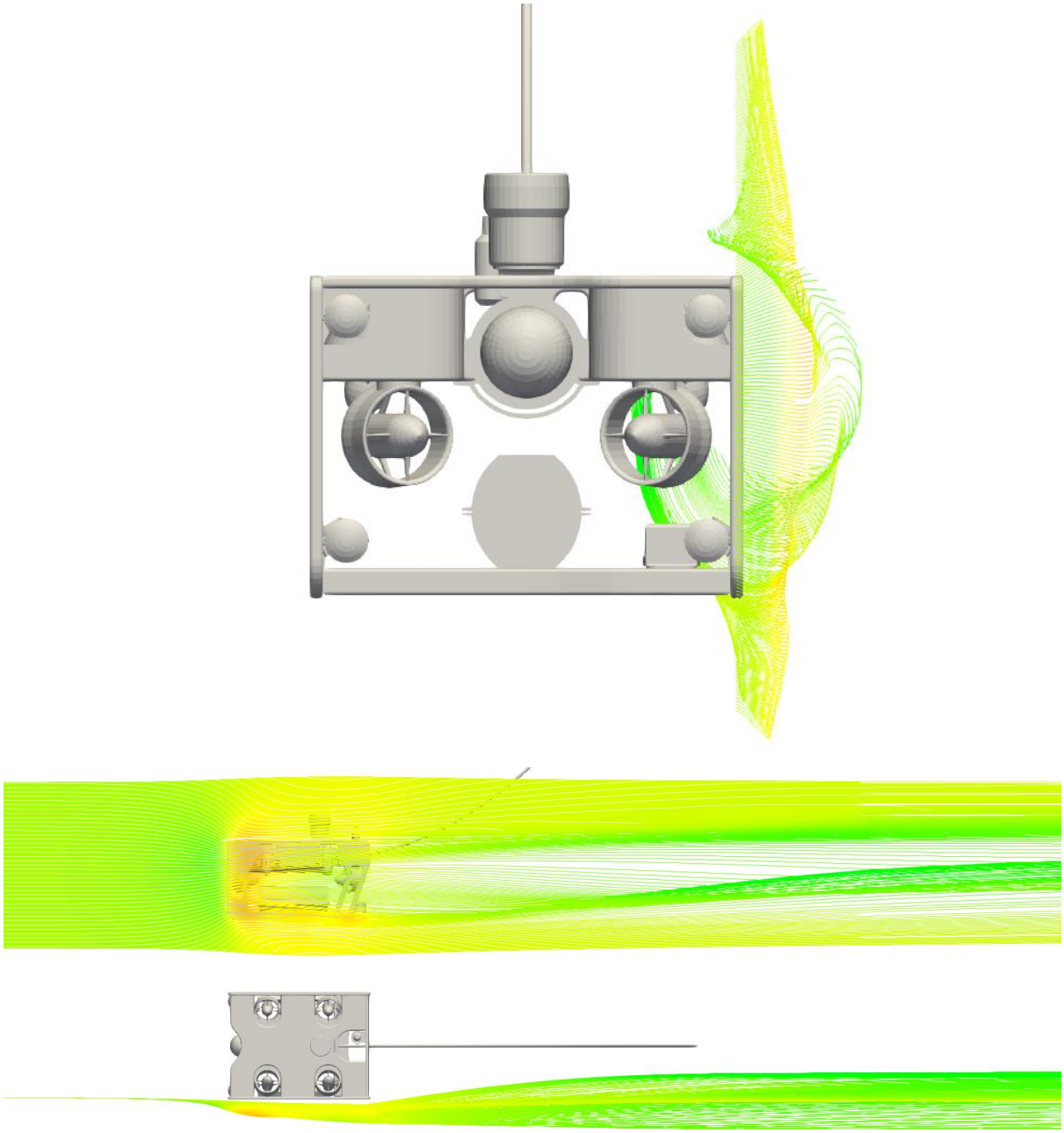


Figura 100. *Vistas de las líneas de corriente verticales sobre el chasis lateral derecho.*

De igual manera que las líneas de corriente verticales, se pueden generar líneas de corriente horizontales. En las Figura 101 y 102 se muestra las líneas de corriente horizontales generadas a la altura central del cilindro de electrónica.

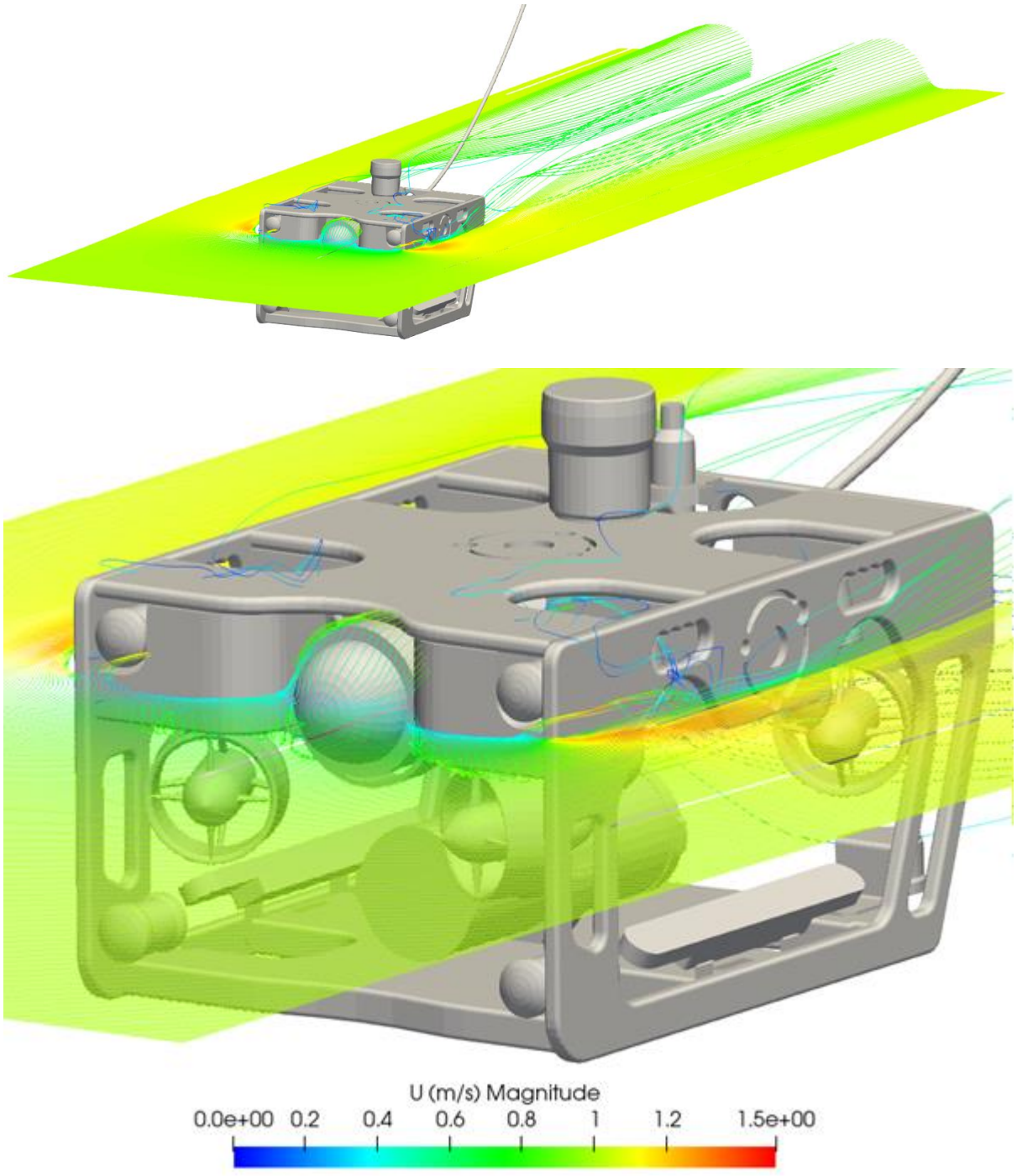


Figura 101. *Velocidad de las líneas de corriente horizontales sobre el cilindro de elect.*

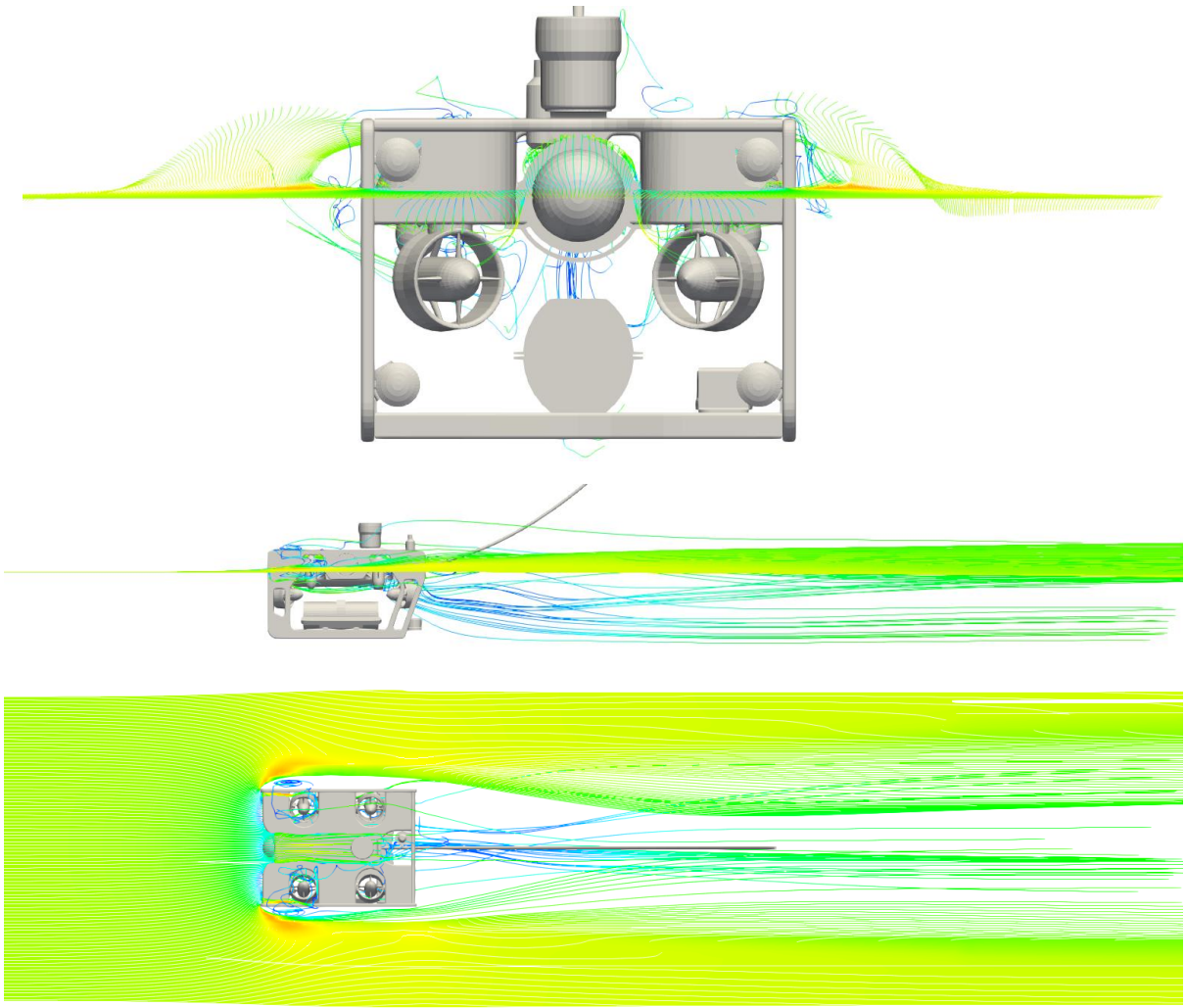


Figura 102. *Vistas de las líneas de corriente horizontales sobre el cilindro de elect.*

En Figuras 101 y 102 se observa como la trayectoria del flujo sigue tres caminos distintos en su interacción con Sibiu Pro: derecho, izquierdo y central.

Con respecto a los flujos derecho e izquierdo, se aprecian separaciones de flujo pronunciadas debidas a las caras planas frontales de los chasis laterales derecho e izquierdo, generando la estela que se muestra en la vista superior. De igual manera que para los flujos superior e inferior vistos anteriormente, estas desviaciones provocan aceleraciones del flujo por decremento de presión, alcanzando velocidades algo menores de hasta 1,4 m/s (color rojo intenso), lo cual supone un incremento del 40%.

Por otra parte, la incidencia del flujo central sobre el cristal de protección de la cámara provoca una reducción apreciable de la velocidad, que es aún más significativa sobre los flotadores debido a su geometría plana perpendicular al flujo. Esta interacción provoca un

desvío del flujo que recorre la superficie inferior del chasis medio, y la superficie superior del cilindro de electrónica.

Si se proyectan las misma líneas de corriente horizontales sobre la tapa superior de Sibiu Pro, se obtiene la interacción de corriente mostrada en las Figuras 103 y 104.

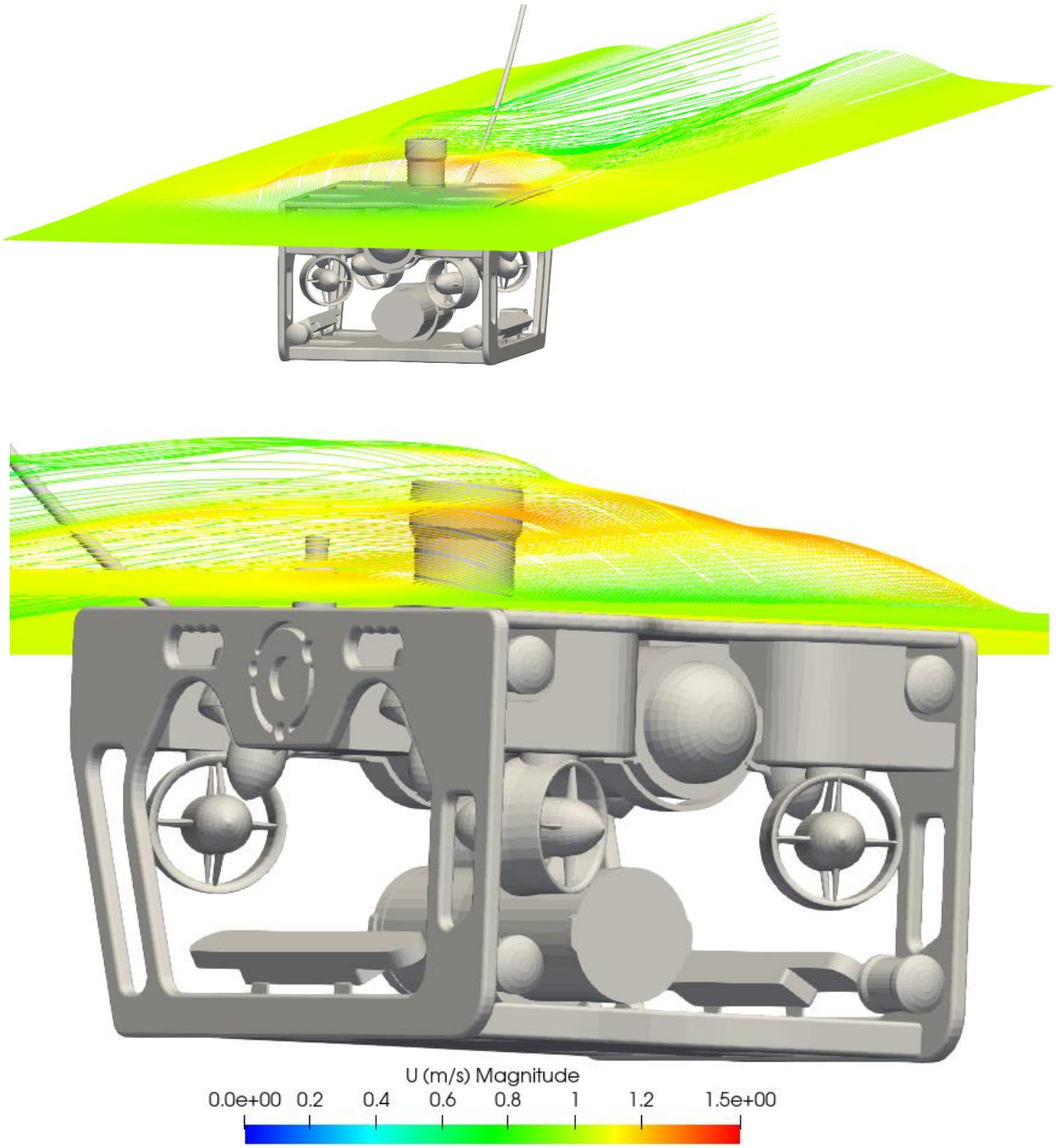


Figura 103. *Velocidad de las líneas de corriente horizontales sobre la tapa superior.*

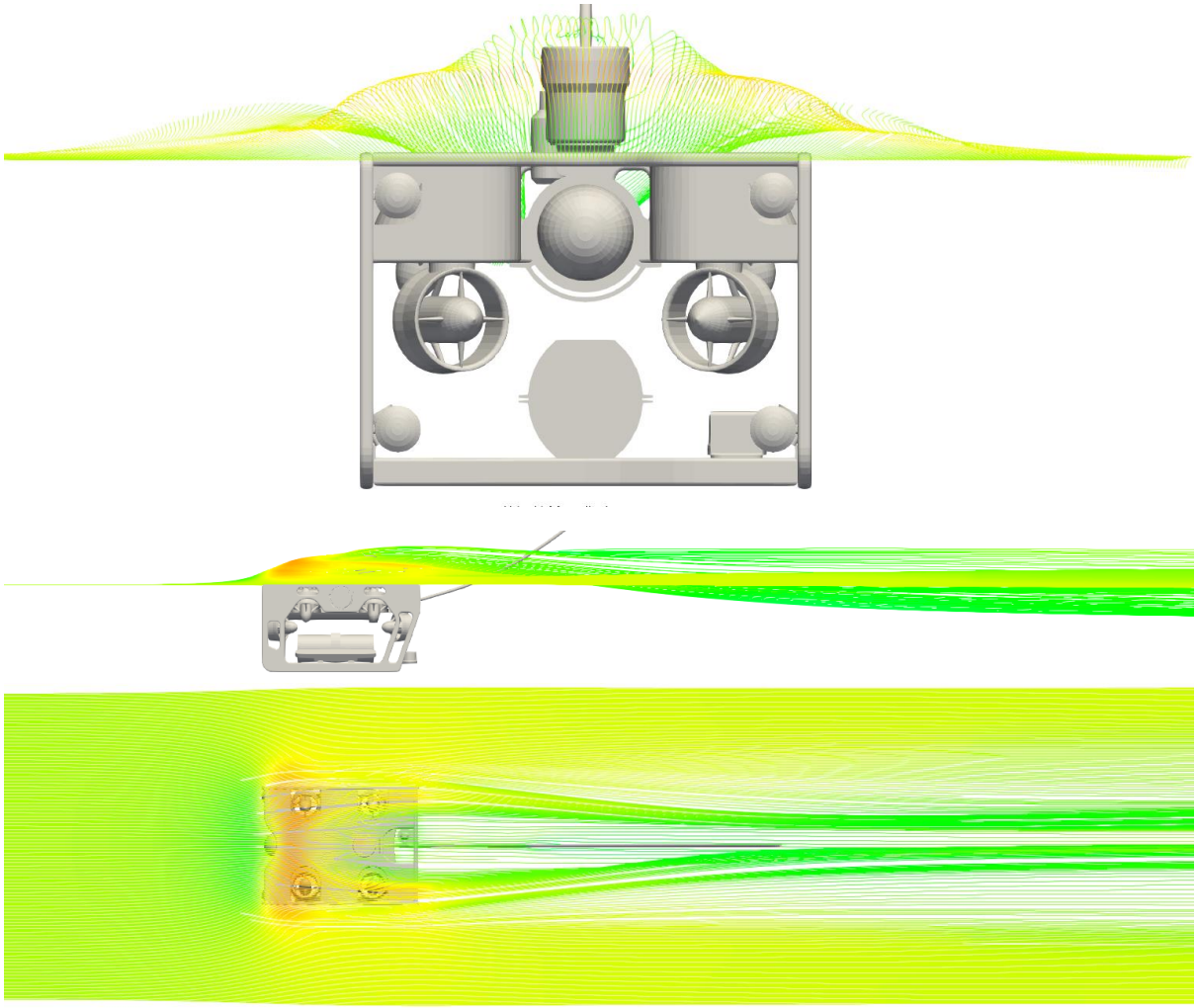


Figura 104. *Vistas de las líneas de corriente horizontales sobre la tapa superior.*

En Figuras 103 y 104 se muestra la gran magnitud de la separación y aceleración del flujo que provoca cara frontal de la tapa superior del ROV, que se desvía nuevamente debido al sonar 360. El resultado de las desviaciones del flujo se muestra en la estela generada.

Tal como se comentó previamente, la superficie de baja presión superior donde comienza la separación del flujo superior incrementa la velocidad de dicho flujo hasta 1,6 m/s.

En el mismo enlace de video de YouTube comentado anteriormente se muestran todas las interacciones de líneas de corriente planas mostradas en este apartado (minuto 0:32). Se recomienda pausar el video para apreciar cada tramo de interacción.

https://www.youtube.com/watch?v=xHGl1xOGoZA&ab_channel=kttseadrones

En ParaView® existe la posibilidad de generar las líneas de corriente que atreviesan una esfera de posición y radio definido. Si se coloca dicha esfera en el centro geométrico de Sibiu Pro, con un diámetro de 0,6 m se pueden generar las 1.000 líneas de corriente mostradas en las Figuras 105 y 106.

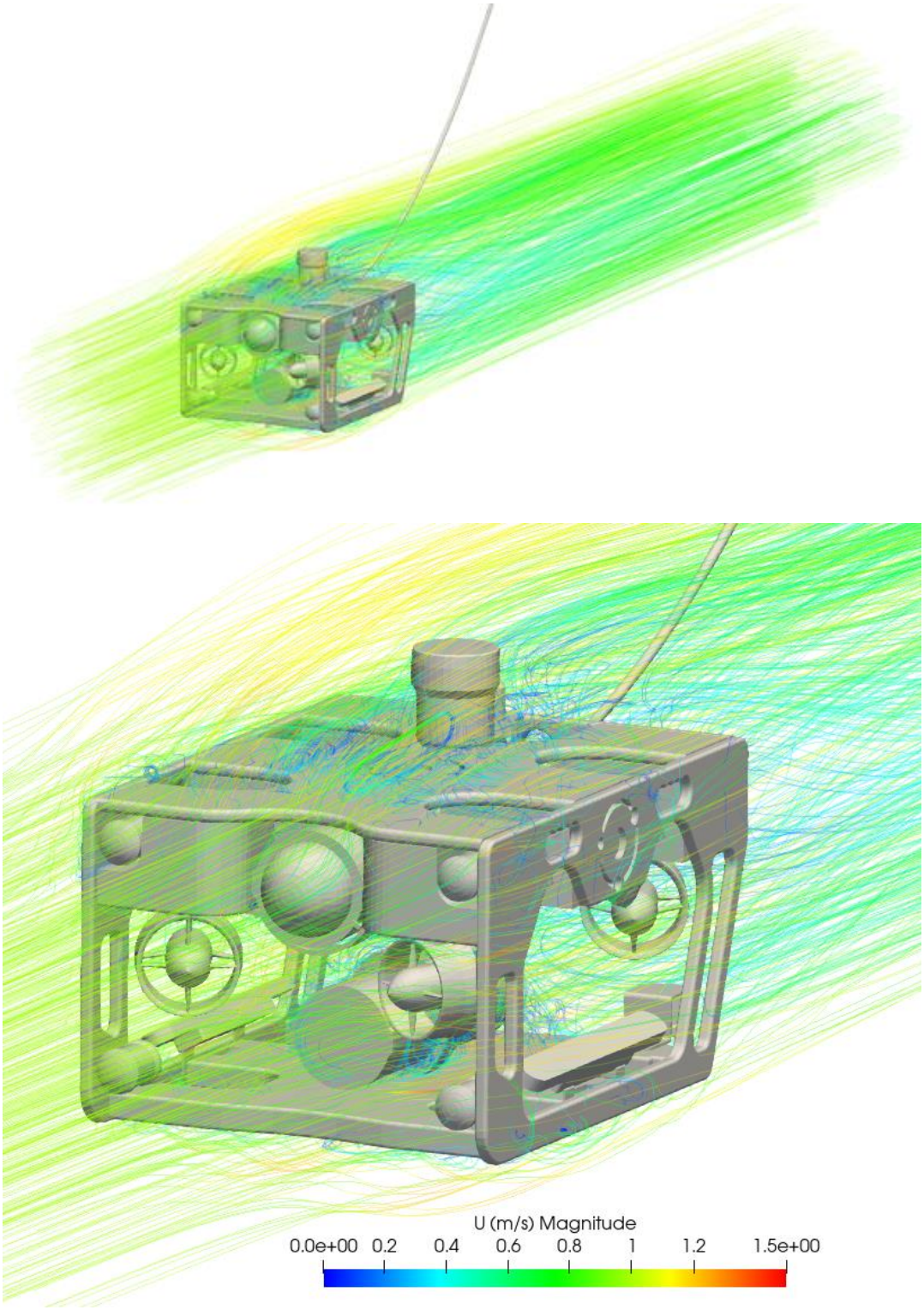


Figura 105. *Velocidad de las líneas de corriente esférica a través de Sibiu Pro.*

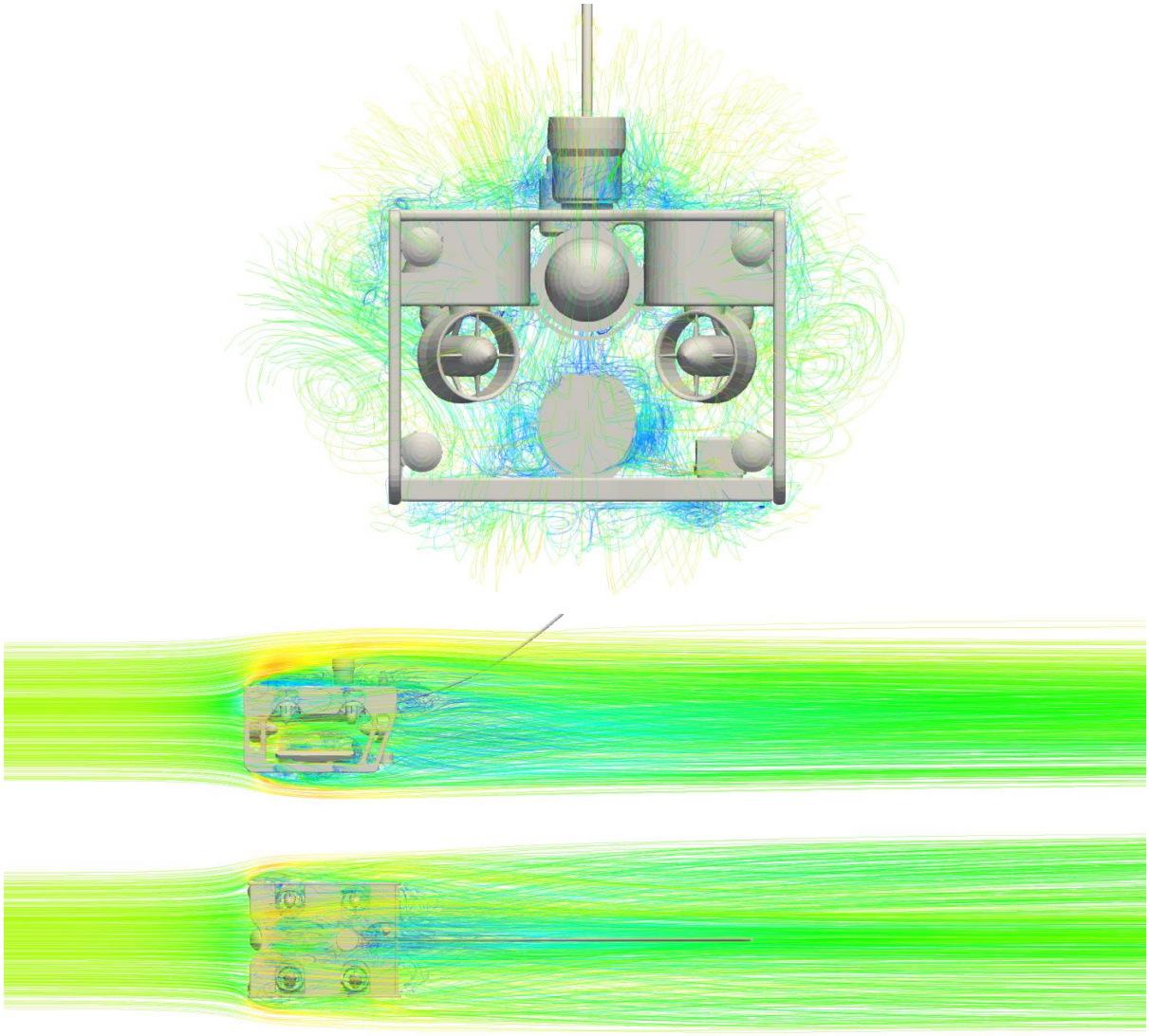


Figura 106. *Vistas de las líneas de corriente esférica a través del Sibiu Pro.*

Las Figuras 105 y 106 representan la trayectoria y velocidad de todo el flujo que incide sobre Sibiu Pro, ya sea interior o exteriormente. A diferencia de las líneas de corriente vistas anteriormente, en la Figuras 105 y 106 se puede apreciar flujo de interacción a muy baja velocidad en zonas que coinciden con el volumen generado por las nubes de presión. Este flujo se queda atrapado en los vórtices turbulentos desprendidos por la geometría poco afilada de Sibiu Pro.

La evolución a lo largo del tiempo de las líneas de corriente que atraviesan la esfera puede verse en el mismo enlace de video de YouTube comentado anteriormente (minuto 0:45).

https://www.youtube.com/watch?v=xHGl1xOGoZA&ab_channel=kttseadrones

5.3.2. Velocidad del flujo

El filtro *Slice* de ParaView® permite visualizar en 2 dimensiones el campo de velocidades del fluido que atraviesa un cuerpo, mostrándose la estela que genera en su avance.

En la Figura 107 se muestra la magnitud de la velocidad del fluido atraviesa Sibiu Pro en un plano vertical centrado en el mismo, con un estilo de visualización *Surface LIC* que permite apreciar la generación de vórtices.

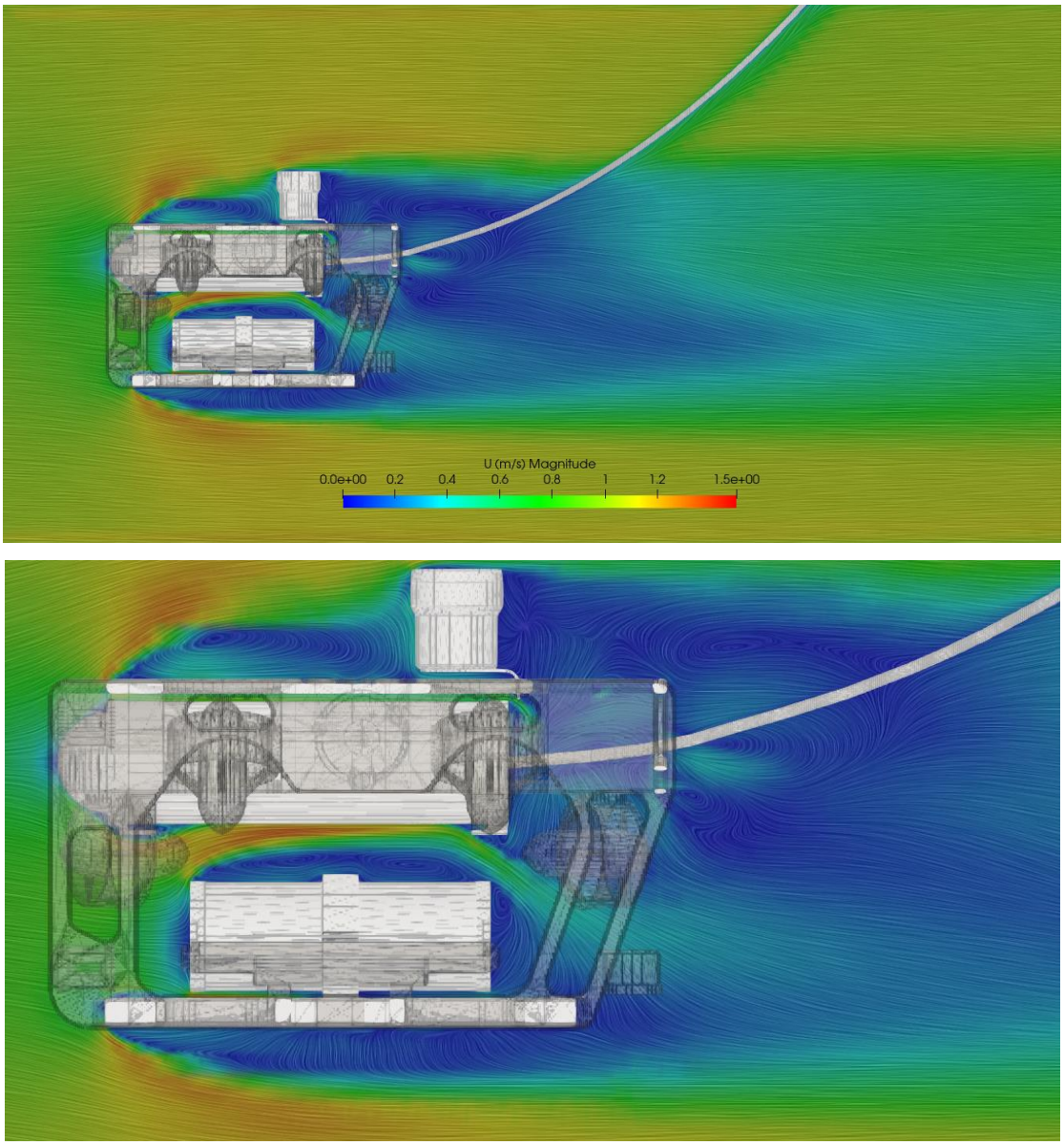


Figura 107. *Slice vertical de velocidad del flujo centrado a través de Sibiu Pro.*

De igual manera, en la Figura 108 se muestra la magnitud de la velocidad del fluido atraviesa Sibiu Pro en un plano horizontal a la altura del cilindro de electrónica.

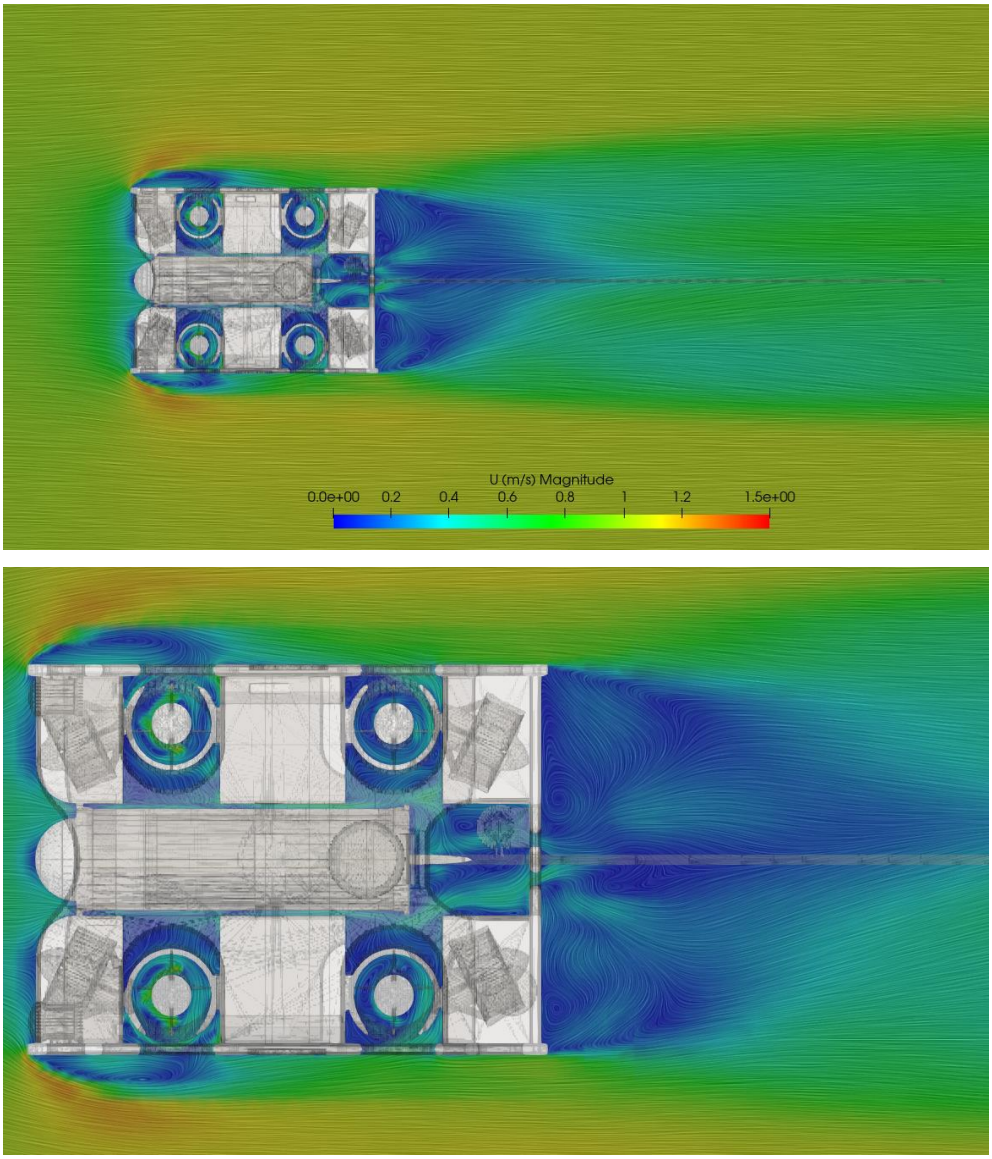


Figura 108. *Slice horizontal de velocidad del flujo sobre el cilindro de electrónica.*

La visualización que aportan ambas Figuras 107 y 108 es complementaria a las de las líneas de corriente mostradas previamente, ya que en estas si se aprecia el decremento de velocidad del fluido en su interacción con el ROV.

Además, se verifica la generación y desprendimiento de vórtices turbulentos en la región correspondiente con las nubes de presión vistas en apartados anteriores.

La evolución a lo largo del tiempo del campo de velocidad vertical y horizontal con el desprendimiento de vórtices mostrado puede verse en el mismo enlace de video de YouTube comentado anteriormente (minuto 1:20).

https://www.youtube.com/watch?v=xHGl1xOGoZA&ab_channel=kttseadrones

5.3.3. Presión del flujo

De igual manera que para la velocidad, el filtro *Slice* permite visualizar en 2 dimensiones el campo de presiones del flujo que somete a un cuerpo.

En la Figura 109 se aprecia la magnitud de la presión del fluido que interacciona con Sibiu Pro en un plano vertical centrado en el mismo, con una combinación de los filtros *Slice* y *Contour* para mostrar las isobaras.

Tal como se comentó en apartados anteriores, las presiones más altas ocurren donde el fluido incide sobre caras frontales planas y perpendiculares al sentido del flujo. Durante el movimiento de avance de Sibiu Pro el fluido alcanza presiones de hasta 560 Pa sobre la cara frontal de la batería. Mientras tanto, la presión de incidencia sobre el cristal de protección es algo menor debido a su forma semiesférica (color rojo menos intenso).

Del mismo modo, se generan volúmenes de fluido de presión negativa en superficies donde comienza la separación del flujo, tal como se mostró previamente. Esto ocurre en la transición de las caras frontales con el resto de caras de la batería, el sonar 360, la tapa inferior y la tapa superior, alcanzándose valores de hasta -850 Pa en este último.

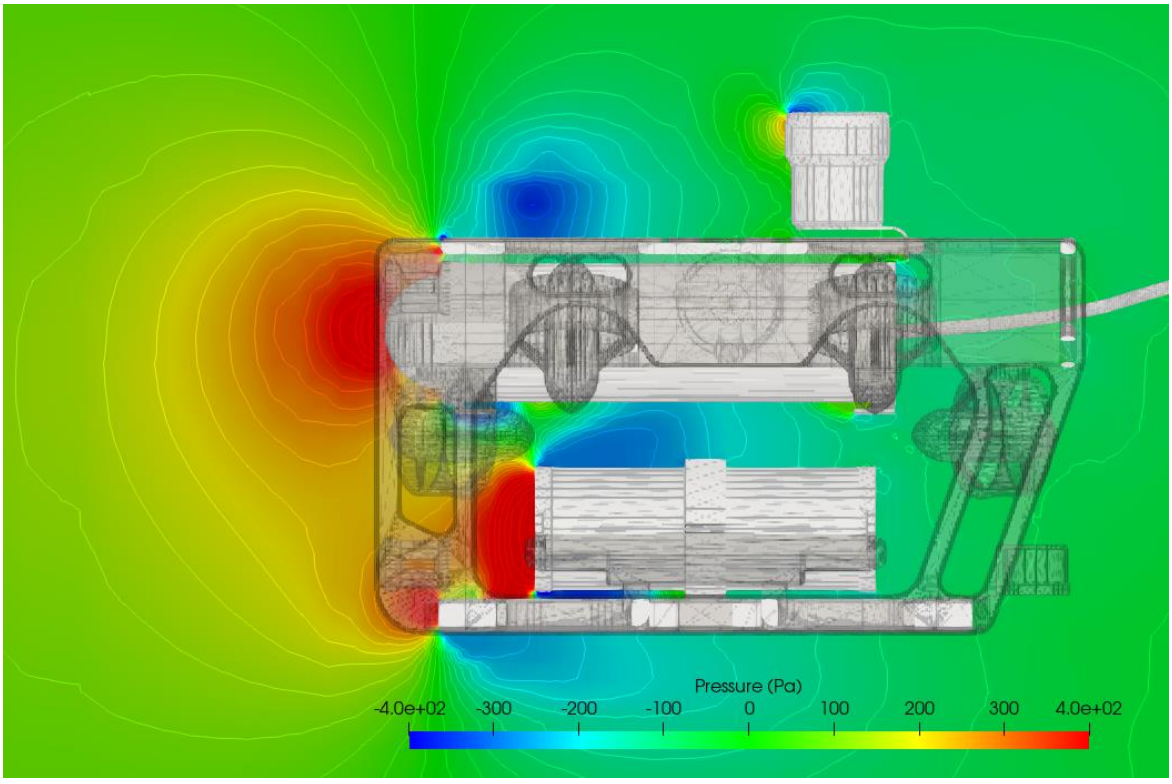


Figura 109. *Slice vertical de presión del flujo centrado a través de Sibiu Pro.*

De igual manera, en la Figura 110 se muestra la magnitud de la presión del fluido ejercida sobre Sibiu Pro en un plano horizontal a la altura del cilindro de electrónica.

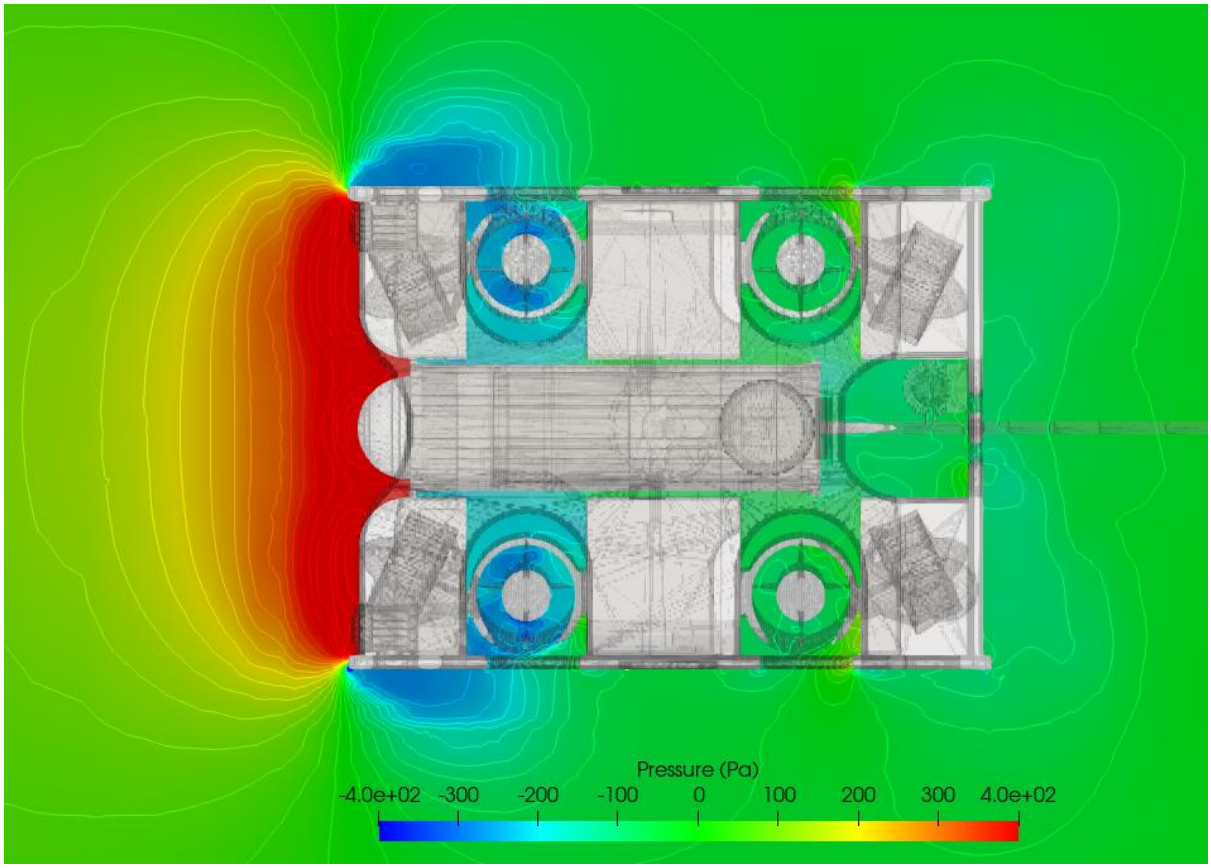


Figura 110. *Slice horizontal de presión del flujo sobre el cilindro de electrónica.*

En la Figura 110 se aprecia el gran volumen de altas presiones del fluido en su incidencia sobre el cilindro de electrónica, flotadores delanteros y chasis laterales.

La evolución a lo largo del tiempo del campo de presiones vertical y horizontal mostrado puede verse en el mismo enlace de video de YouTube comentado anteriormente (minuto 0:57).

https://www.youtube.com/watch?v=xHGl1xOGoZA&ab_channel=kttseadrones

En este se muestra como desde la superficie donde comienza la separación de flujo se desprenden volúmenes de flujo a baja presión, con la consiguiente generación de vórtices. Estos volúmenes flujo de baja presión desaparecen en su interacción con el flujo ya separado a mayor velocidad.

5.4. Validación experimental

En este apartado se compara de manera cualitativa los resultados simulados y experimentales de las líneas de corriente verticales centradas y estela de flujo generada por la interacción flujo-ROV.

En la Figura 111 se aprecia la similitud de separación de flujo vertical centrada de simulación y experimental. Ambos procedimientos resultan en tres trayectorias aceleradas del flujo: superior, inferior y central (más acusadas para los dos primeros). Tal como se ha comentado en apartados anteriores, la magnitud de la separación de flujo se debe principalmente a la incidencia del mismo sobre las caras frontales planas de Sibiu Pro.

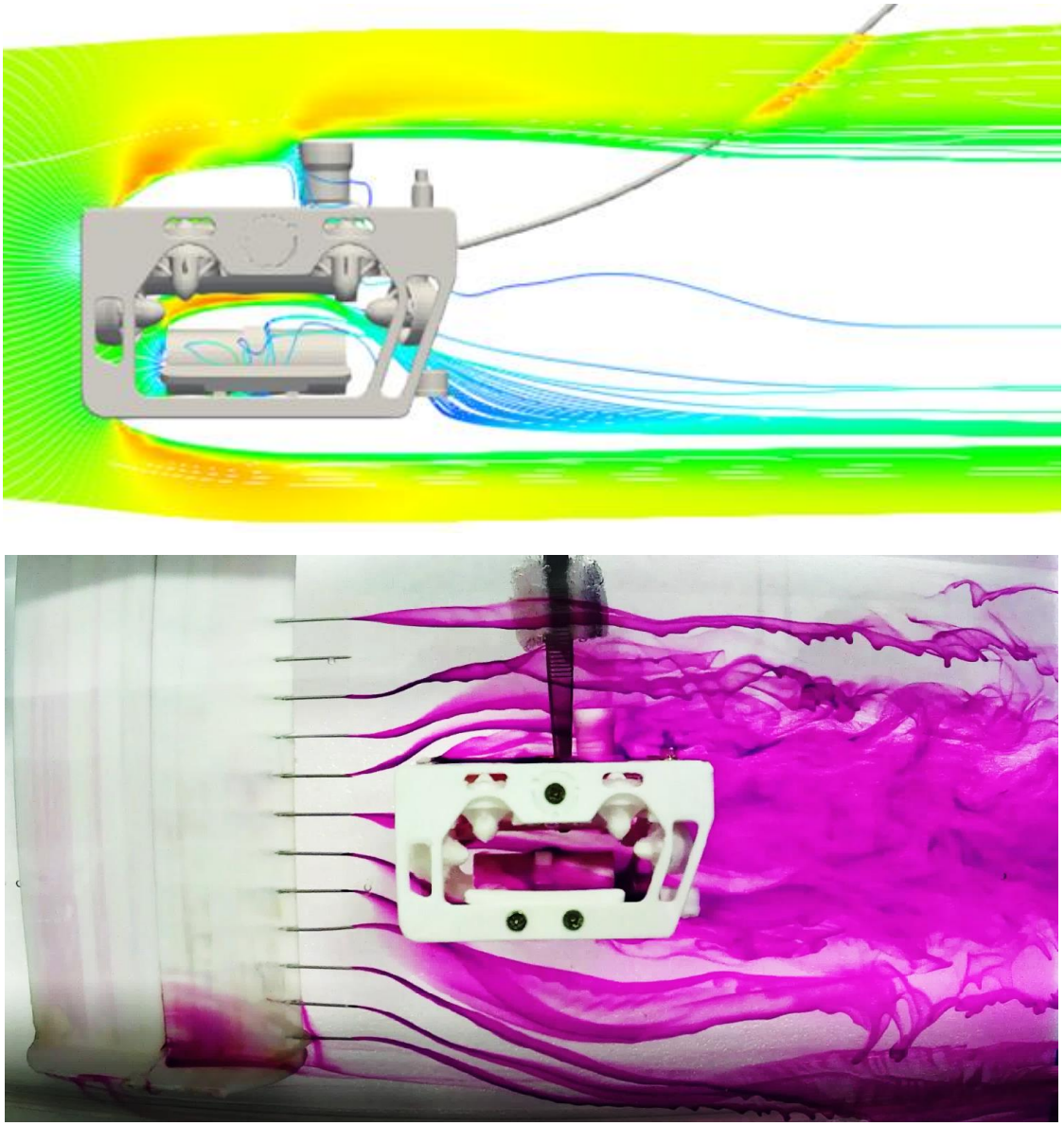


Figura 111. *Validación experimental de las líneas de corriente verticales centradas.*

La estela de flujo simulado se corresponde experimentalmente con la mezcla y difuminado las líneas del colorante que se forma en la parte trasera del mismo, donde este queda frenado y retenido en los vórtices. De igual forma, se puede comprobar como el colorante queda adherido superficialmente al ROV debido a la fricción superficial que este genera sobre el flujo. Este fenómeno se muestra de manera más clara en la Figura 112.

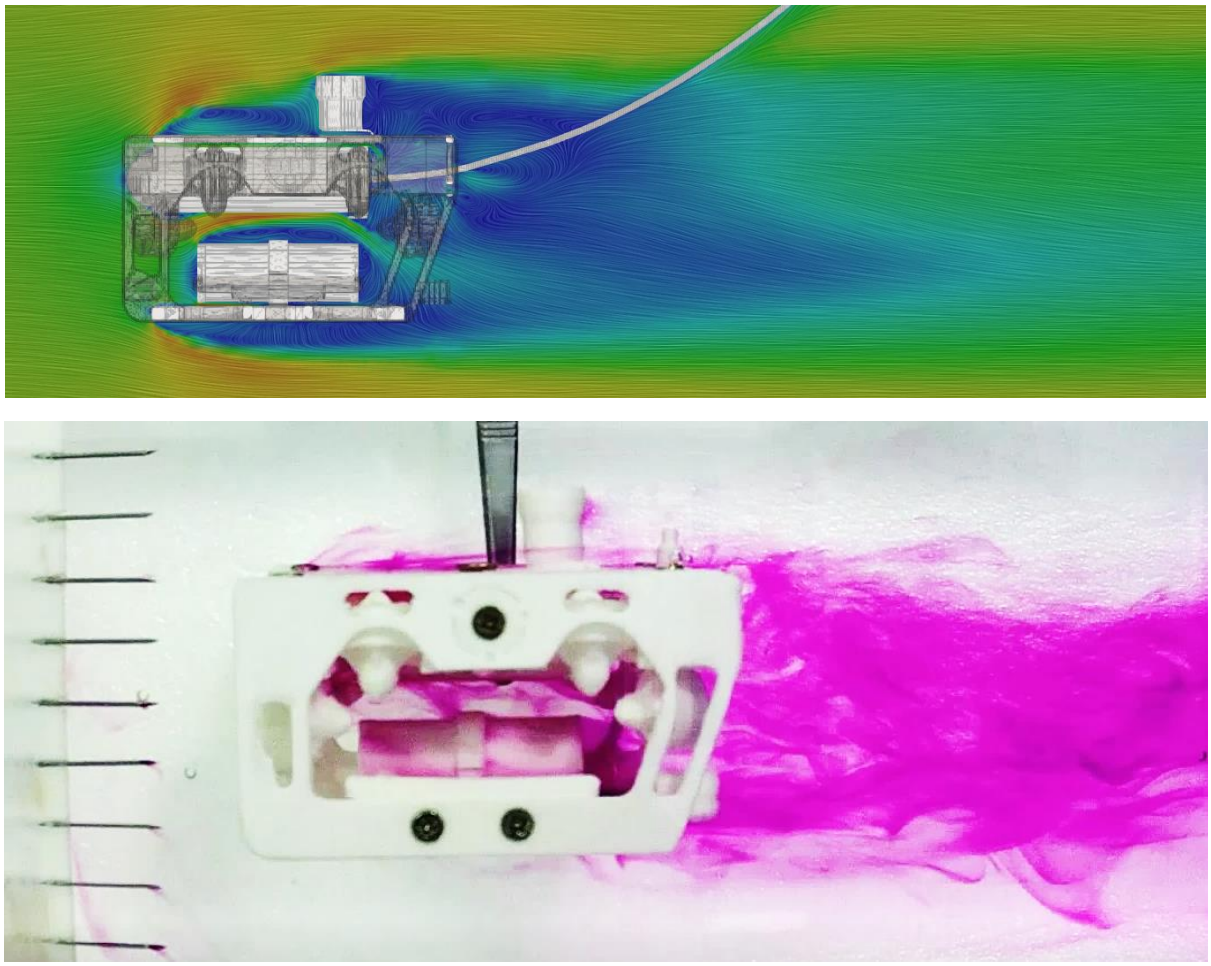


Figura 112. *Validación experimental de la estela del flujo.*

La evolución a lo largo del tiempo de las líneas de corriente verticales y la estela de flujo experimental puede verse en el mismo enlace de video de YouTube comentado anteriormente (minuto 1:50).

https://www.youtube.com/watch?v=xHGl1xOGoZA&ab_channel=kttseadrones

Capítulo 6
Conclusiones

En este estudio se ha propuesto una metodología CFD y experimental para conocer el comportamiento hidrodinámico de un vehículo submarino de tipo ROV.

Para el análisis por métodos numéricos se han utlizado las herramientas de simulación y postprocesado CFD open-source de OpenFOAM® y ParaView®. En la simulación se hace necesaria la ejecución de modelos de turbulencia basados en las ecuaciones de Navier-Stokes promedidas por Reynolds (RANS), debido a la alta complejidad geométrica del vehículo en cuestión y al coste computacional que ello supone. El procedimiento seguido puede servir como guía metodológica para iniciarse en estudios CFD aplicados en vehículos submarinos mediante las herramientas open-source mencionadas.

Para el análisis experimental se ha introducido un modelo a escala 5/70 del ROV en el canal de flujo del laboratorio de Mecánica de Fluidos. Para generar las líneas de corriente verticales centradas se ha diseñado un colector con agujas insertadas por donde fluye un colorante de permanganato potásico diluido. Tanto el ROV a escala como el colector han sido fabricados mediante impresora 3D.

Los dos procedimientos seguidos han necesitado previamente del manejo de herramientas CAD donde se modelen y ensamblen cada una de las piezas del submarino.

Los resultados simulados obtenidos permiten calcular los coeficientes y fuerzas hidrodinámicas de cada una de sus componentes para cada uno de los movimientos principales, y analizar la interacción avance del ROV sobre el flujo y viceversa. Tras comparar estos resultados cuantitativos en el ROV Sibiu Pro con los obtenidos por Li *et al.* (2020) en el BlueROV2 y analizar geométricamente ambos drones submarinos, se puede entonces constatar la validez del procedimiento de simulación.

Con respecto a la comparación de resultados simulados y experimentales, debido a la similitud entre ambos se puede concluir que para la estimación de las líneas de corriente pueden sustituirse los ensayos en canales de experiencias hidrodinámicas por procedimientos de simulación. Este hecho repercute especialmente en las fases preliminares de diseño de vehículos acuáticos, momento en el que introducir alguna modificación geométrica del modelo supone un alto costo de experimentación.

Para finalizar, es de mencionar que la configuración de archivos de OpenFOAM® del caso SibiuPro abre la puerta a nuevos estudios CFD del mismo donde se realicen cambios de componentes que provocarían distintas perturbaciones. Para ello simplemente habría que modificar el modelo CAD y la longitud característica del ROV si se viese alterada.

Posibles trabajos futuros del caso en cuestión podrían enfocarse en el diseño de deflectores que desvíen el flujo para reducir la fuerza de arrastre que se ejerce sobre las caras frontales planas de Sibiu Pro. Esta modificación supondría una mejora significativa del comportamiento hidrodinámico de ROV, disminuyendo su consumo de batería y aumentando su maniobrabilidad y control.

Otra posible línea de investigación se puede centrar en la verificación experimental de las fuerzas y coeficientes hidrodinámicos mediante la instalación y utilización de células de carga, tal como se desarrolla en Gabl *et al.* (2020). El complejo sistema que en éste se desarrolla, sería la manera más fiable de saber si los resultados obtenidos mediante CFD de verdad se ajustan a la realidad.

Anexos

Anexo I

OpenFOAM®

OpenFOAM® (Open Source Field Operation And Manipulation) es el software líder de código abierto y gratuito para dinámica de fluidos computacional (CFD), propiedad de OpenFOAM Foundation y distribuido exclusivamente bajo la Licencia Pública General (GPL). La GPL brinda a los usuarios la libertad de modificar y redistribuir el software y una garantía de uso gratuito continuo, dentro de los términos de la licencia.

OpenFOAM® es desarrollado y mantenido por personas que contribuyen con su trabajo al proyecto, con el apoyo y consentimiento de las empresas que los emplean. El proyecto opera a través de una red de confianza entre las personas, donde se otorga mayor autoridad a los contribuyentes que constantemente producen un trabajo de alta calidad y demuestran un compromiso a largo plazo.

En el centro de OpenFOAM® se encuentra un kit de desarrollo de software ("devkit") único y altamente extensible para CFD, que consta de 1 millón de líneas de código C ++. OpenFOAM® incluye cientos de robustas aplicaciones CFD creadas a partir del devkit, que se pueden ampliar y personalizar de forma rápida y cómoda. Las aplicaciones se utilizan para crear simulaciones CFD por parte de la industria, el mundo académico y los institutos gubernamentales y de investigación en todos los campos de la ingeniería y la ciencia (OpenFOAM Foundation, 2011).

Las aplicaciones de OpenFOAM® van más allá de simulaciones de CFD para flujos compresibles e incompresibles. Entre estas aplicaciones se encuentra la resolución de problemas de ingeniería tales como transferencia de calor, combustión, análisis de tensión estructural por elementos finitos, electromagnetismo o fenómenos de transporte entre otros.

OpenFOAM® es en sí, un marco para desarrollar aplicaciones ejecutables que utilizan la funcionalidad de paquete contenida en una colección de aproximadamente 100 bibliotecas C ++. OpenFOAM® cuenta con aproximadamente 250 aplicaciones prediseñadas que se dividen en dos categorías: solucionadores, cada uno de los cuales está diseñado para resolver un problema específico en la mecánica de fluidos (o continuos); y utilidades, que están diseñadas para realizar tareas que involucran la manipulación de datos. Los solucionadores de OpenFOAM® cubren una amplia gama de problemas en dinámica de fluidos. Los usuarios pueden ampliar la colección de solucionadores, utilidades y bibliotecas en OpenFOAM®, utilizando algunos conocimientos previos del método subyacente, la física y las técnicas de programación involucradas. OpenFOAM® se suministra con entornos de pre y postprocesado de datos. La interfaz para el pre y postprocesamiento son en sí mismas utilidades de OpenFOAM®, lo que garantiza un manejo de datos consistente en todos los entornos.

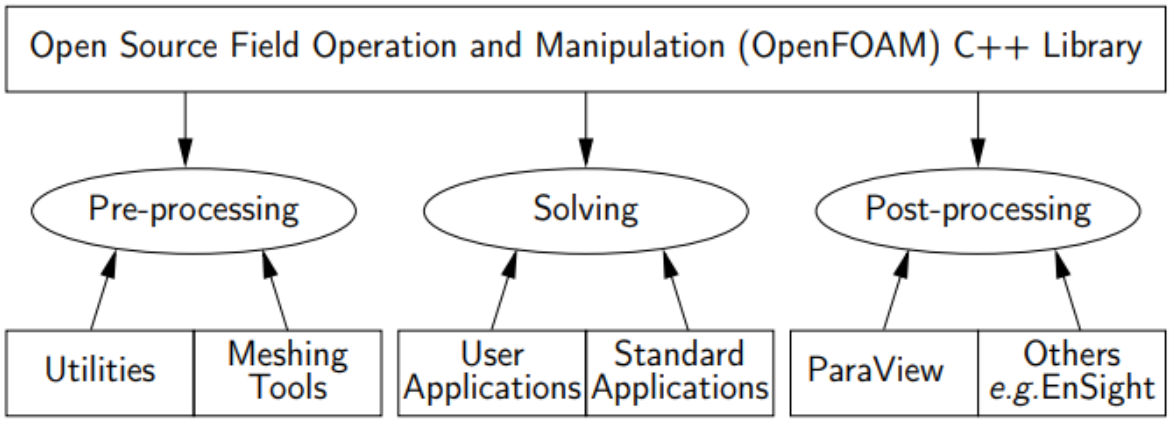


Figura 113. *Visión general de la estructura de OpenFOAM®.*

Quizás en este punto te haces la pregunta: "¿Por qué debería invertir mi tiempo en aprender OpenFOAM®?" Bueno, hay muchos argumentos y estos son algunos de ellos (OpenFOAM Wiki, 2021):

- OpenFOAM® es uno de los 3 programas de CFD más utilizados.
- OpenFOAM® es uno de los 5 software más utilizados en HPC.
- OpenFOAM® cuenta con la confianza de la industria, los centros de I+D y las universidades.
- OpenFOAM® ofrece la oportunidad de una alta paralelización.
- OpenFOAM® es de código abierto, gratuito y con un código fuente abierto.
- Puede implementar sus propios modelos, si lo desea.
- Hay miles de usuarios en todo el mundo para la investigación y la ingeniería de producción.
- Hay mucha ayuda de usuarios experimentados en la web.
- Hay reuniones de usuarios locales en varios países.

Figura 114. *Logo de OpenFOAM®.*

Anexo II

ParaView®

ParaView® es una potente aplicación de visualización y análisis de datos multiplataforma de código abierto que se encuentra empaquetada en la instalación de OpenFOAM para el postprocesado de simulaciones. Los usuarios de ParaView® pueden crear visualizaciones rápidamente para analizar sus datos utilizando técnicas cualitativas y cuantitativas. La exploración de datos se puede realizar de forma interactiva en 3D o mediante programación utilizando las capacidades de procesamiento por lotes de ParaView®. ParaView® fue desarrollado para analizar conjuntos de datos extremadamente grandes utilizando recursos informáticos de memoria distribuida. Se puede ejecutar en supercomputadoras para analizar conjuntos de datos de petaescala, así como en computadoras portátiles para datos más pequeños.

El código base de ParaView® está diseñado de tal manera que todos sus componentes se pueden reutilizar para desarrollar rápidamente aplicaciones verticales. Esta flexibilidad permite a los desarrolladores de ParaView® desarrollar rápidamente aplicaciones que tienen una funcionalidad específica para un dominio de problema específico. ParaView® se ejecuta en sistemas de procesador único y paralelo de memoria distribuida y compartida. Se ha implementado con éxito en Windows, Mac OS X, Linux, SGI, IBM Blue Gene, Cray y varias estaciones de trabajo, clústeres y supercomputadoras Unix. Bajo el capó, ParaView® usa Visualization Toolkit (VTK) como el motor de procesamiento y renderizado de datos y tiene una interfaz de usuario escrita usando Qt® Los objetivos del equipo de ParaView® incluyen lo siguiente.

- Desarrolle una aplicación de visualización multiplataforma de código abierto.
- Admite modelos de computación distribuidos para procesar grandes conjuntos de datos.
- Cree una interfaz de usuario abierta, flexible e intuitiva.
- Desarrollar una arquitectura extensible basada en estándares abiertos. (Paraview, 2007)

Figura 115. *Logo de ParaView®.*

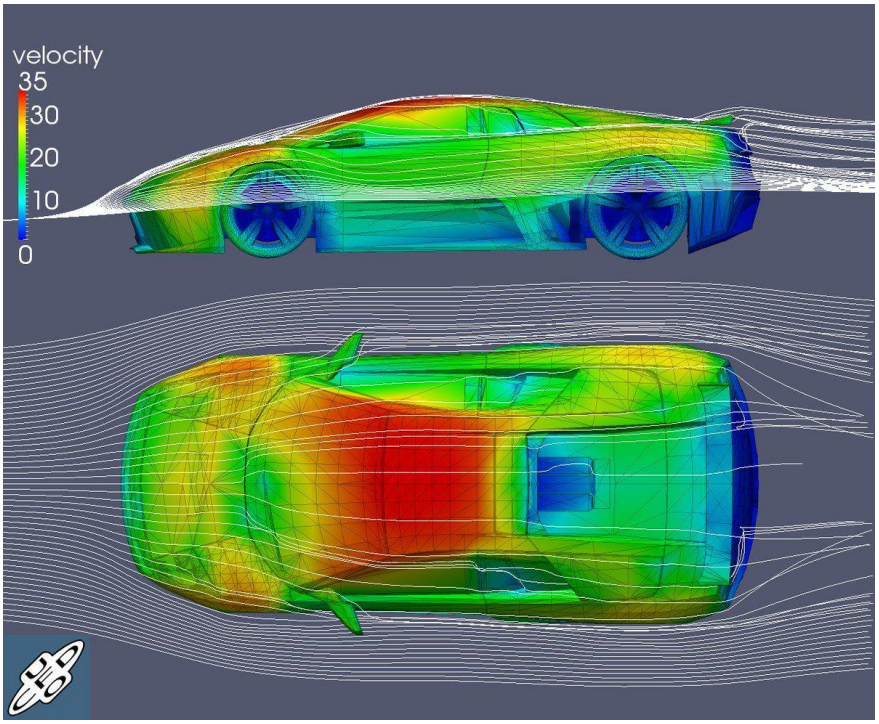


Figura 116. *Ejemplo de simulación CFD en ParaView®.*

Anexo III
Manuales de instalación en Windows 10

En este anexo se ofrece una guía de instalación en español del software y demás herramientas informáticas utilizadas para la simulación hidrodinámica del ROV. Esta guía comprende únicamente la instalación de las herramientas mencionadas en Windows 10 - 64 bits, que es el sistema operativo en el cual se desarrolló todo el trabajo.

III.1 Instalación de Ubuntu 18.04 LTS

Ubuntu® es un sistema operativo de software libre y código abierto. Es una distribución de Linux®.

Figura 117. *Logo de Ubuntu®.*

La descarga, instalación y manejo de OpenFOAM® en Windows 10 se realiza en la terminal de Ubuntu mediante comandos de Linux® y es esta la razón de instalar Ubuntu® en Windows 10.

La versión de Ubuntu® que se descargará es Ubuntu 18.04 LTS, la cual es la última versión en la cual se puede instalar OpenFOAM 5.0

Pasos de instalación:

1. Abrir Microsoft Store mediante su búsqueda en el menú de inicio de Windows 10.
2. Dentro de Microsoft Store se escribe *Ubuntu 18.04 LTS* en la barra de búsqueda situada arriba a la derecha de la ventana y se selecciona.
3. Se pulsa en instalar para proceder a su descarga.
4. Una vez se ha completado la descarga se abre la terminal Ubuntu 18.04 LTS en el menú de inicio de Windows 10. En este momento se procede a instalarse automáticamente.
5. El terminal pide introducir un nuevo nombre de usuario para Ubuntu.
6. Se introduce y *Enter*.
7. El terminal pide introducir una nueva contraseña para ese usuario. Se introduce y *Enter*.
8. La contraseña no aparecerá por pantalla mientras se está escribiendo.
9. El terminal pide reescribir la nueva contraseña. Se introduce y *Enter*.
10. Instalación completada.

III.2 Instalación del X Server. VcXsrv: XLaunch

Uno de los inconvenientes de instalar Ubuntu® en Windows es que carece de la Interfaz Gráfica de Linux®, simplemente se cuenta con el terminal de Ubuntu®. Es por ello que, como se verá, no se permita abrir aplicaciones gráficas como ParaView® mediante la terminal. De tal forma, se hace necesario instalar VcXsrv, un Windows X Server que haga de medio para abrir aplicaciones gráficas.

Pasos de instalación:

1. Abrir el siguiente enlace web: https://sourceforge.net/projects/vcxsrv/
2. Pinchar en *Download* para descargar.
3. Una vez la descarga está completada, se busca el instalador en la carpeta de descargas del Windows 10 y se ejecuta.
4. Preguntará: ¿Quieres permitir que esta aplicación de un anunciante desconocido haga cambios en el dispositivo? Se pulsa en Sí.
5. Se sigue los siguientes pasos de instalación:

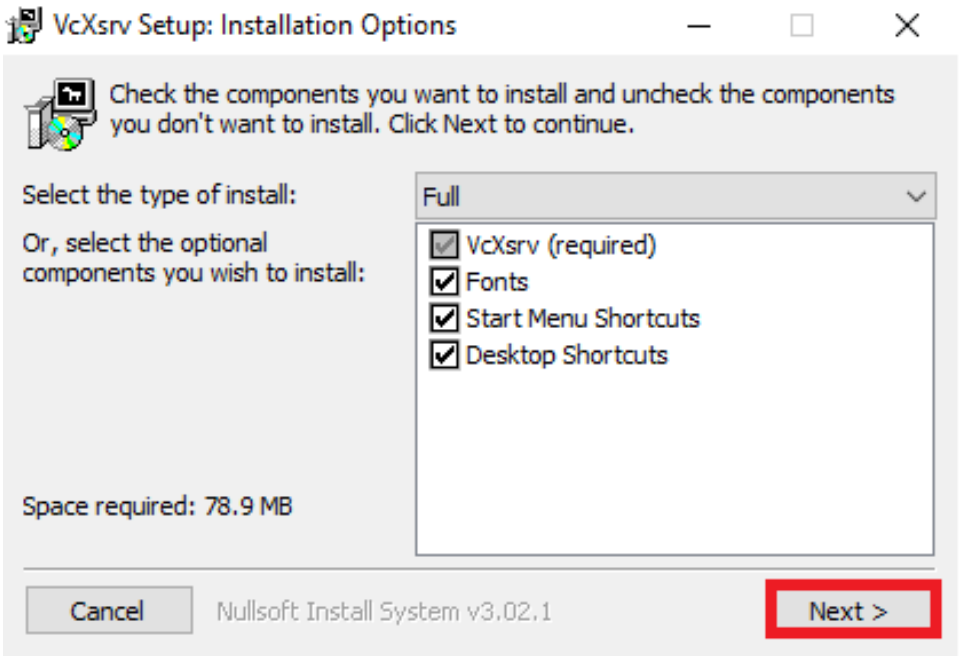


Figura 118. *Instalación del X Server. VcXsrv: XLaunch. Paso 5.1.*

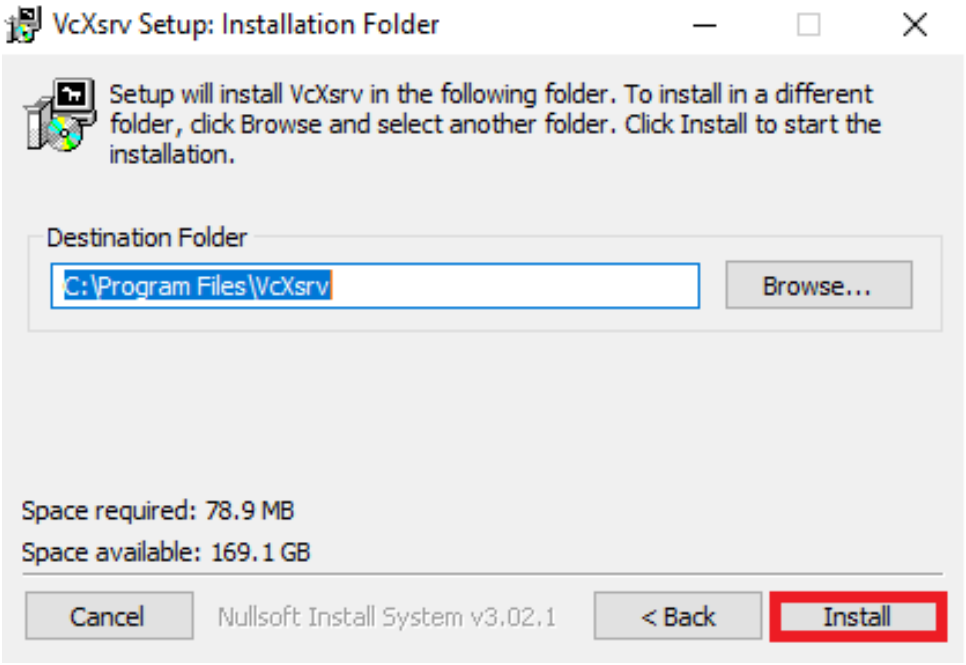


Figura 119. *Instalación del X Server. VcXsrv: XLaunch. Paso 5.2.*

La instalación comienza.

6. Una vez se haya completado la instalación, se crea en acceso directo en el escritorio llamado XLaunch, que es el que se usará para abrir ParaView®.

III.3 Instalación de OpenFOAM 5.0 & ParaView 5.4.0

Los pasos de instalación de OpenFOAM 5.0 y ParaView 5.4.0 se detallan en inglés en la siguiente página web oficial de OpenFOAM®:

https://openfoam.org/download/5-0-ubuntu/

Igualmente se explican los siguientes pasos de instalación en español con los problemas de instalación que surgieron durante el proceso:

1. Se abre el terminal Ubuntu 18.04 LTS

Los pasos siguientes requieren de conexión a internet ininterrumpida para que los paquetes de instalación se descarguen correctamente.

2. Agregar dl.openfoam.org a la lista de repositorios de software.

Se copia y pega en el terminal las siguientes instrucciones:

```
sudo sh -c "wget -O - http://dl.openfoam.org/gpg.key | apt-key add -"
```

Una vez se ha copiado, basta con pulsar *click derecho* sobre el terminal para pegar. Entonces se pulsa *Enter*. Pedirá además introducir la contraseña del usuario de Ubuntu®. Se introduce y *Enter*. Mientras no aparezca el nombre de usuario en verde como última línea, significa que la ejecución del comando no ha terminado. Para cancelar una instrucción se pulsa *Ctrl+C* durante la ejecución.

La terminal pondrá un mensaje final de *OK* si se ha realizado correctamente.

```
sudo add-apt-repository http://dl.openfoam.org/ubuntu
```

La terminal pondrá un mensaje final de *Reading package lists... Done* si se ha realizado correctamente.

3. Actualizar la lista de paquetes de apt.

Se copia y pega en el terminal la siguiente instrucción:

```
sudo apt-get update
```

La terminal pondrá un mensaje final de *Reading package lists... Done* si se ha realizado correctamente.

4. Instala OpenFOAM 5.0 y paraviewopenfoam54 como dependencia.

Se copia y pega en el terminal la siguiente instrucción:

```
sudo apt-get -y install openfoam5
```

5. Actualizar los paquetes de instalación.

Se copia y pega en el terminal las siguientes instrucciones:

```
sudo apt-get update
```

La terminal pondrá un mensaje final de *Reading package lists... Done* si se ha realizado correctamente.

```
sudo apt-get upgrade
```

La terminal escribirá una lista por pantalla de los paquetes que se actualizarán y preguntará si se quiere continuar. Se escribe *Y* para seleccionar que sí se quiere continuar.

Se actualizan los paquetes de instalación.

6. Instalar el comando gedit.

Se copia y pega en el terminal la siguiente instrucción:

```
sudo apt install gedit
```

La terminal escribirá una lista por pantalla de los paquetes que se actualizarán y preguntará si se quiere continuar. Se escribe *Y* para seleccionar que sí se quiere continuar.

7. Se ejecuta el X Server (XLaunch). Este paso puede hacerse antes.

Se abre el acceso directo en Windows 10 de XLaunch y se pulsa *Siguiente* tres veces hasta pulsar en *Finalizar* dejando seleccionadas las opciones que aparecen en las siguientes capturas:

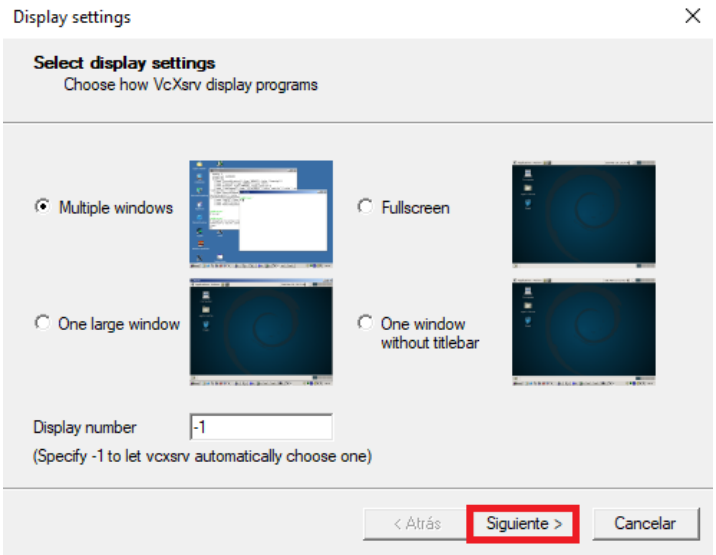


Figura 120. *Instalación de OpenFOAM 5.0 & ParaView 5.4.0. Paso 7.1.*

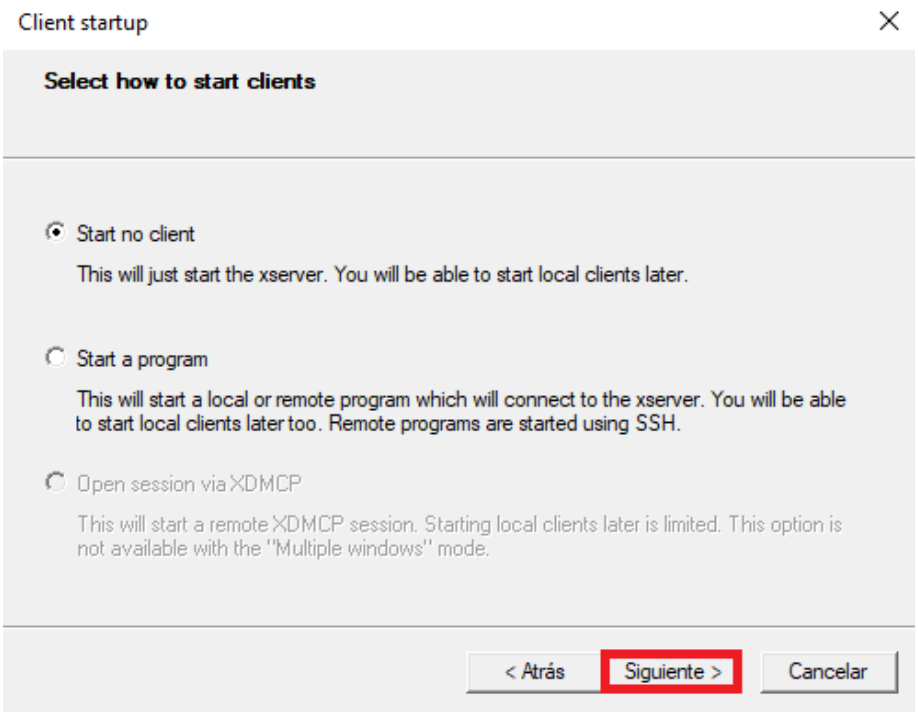


Figura 121. *Instalación de OpenFOAM 5.0 & ParaView 5.4.0. Paso 7.2.*

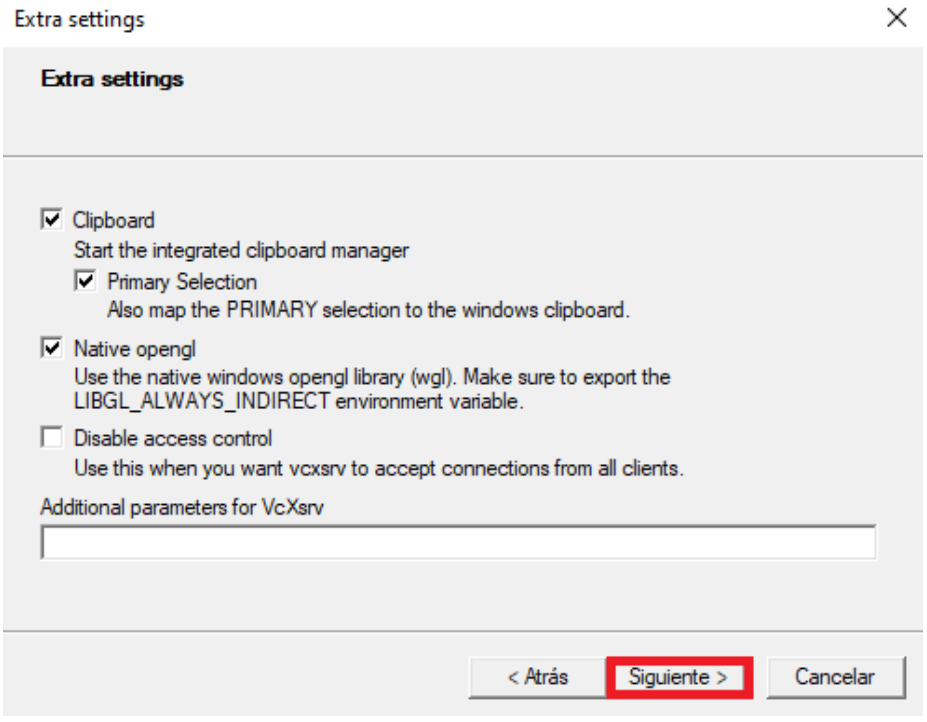


Figura 122. *Instalación de OpenFOAM 5.0 & ParaView 5.4.0. Paso 7.3.*

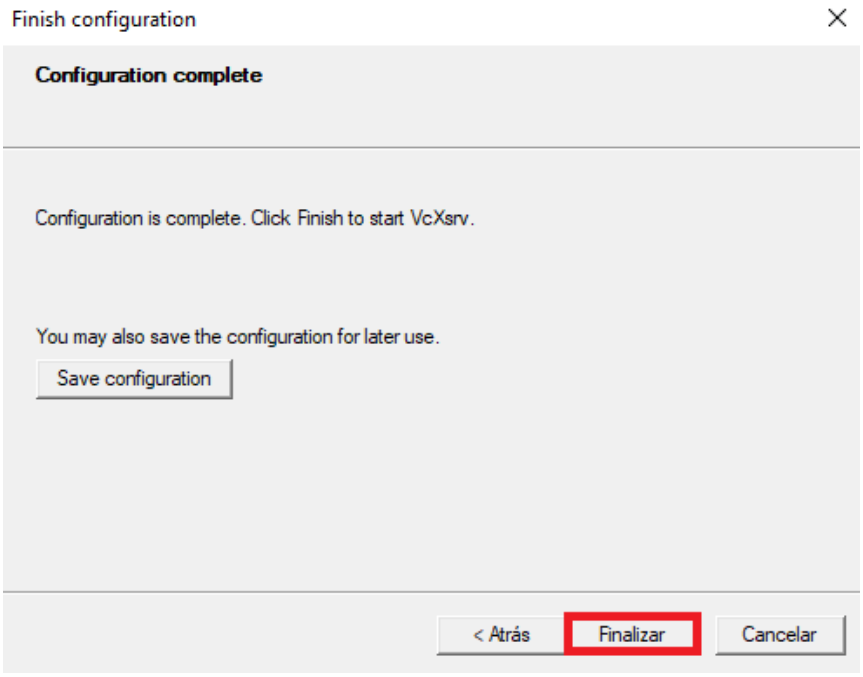


Figura 123. *Instalación de OpenFOAM 5.0 & ParaView 5.4.0. Paso 7.4.*

8. Se conecta el X Server en Ubuntu.

Se copia y pega en el terminal la siguiente instrucción:

```
export DISPLAY=0:0
```

Para comprobar que los pasos 7 y 8 se han realizado correctamente se puede escribir el siguiente comando:

```
gedit
```

Si se abre una ventana con un documento en blanco significa que el X Server se ha ejecutado y conectado correctamente. De lo contrario, si aparece el siguiente mensaje de error, significa que el X Server no está conectado a Ubuntu®.

```
Unable to init server: Could not connect: Connection refused

(gedit:24502): Gtk-WARNING **: 11:46:43.826: cannot open display: 0:0
```

Figura 124. *Instalación de OpenFOAM 5.0 & ParaView 5.4.0. Paso 8.1.*

Estos pasos 7 y 8 son siempre necesarios cada vez que se quiera usar el comando de edición de texto *gedit* o abrir ParaView® desde Ubuntu® mediante el siguiente comando de OpenFOAM®:

```
paraFoam
```

9. Se abre el archivo de OpenFOAM® llamado *.bashrc*:

```
gedit ~/.bashrc
```

10. Tras la última línea del archivo *.bashrc* se copia y pega la siguiente línea de código, tal como se muestra en la siguiente captura:

```
source /opt/openfoam5/etc/bashrc
```

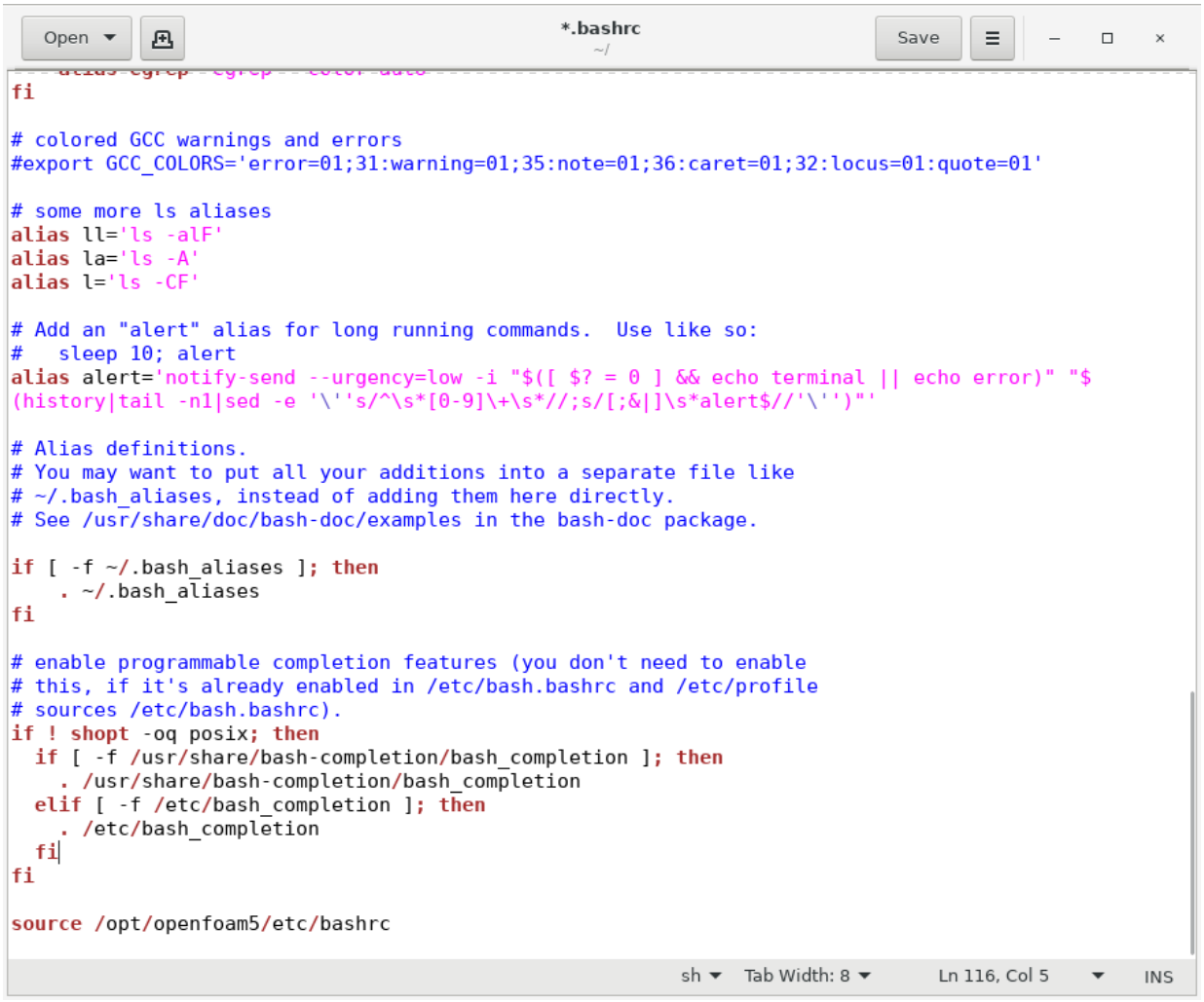


Figura 125. *Instalación de OpenFOAM 5.0 & ParaView 5.4.0. Paso 10.1.*

Tras pegar, se pulsa en *Save* y cerrar. Entonces se cierra y vuelve abrir el terminal de Ubuntu®.

III.4 Prueba de instalación de OpenFOAM 5.0 & ParaView 5.4.0

Se puede comprobar de manera rápida si la instalación se ha realizado correctamente introduciendo el siguiente comando de OpenFOAM®:

```
simpleFoam -help
```

Si aparece por pantalla los mensajes de la siguiente captura, significa que la instalación se completó correctamente:

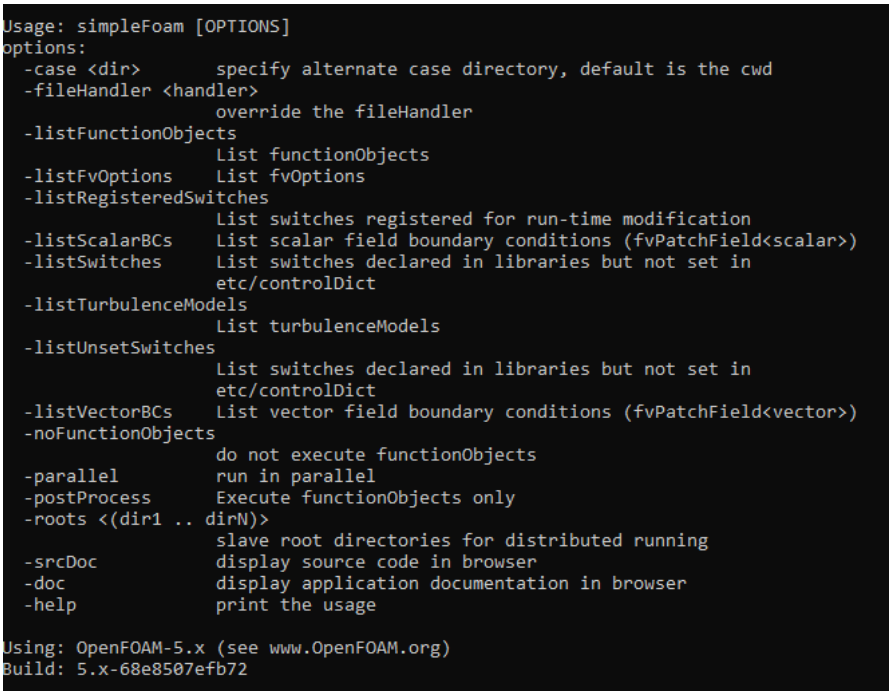


Figura 126. *Prueba de instalación de OpenFOAM 5.0 & ParaView 5.4.0.*

El comando muestra los requisitos que se deben especificar para que funcione el comando de OpenFOAM® llamado s*impleFoam*.

Si por el contrario, aparece un mensaje de error diciendo que no reconoce el comando, significa que algún paso de la instalación no se realizó correctamente o en el orden correcto.

Si durante la instalación ocurrió algún otro error, no comentado, la recomendación es intentar buscarlo en el siguiente enlace web, o bien copiar y pegar el error en algún navegador web para investigar en foros si algún usuario previamente tuvo el mismo error y si alguien ha conseguido solucionarlo de alguna determinada forma.

https://openfoam.org/download/5-0-ubuntu/

III.5 Simulación de prueba de OpenFOAM 5.0 & ParaView 5.4.0

Los pasos de la simulación de prueba de OpenFOAM 5.0 y ParaView 5.4.0 se detallan en inglés en la siguiente página web oficial de OpenFOAM®:

https://openfoam.org/download/5-0-ubuntu/

Igualmente se explican los siguientes pasos de simulación en español:

1. Crea el directorio $FOAM_RUN, donde colocar los casos de los tutoriales.

```
mkdir -p $FOAM_RUN
```

2. Accede al directorio $FOAM_RUN y copia y pega en este, el caso *pitzDaily.*

   Su ruta es *$FOAM_TUTORIALS/incompressible/simpleFoam/pitzDaily*

```
cd $FOAM_RUN
cp -r $FOAM_TUTORIALS/incompressible/simpleFoam/pitzDaily .
```

Nótese el espacio y punto del último comando para que la copia se realice correctamente y sin errores. Los tutoriales de OpenFOAM® están organizados según el tipo de fluido en directorios y según el solucionador en subdirectorios. Dentro de cada subdirectorio hay gran variedad de tutoriales.

3. Accede al caso *pitzDaily* desde $FOAM_RUN.

```
cd pitzDaily
```

4. Crea el mallado de *pitzDaily* con el comando básico de OpenFOAM llamado *blockMesh.*

```
blockMesh
```

5. Ejecuta el solucionador *simpleFoam* sobre el caso *pitzDaily* para correr la simulación.

```
simpleFoam
```

Se debe mostrar como la simulación converge en 287 instantes.

6. Abre en ParaView® el mallado de *pitzDaily* con los datos extraídos de la simulación para realizar el post-procesado de dichos datos. Se realiza mediante el comando *paraFoam.*

```
paraFoam
```

Se recuerda que se debe tener conectado el X Server antes de abrir ParaView® (ver puntos 7 y 8 de Instalación OpenFOAM 5.0 & ParaView 5.4.0)

III.6 Instalación de ParaView 5.8.1

A la hora del postprocesado de datos obtenidos en OpenFOAM® es recomendable (aunque no imprescindible) contar con alguna versión de ParaView® instalada aparte en Windows 10. Para ello se seguirán los siguientes pasos de instalación y descarga de ParaView 5.8.1, la cual es la última versión de ParaView® disponible en el momento de realización de este trabajo.

Pasos de instalación:

1. Abrir el siguiente enlace web: https://www.paraview.org/download/
2. Para la versión 5.8 de Windows se elige: ParaView-5.8.1-Windows-Python3.7-msvc2015-64bit.exe
3. Una vez la descarga se ha completado, se busca el instalador en la carpeta de descargas del Windows 10 y se ejecuta.
4. Preguntará: ¿Quieres permitir que esta aplicación de un anunciante desconocido haga cambios en el dispositivo? Se pulsa en Sí.
5. Se sigue los siguientes pasos de instalación:

Figura 127. *Instalación de ParaView 5.8.1. Paso 5.1.*

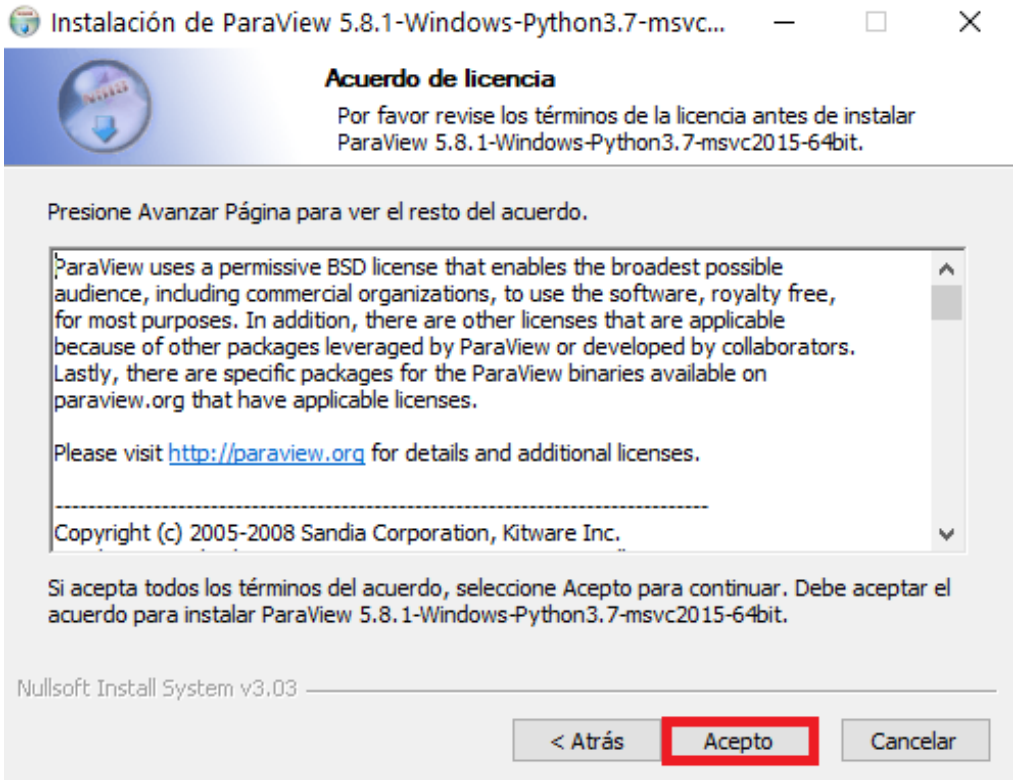


Figura 128. *Instalación de ParaView 5.8.1. Paso 5.2.*

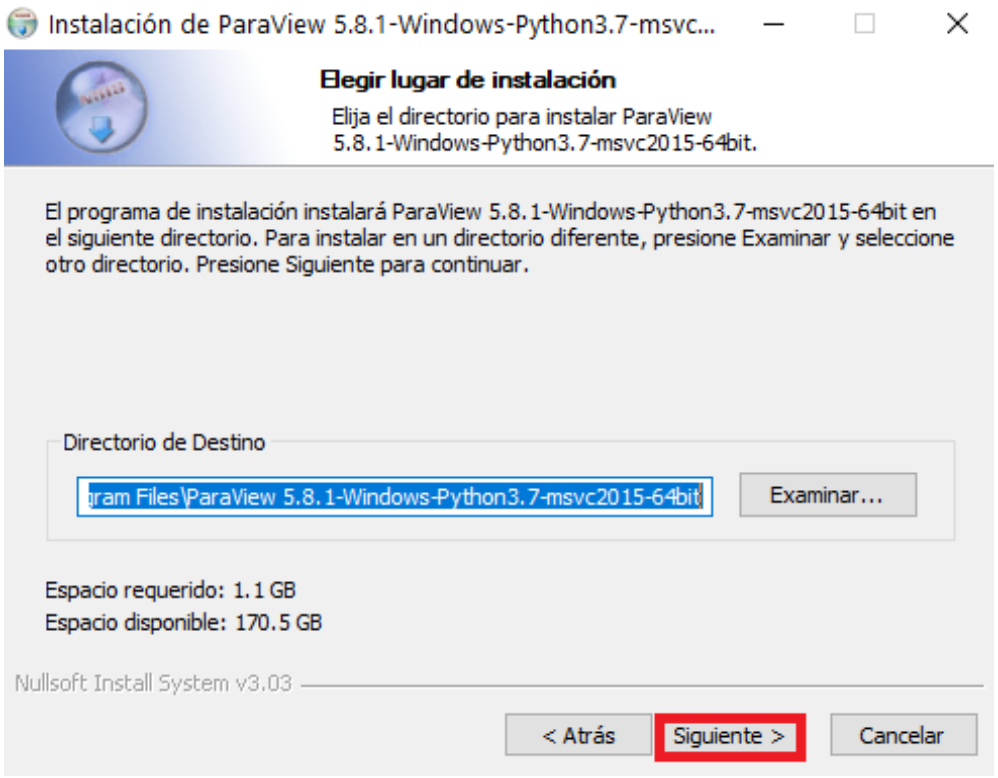


Figura 129. *Instalación de ParaView 5.8.1. Paso 5.3.*

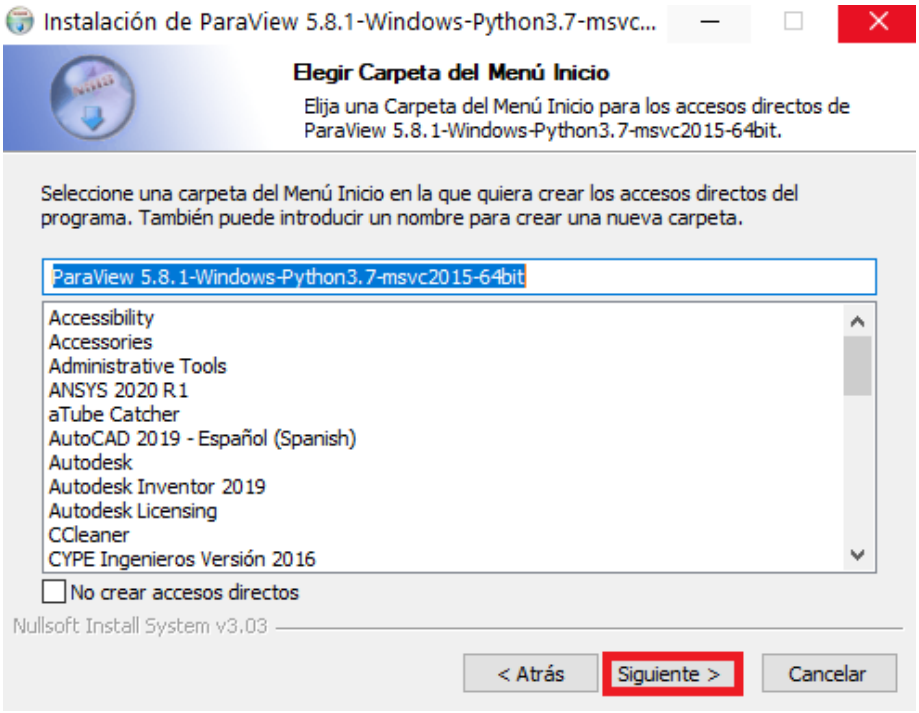


Figura 130. *Instalación de ParaView 5.8.1. Paso 5.4.*

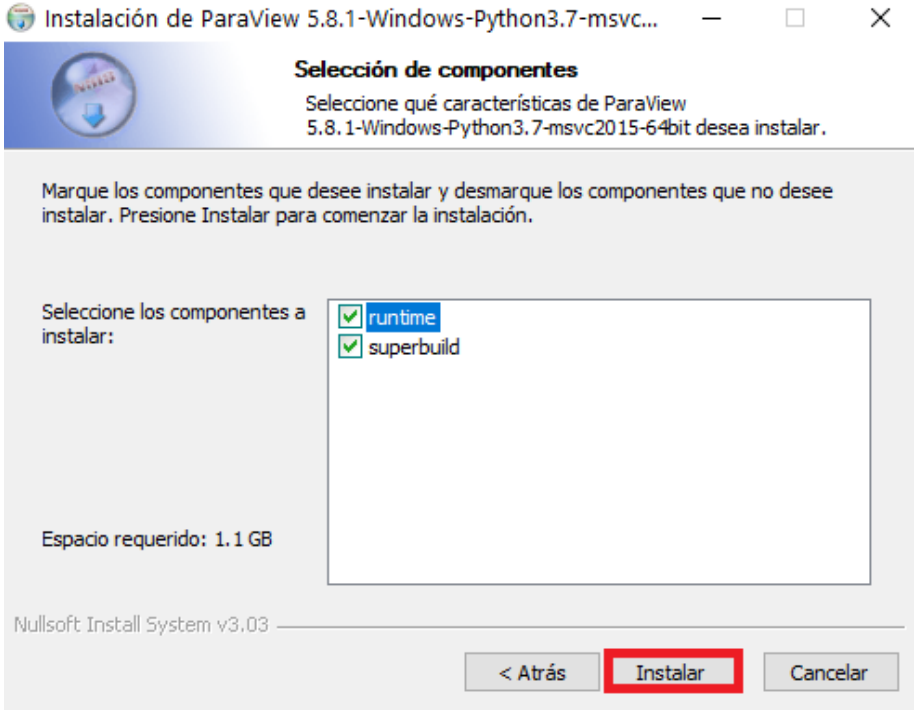


Figura 131. *Instalación de ParaView 5.8.1. Paso 5.5.*

Una vez que la instalación se ha completado, se pulsa en *Terminar* y aparecerá el acceso directo de ParaView 5.8.1. en el escritorio.

Anexo IV

Comandos básicos de Linux®

En este anexo se explican los comandos más básicos de Linux® mediante aplicación práctica en uno de los tutoriales propios de OpenFOAM®. La finalidad de este anexo es que el lector de este trabajo cuente con cierta base de manejo de Linux® para entender cómo funciona OpenFOAM®.

1. Comando cd (change directory).

cd es el comando que se usa para acceder directorios/carpetas, ya sea dentro de Windows o de Ubuntu®. Para ello se escribe *cd /<ruta-del-directorio>/*

Cómo acceder al escritorio de Windows 10.

La ruta para acceder al escritorio es */mnt/c/Users/<USER>/desktop*

Donde pone *<USER>* se debe escribir el nombre de usuario que se está utilizando en Windows.

```
cd /mnt/c/Users/<USER>/desktop
```

Sube un directorio, en este caso desde el directorio desktop hasta *<USER>*.

```
cd ..
```

Retrocede hasta el directorio justamente anterior al que estabas y muestra su ruta por pantalla.

```
cd -
```

Retrocede hasta el inicio de Ubuntu®, en este caso desde *<USER>* hasta el inicio.

```
cd
```

En azul se muestra el directorio al cual se está accediendo en cada momento:

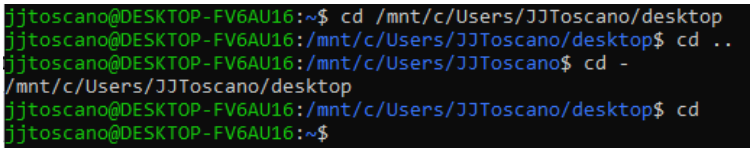


Figura 132. *Comando cd.*

Cómo acceder a la instalación de OpenFOAM 5.0 en Ubuntu® desde el inicio.

```
cd /opt/openfoam5
```

2. Comando ls (list). Es similar al comando dir

Cómo listar los archivos y carpetas del tutorial *cavity*.

Se accede al escritorio desde el inicio, tal como se ha visto anteriormente.

```
cd /mnt/c/Users/<USER>/desktop
```

Se accede a la carpeta *cavity* desde el escritorio.

```
cd cavity
```

Se listan los archivos y carpetas contenidos en *cavity*.

```
ls
```

Se listan los archivos contenidos en *cavity* y de sus directorios si los hubiese.

```
ls *
```

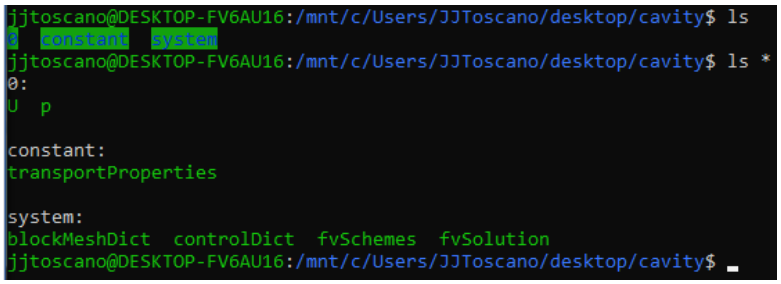


Figura 133. *Comando ls 1.*

3. Comando cp (copy).

Cómo copiar una carpeta de un caso tutorial desde Ubuntu® al escritorio de Windows.

Primero se debe acceder al escritorio.

```
cd /mnt/c/Users/<USER>/desktop
```

Se copia y pega en el directorio, al cual se acaba de acceder, el tutorial *cavity* desde la instalación de OpenFOAM 5.0 en Ubuntu®, cuya ruta es *$FOAM_TUTORIALS/incompressible/icoFoam/cavity/cavity* que es equivalente escribir la ruta cómo:

/opt/openfoam5/tutotials/incompressible/icoFoam/cavity/cavity

```
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity/cavity .
```

Nótese el espacio y punto del último comando para que la copia se realice correctamente y sin errores.

Se puede comprobar que se ha pegado correctamente viendo si aparece la carpeta en el escritorio de Windows.

4. Comando mv (move).

Cómo mover un directorio hacia otro directorio.

Se comprueba el directorio *cavity* del escritorio mediante *ls **.

```
jjtoscano@DESKTOP-FV6AU16:/mnt/c/Users/JJToscano/desktop/cavity$ ls *
0:
U  p

constant:
transportProperties

system:
blockMeshDict  controlDict  fvSchemes  fvSolution
```

Figura 134. *Comando ls 2.*

Se aprecia cómo se cuenta con 3 carpetas: *0*, *constant* y *system*. La tipografía en blanco significa que son directorios, mientras que la tipografía en verde revela que son archivos de texto.

Si se quiere mover un directorio hacia otro directorio que no es el directorio actual, basta con escribir: *mv <ruta-del-directorio-a-mover>/ <ruta-donde-se-quiere-mover>/*.

Nótese el espacio para separar argumentos.

Si desde *cavity* se quiere mover el directorio *0* hacia el directorio *constant*.

```
mv 0/ constant/
```

```
jjtoscano@DESKTOP-FV6AU16:/mnt/c/Users/JJToscano/desktop/cavity$ mv 0/ constant/
jjtoscano@DESKTOP-FV6AU16:/mnt/c/Users/JJToscano/desktop/cavity$ ls *
constant:
0  transportProperties

system:
blockMeshDict  controlDict  fvSchemes  fvSolution
```

Figura 135. *Comando ls 3.*

Si ahora desde *cavity* se quiere volver a traer la carpeta *0* pero con otro nombre, bastaría con escribir: *mv < ruta-del-directorio-a-mover>/ <nombre>*

Nótese el espacio para separar argumentos.

```
mv constant/0 1
```

```
jjtoscano@DESKTOP-FV6AU16:/mnt/c/Users/JJToscano/desktop/cavity$ mv constant/0 1
jjtoscano@DESKTOP-FV6AU16:/mnt/c/Users/JJToscano/desktop/cavity$ ls *
1:
U  p

constant:
transportProperties

system:
blockMeshDict  controlDict  fvSchemes  fvSolution
jjtoscano@DESKTOP-FV6AU16:/mnt/c/Users/JJToscano/desktop/cavity$
```

Figura 136. *Comandos ls 4.*

5. Comando gedit. Editor de texto.

Cómo editar el archivo de texto *U* mediante su ruta desde el directorio *cavity*.

Para poder usar este comando en Windows, se necesita tener previamente ejecutado y conectado el X Server (Ver pasos 7 y 8 del Anexo III.3):

```
gedit 0/U
```

Abre el editor de texto y se permite su edición. Se pulsa *Save* para finalizar.

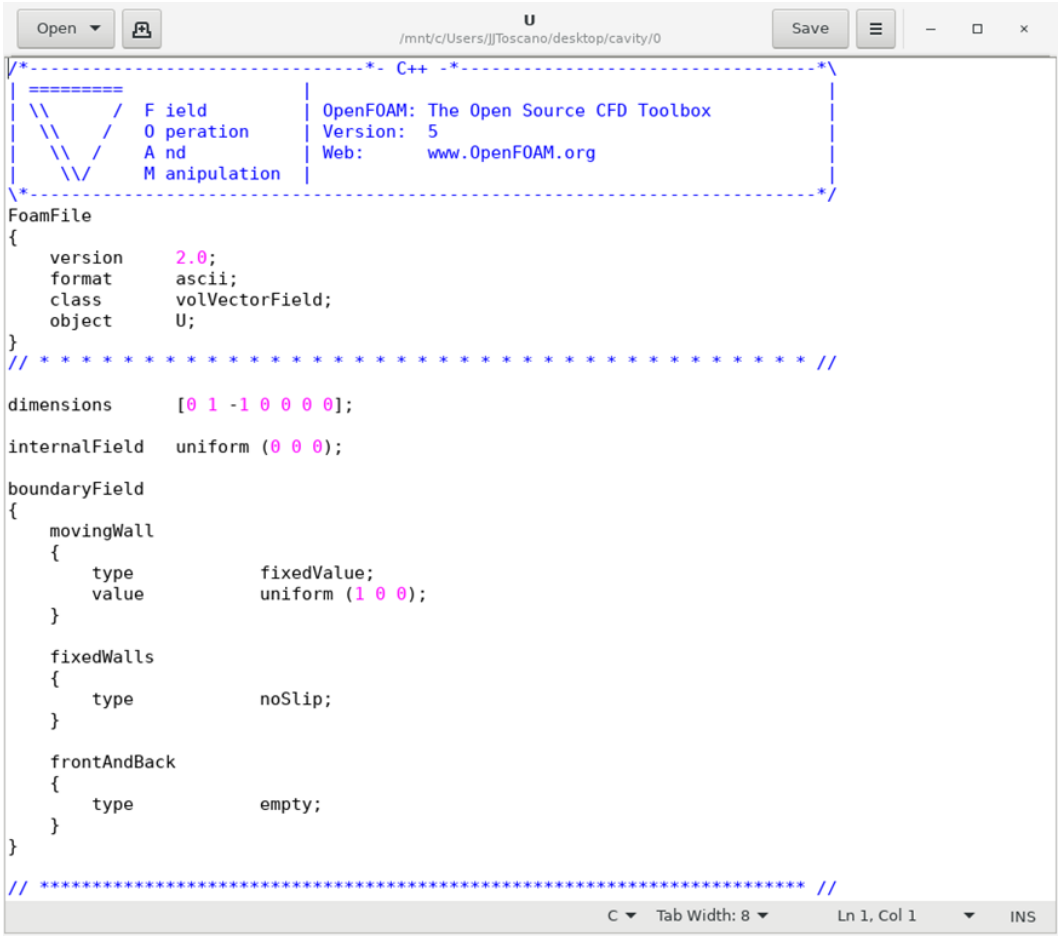


Figura 137. *Comadno gedit.*

6. Comando rm (Remove).

Cómo borrar el directorio *0* desde el directorio *cavity*.

```
rm -r 0
```

Borra el directorio y todos los archivos y directorios contenidos en él.

7. Comando pwd.

Saca por pantalla la ruta del directorio en el cual se encuentra actualmente.

```
pwd
```

8. Comando sudo (SuperUser DO).

Permite realizar tareas que requieren permisos de administrador.

9. Comando du -h.

Expresa en bytes, kilobytes y megabytes cuanto ocupa cada directorio contenido en el directorio actual.

10. Comando touch.

Crea archivos de texto vacíos con la extensión que se elija en el directorio actual.

11. Comando history.

Despliega una lista numéricamente ordenada del historial de comandos ejecutados por el usuario de Ubuntu® desde que se instaló.

12. Comando man.

Se escribe seguido de un comando de Linux® para desplegar las instrucciones manuales de ese comando en cuestión. Ejemplo: *man ls* (pulsa *q* para salir del manual).

13. Trucos extra.

- Reutilizar un comando: Ubuntu® permite escribir automáticamente y editar comandos previamente usados mediante su selección con las teclas de arriba y abajo.
- Comando *clear*: limpia el terminal de comandos usados.
- Tabular: si un archivo o directorio se encuentra dentro del que se está actualmente, mediante la tecla tabulador *TAB* se puede autocompletar su nombre. Por ejemplo si en el directorio actual existe una carpeta llamada *documento* bastará con escribir *cd doc* y pulsar el tabulador *TAB* para que se autocomplete *cd documento*. Lo mismo ocurre con comandos de OpenFOAM®, si se escribe tantas letras necesarias para que se reconozca y diferencie el comando.
- *Ctrl+C*: Cancela de forma segura la ejecución de un comando. Útil para abortar una simulación de OpenFOAM 5.0, cerrar ParaView 5.4.0 o *gedit*.
- *Ctrl+Z*: Fuerza la detención de un comando.
- *Ctrl+A*: Mueve el cursor de texto al comienzo de la línea.
- *Ctrl+E*: Mueve el cursor de texto al final de la línea.
- Ejecución paralela comandos: comando1;comando2 (Ejecuta los comandos 1 y 2 a la vez).
- Ejecución serie comandos: comando1&&comando2 (Ejecuta el comando 1 y después el 2).
- Crear archivos de texto de ejecuciones de comandos: blockMesh > bloque
- (crea el archivo de texto *bloque* a partir de la ejecución del *blockMesh*).

Anexo V

Guía básica de OpenFOAM® y ParaView®. Lid-driven cavity flow

En este anexo se describe en detalle el proceso de configuración, preprocesado, simulación y postprocesado mediante ParaView® para un caso tutorial de OpenFOAM® llamado *Lid-driven cavity flow*. La guía está basada en el punto 2.1 del documento de OpenFOAM® llamado *OpenFOAM v5 User Guide* (The OpenFOAM Foundation, 2016). Para seguir este anexo adecuadamente el lector debe tener instalado en un PC al menos la versión 5 de OpenFOAM® (ver Anexo III) y tener conocimientos mínimos de comandos básicos de Linux (ver Anexo IV), así como de simulación CFD (ver Capítulo 3). Además es recomendable leer la descripción general de OpenFOAM® (Anexo I) y de ParaView® (Anexo II).

Dentro del paquete de instalación de OpenFOAM 5.0 se encuentra el directorio *$FOAM_TUTORIALS* que contiene este y muchos más casos de ejemplo para mostrar la aplicación práctica de todos los solucionadores y utilidades. Los tutoriales están organizados en directorios de acuerdo con el tipo de fluido y en subdirectorios de acuerdo con el solucionador. Por ejemplo, todos los casos del solucionador *icoFoam* están organizados en *$FOAM_TUTORIALS/incompressible/icoFoam*, donde *incompressible* indica el tipo de fluido.

Para no perder el caso *cavity* del paquete de OpenFOAM 5.0 por modificación de archivos, la recomendación es siempre copiar el tutorial en cuestión desde su directorio y

pegarlo en una ruta fácilmente accesible en Windows, por ejemplo el escritorio. Para ello se escriben los siguientes comandos.

```
cd /mnt/c/Users/<USER>/desktop
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity/cavity .
cd cavity
```

El directorio final de *cavity* es un caso entre varios dentro del directorio de mismo nombre *cavity*. A partir de aquí se pueden explorar fácilmente directorios y visualizar o modificar los archivos del caso *cavity* desde el escritorio de Windows.

V.1 Descripción del caso. Lid-driven cavity flow

Este tutorial describe como preprocesar, simular y postprocesar un caso que involucra temperatura isotérmica y flujo incompresible en un dominio de dos dimensiones cuadrado. La geometría consiste en una cavidad, tal como se muestra en la Figura 138, en la cual todas las fronteras son muros. El muro superior se mueve en la dirección del eje x a una velocidad de 1 m/s mientras que los otros tres son estáticos. Este muro superior será el responsable de alterar el estado de reposo del fluido interno en la cavidad, sin que existan entradas ni salidas para el mismo. El flujo se asumirá como laminar y será resuelto en un mallado uniforme usando el solucionador *icoFoam*, adecuado para flujos laminares, isotérmicos e incompresibles.

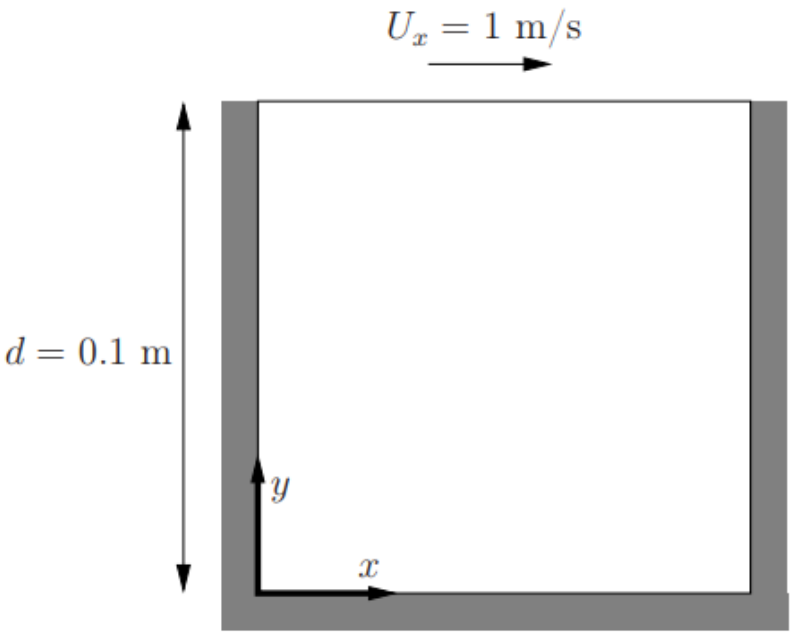


Figura 138. *Geometría del caso Lid-driven cavity Flow.*

V.2 Estructura general de un caso CFD por ejecutar

La peculiaridad más llamativa de OpenFOAM® es que no cuenta con interfaz gráfica de usuario (GUI). De tal forma, su manera de trabajo consiste en configurar (editar) archivos contenidos en directorios dentro de los casos, y ejecutar comandos de OpenFOAM®.

En OpenFOAM® todos los casos de tipo CFD cuentan con una estructura similar de directorios, la cual consiste en tres directorios llamados *constant*, *system* y *0*, con sus respectivos archivos. Esta estructura, en cuanto a nombres de directorios o archivos y su ubicación, está definida así para que cuando se ejecute algún comando de OpenFOAM® y requiera para ello de algún archivo en concreto, lo encuentre y no dé error. Es por ello que no se permite borrar, modificar nombres o reubicar ninguno de estos directorios o archivos.

Sin embargo, si ocurrirá que a la hora de ejecutar comandos durante el preprocesado, simulación y postprocesado, sí que se creen nuevos directorios de archivos que servirán para simular o postprocesar datos. Debido a ello, es recomendable tener una copia de seguridad del caso sin ejecutar por ejemplo en un formato zip, para así tener acceso al caso original y poder consultar estos archivos originales por si se quiere probar otra configuración del mismo.

En este apartado se describen los tres directorios y cada uno de sus archivos de un caso CFD por ejecutar.

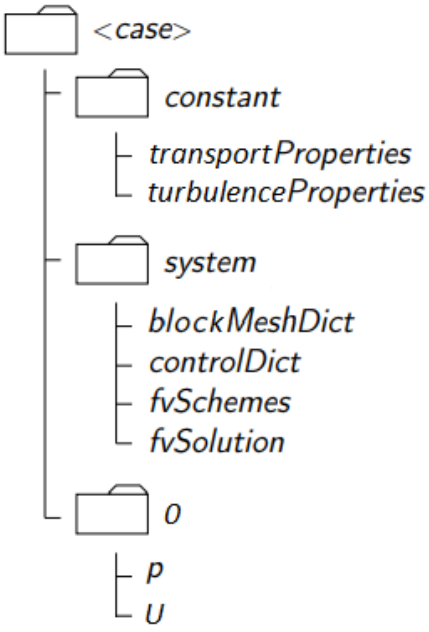


Figura 139. *Estructura básica de un caso CFD por ejecutar.*

V.2.1 constant

El directorio *constant* contiene las propiedades constantes del fluido y del modelo de turbulencia en los archivos *transportProperties* y *turbulenceProperties* respectivamente.

V.2.1.1 transportProperties

El archivo *transportProperties* expresa las propiedades constantes del fluido.

Tal como se ha comentado, en OpenFOAM® los casos se configuran editando archivos. El usuario debe escoger un comando para editar texto como *nano* o *gedit*, o bien editar de manera tradicional abriendo archivos mediante Windows. Editar casos es posible en OpenFOAM® debido a su diseño de formatos tipo diccionario con palabras clave, que transmiten una información fácilmente entendida por el usuario sin necesidad de tener conocimientos avanzados del lenguaje C++.

Por motivos didácticos, los archivos que se vean en este tutorial se abrirán y editarán mediante el comando *gedit*. Encontrándose en la ruta de *cavity*, se escribe la siguiente línea de comando, estando previamente ejecutado y conectado el X Server (Ver pasos 7 y 8 del Anexo III.3):

```
gedit constant/transportProperties
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  5                                     |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

nu              [0 2 -1 0 0 0 0] 0.01;

// ************************************************************************* //
```

Figura 140. *transportProperties de cavity.*

La estructura del encabezado de los archivos de OpenFOAM® es muy similar. Arriba del todo se encuentra un banner comentado (en azul) con el logotipo e iniciales de OpenFOAM® para referirse a que este es un archivo del mismo. Justo después se indica el subdiccionario *FoamFile* con palabras clave cuyo significado puede verse en la Tabla 6.

Tabla 6. *Palabras clave de FoamFile.*

Palabra clave	Descripción	Entrada
version	E/S del formato de versión	*2.0*
format	Formato de datos	*ascii / binary*
class	Construcción a partir de datos	*dictonary* u otro, por ejemplo *volVectorField*
location	Ruta del archivo desde el caso	(opcional)
object	Nombre del archivo	Por ejemplo *controlDict*

Tras el encabezado, aparecen las palabras claves y diccionarios realmente importantes del archivo. Para el archivo *cavity* de *tranportProperties* esa información importante se define únicamente mediante la palabra clave *nu*, que se refiere a la propiedad física viscosidad cinemática del fluido. Para expresarlo en una línea de código de C ++ se escribe *nu* de clase *word* seguido de espacio, seguido del ajuste de dimensión de clase *dimensionSet*, seguido de espacio y seguido del valor escalar de clase *scalar* con punto y coma al final para cerrar la instrucción. De tal manera, la dimensión de *nu* se expresa con el *dimensionSet*, que consiste en un vector de siete escalares entre corchetes que codifican la dimensión de la propiedad física, según la posición del componente del vector referido al código de la Tabla 7.

Tabla 7. *Vector dimensionSet para especificar las Unidades Básicas según SI.*

Posición del vector	Propiedad	Unidad del SI
1	Masa	kilogramo (kg)
2	Longitud	metro (m)
3	Tiempo	segundos (s)
4	Temperatura	Kelvin (K)
5	Cantidad de materia	mol (mol)
6	Intensidad	Amperio (A)
7	Intensidad luminosa	candela (cd)

La dimensión de cada propiedad física es la multiplicación de todas las unidades elevadas al número que aparezca en su posición. Siguiendo el código de la posición del vector para *nu*, se expresa kilogramo elevado a *0*, por metro elevado a *2*, por segundo elevado a menos *1*, por el resto de unidades elevadas a *0*. Esto da unas dimensiones de m^2/s, justamente las dimensiones de la viscosidad cinemática en el SI. La configuración de *nu* es 0.01 m^2/s.

V.2.1.2 turbulenceProperties

El archivo *turbelenceProperties* expresa el modelo de turbulencia que se va a seguir en la simulación. Dentro del caso *cavity*, se carece de modelo de turbulencia porque la descripción del caso indica que se encuentra en régimen laminar. Esto es así de acuerdo con el Número de Reynolds, tal como expresa la siguiente ecuación:

$$Re = \frac{D\,U}{\nu} = \frac{0{,}1\,m\,\cdot 1\,{}^{m}/_{s}}{0{,}01\,{}^{m^2}/_{s}} = 10 \qquad (32)$$

donde de d y $|U|$ son la longitud y la velocidad característica absoluta respectivamente, y ν es la viscosidad cinemática.

Sin embargo, se muestra el siguiente ejemplo de *turbulenceProperties* de otro caso. Por simplicidad y claridad, desde la Figura 141 se omitirá el banner superior y subdirectorio *FoamFile* de todos los archivos de OpenFOAM®.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType RAS;

RAS
{
    RASModel        kEpsilon;

    turbulence      on;

    printCoeffs     on;
}

// ************************************************************************* //
```

Figura 141. *Ejemplo de turbulenceProperties.*

En este archivo se indica la entrada que expresa el *simulationType* que para el ejemplo es *RAS* (Reynolds Average Simulation) y por otra parte la entrada que expresa el *RASModel* que para el ejemplo es el modelo de turbulencia *kEpsilon*, perteneciente a RAS.

V.2.2 system

El directorio *system* contiene la información relativa al cálculo de la solución mediante los archivos *controlDict*, *fvSchemes* y *fvSolution*. Además cuenta con el archivo que contiene la información necesaria generar la geometría mallada en la cual se va a desarrollar la simulación, llamado *blockMeshDict*.

V.2.2.1 blockMeshDict

Tal como se ha comentado, *blockMeshDict* es el archivo donde se define cómo será la geometría del mallado donde se desarrollará la simulación. Por defecto, OpenFOAM® siempre opera en las 3 dimensiones cartesianas, y todas las geometrías son generadas en 3 dimensiones. Entonces, para problemas en 2 dimensiones como es el caso *cavity*, se deberá generar igualmente una geometría en 3 dimensiones y especificar una condición de contorno especial para aquellas caras de la geometría normales a la dirección X, Y o Z donde no se requiere calcular la solución.

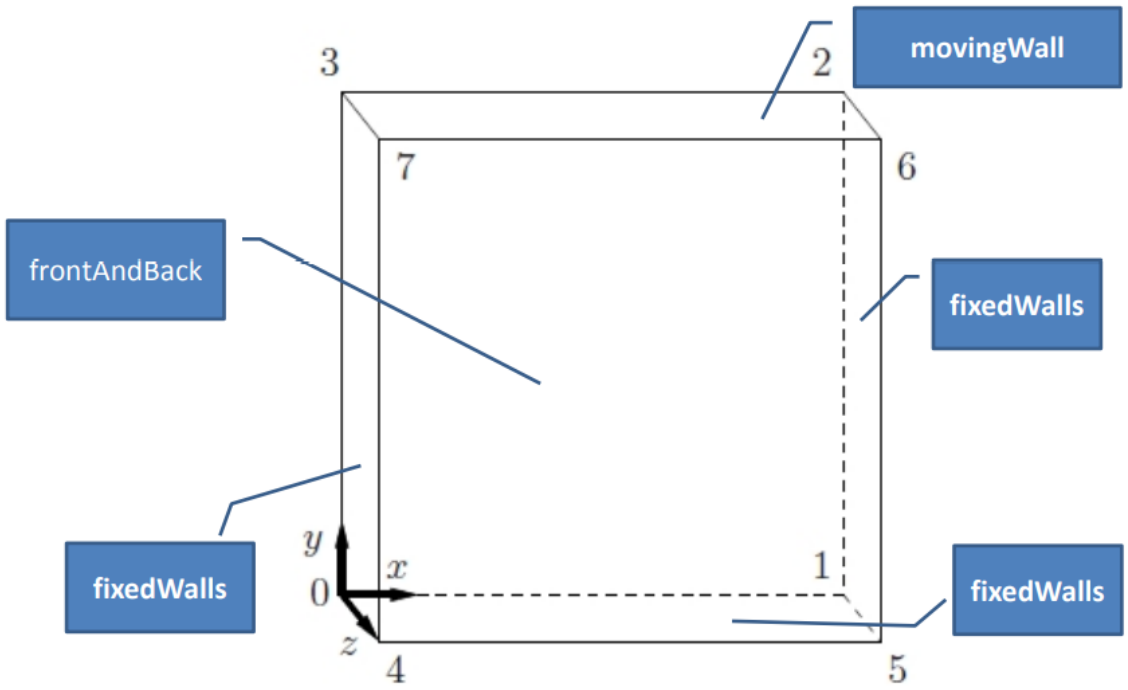


Figura 142. *Geometría 3D con el nombre de los parches del caso cavity.*

A continuación se explica cada uno de los subdiccionarios, funciones, palabras claves y demás especificaciones del archivo *blockMeshDict*.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 0.1;

vertices
(
    (0 0 0)
    (1 0 0)
    (1 1 0)
    (0 1 0)
    (0 0 0.1)
    (1 0 0.1)
    (1 1 0.1)
    (0 1 0.1)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
    movingWall
    {
        type wall;
        faces
        (
            (3 7 6 2)
        );
    }
    fixedWalls
    {
        type wall;
        faces
        (
            (0 4 7 3)
            (2 6 5 1)
            (1 5 4 0)
        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);

mergePatchPairs
(
);

// ************************************************************************* //
```

Figura 143. *blockMeshDict de cavity.*

Primero se define una entrada que convierte todas las dimensiones del mallado a metros, multiplicando estas dimensiones por un factor de escala de 1/10.

A continuación se definen los vértices del hexaedro del caso *cavity* mediante la función *vertices()*. Los vértices se definen mediante sus tres coordenadas del espacio entre paréntesis y separados por espacios. El primer vértice está asignado automáticamente con el nombre *0*, el siguiente con el *1*, y así hasta llegar al vértice *7*, definiéndose así el hexaedro en 8 vértices. Los ejes locales se forman como: eje X del vértice *0* al *1*, eje Y del vértice *1* al *2* y eje Z del vértice *0* al *4*. El orden de definición de los vértices deber ser, primero la cara perteneciente plano XY en un sentido antihorario y después la otra cara paralela al plano XY empezando por el vértice de similares coordenadas XY que el primer vértice y continuando la definición en el mismo sentido antihorario.

La función *blocks()* forma el hexaedro y define como será el mallado contenido en el hexaedro usando la sintaxis: *hex (vértices) (cantidad de celdas) simpleGrading (escala de la celda)*. *hex* se refiere a que la geometría será hexaédrica cuyos vértices son los definidos anteriormente separados por espacios y entre paréntesis. Inmediatamente después se define la cantidad de celdas en cada una de las dimensiones del espacio del bloque separados por espacios y entre paréntesis. De tal forma se crearán 20 celdas en los ejes X e Y y 1 una celda en el eje Z, por tanto un total de 400 celdas. El escalado se refiere a la graduación del mallado que se le quiere aplicar a las celdas, definido como cociente entre la longitud de la última celda dividido por la longitud de la primera según cada eje, como puede apreciarse en la Figura 144.

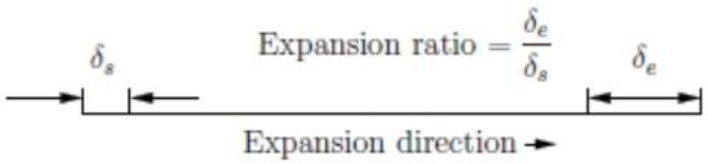


Figura 144. *Graduación del mallado a lo largo de una dirección.*

La función *edges()* se utiliza para curvar aristas. Como se requiere un volumen hexaédrico, esta entrada permanece vacía.

A continuación se definen los nombres de cada conjunto de caras del hexaedro, a lo cual se le llama parches, y cómo interactúa cada parche según su tipo en la simulación, mediante la función *boundary()*. De tal forma se definen, para el caso *cavity*, los parches de tres

conjuntos de caras mediante subdiccionarios: *movingWall* que será el muro móvil en la dirección del eje X a 1 m/s, *fixedWalls* que serán los tres muros fijos y *frontAndBack* que serán las 2 caras normales al eje *Z* donde no se resolverá el problema. Dentro de cada subdiccionario se debe definir el tipo (llamado tipo base) y las caras. Para los muros se establece con la entrada de tipo *wall*, mientras que *frontAndBack* será de tipo *empty*, para así no resolver el problema la dimensión *Z*. Además se debe especificar la función *faces*, que lista cada una de las caras mediante los nombres de sus vértices separados por espacios y entre paréntesis para separar las caras. El orden para escribir los vértices de una cara debe cumplir la regla de la mano derecha viendo la cara desde fuera de la geometría. Se define de esta forma para que el vector superficie tenga un sentido normal hacia fuera de la geometría, y así no haya confusión de sentidos a la hora de definir condiciones iniciales vectoriales como la velocidad *U*.

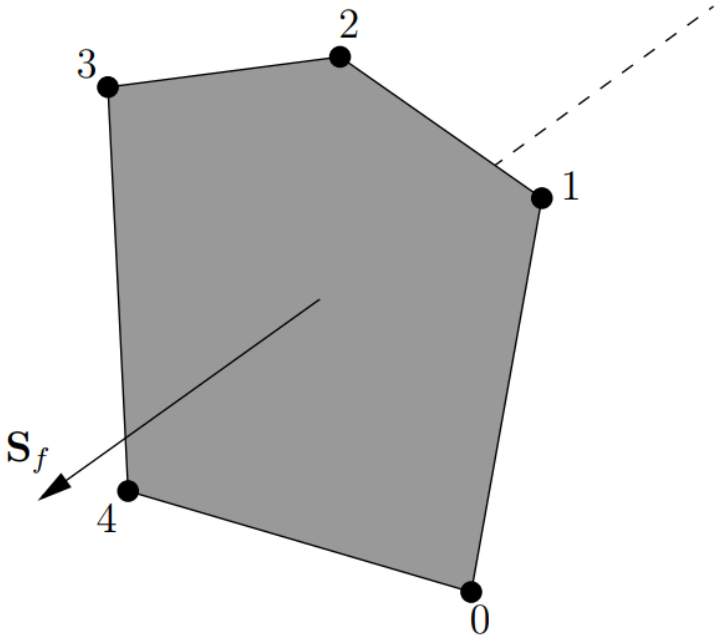


Figura 145. *Sentido del vector superficie S_f de una cara. Regla de la mano derecha.*

La última función llamada *mergePatchPairs()* sirve para conectar más de un bloque definido entre sí. Como se está definiendo un solo bloque, esta entrada permanece vacía.

V.2.2.2 controlDict

El archivo *controlDict* establece los datos de entrada relacionados con la elección del solucionador, el control del tiempo, además de la lectura y escritura de los directorios de tiempo que recogen los resultados.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     icoFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         0.5;

deltaT          0.005;

writeControl    timeStep;

writeInterval   20;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

// ************************************************************************* //
```

Figura 146. *controlDict de cavity.*

En este archivo se explicarán cada una de las palabras claves más importantes y sus entradas:

- *application*: establece el solucionador que se va a utilizar para resolver el caso. Como ya se ha comentado anteriormente, el solucionador de *cavity* es *icoFoam*.

- *startFrom*: indica a partir de donde se va a empezar a correr la simulación. Se suele indicar con la entrada de la siguiente palabra clave, que es *starTime*.

- *startTime*: establece el directorio desde el cual se toman las condiciones de partida. En todos los casos se establece el directorio *0*.

- *stopAt*: señala cuando se quiere terminar la simulación. Se suele indicar con el nombre de la siguiente palabra clave, que es *endTime*.

- *endTime*: fija el último directorio de tiempo que se podrá escribir de la simulación. Su escritura es en principio fija, por lo que no depende de *writeInterval*, pero puede pasar que la solución converja antes de ese tiempo, por lo que no llegará al *endTime* y escribirá el último directorio en el instante de tiempo de convergencia. Como regla general, la simulación debe durar 10 segundos para alcanzar el estado estacionario en flujo laminar. Si embargo eso sería así en casos donde exista entrada y/o salida de fluido, los cuales no existen en el caso *cavity*. Para este caso es suficiente con 0.5 segundos.

- *deltaT*: establece cantidad de instantes o salto de tiempo entre cada simulación. Este valor es muy importante porque asegura precisión temporal y estabilidad numérica de la simulación. Para ello se define el número Courant, que depende de *deltaT*:

$$Co = \frac{\delta t|U|}{\delta x} < 1 \tag{33}$$

  donde δt es el salto de tiempo, $|U|$ es el valor absoluto de la mayor velocidad que atraviesa las celdas a instante *0* y δx es la longitud de la celda en la dirección de la velocidad. La velocidad del fluido varia a lo largo del mallado y se debe asegurar que $Co < 1$ en todas las celdas. Para ello se elige δt basado en el peor caso posible: el

máximo *Co* correspondiendo al efecto combinado de la velocidad más alta del fluido con la longitud más pequeña de la celda. En el caso *cavity* la longitud de la celda en la dirección de la velocidad es fija a lo largo de cualquier dirección, y se puede calcular como se muestra en la siguiente fórmula.

$$\delta x = \frac{d}{n} = \frac{0{,}1\ m}{20} = 0{,}005\ m \tag{34}$$

En la formula, *d* es la longitud del mallado en la dirección de mayor velocidad y *n* es la cantidad de celdas a lo largo de esa longitud. Estos datos se pueden extraer de los configurados en el archivo *blockMeshDict*.

Por lo tanto, para alcanzar un *Co* menor o igual a 1 a lo largo de todo el mallado, el salto de tiempo *deltaT* se debe configurar con un valor menor o igual al valor calculado en la siguiente fórmula.

$$\delta t = \frac{Co\ \delta x}{|U|} = \frac{1\ \cdot\ 0{,}005\ m}{1\ {}^{m}/_{s}} = 0{,}005\ s \tag{35}$$

- *writeControl*: indica la opción para ajustar cada cuanto tiempo se quiere escribir los resultados simulados. En todos los casos se establece como *timeStep* para especificar que los datos son escritos según *writeInterval*.

- *writeInterval*: fija el número de instantes que deben simular hasta que se escriban los datos. Si por ejemplo se pretende escribir los resultados en los instantes *0.1*, *0.2*, *0.3*, *0.4* y *0.5* para un *deltaT* de 0,005 s entonces se quiere escribir los resultados cada 20 veces, por lo que se configura la palabra clave *writeInterval* con la entrada 20.

  Como se verá posteriormente, al ejecutar el solucionador, OpenFOAM® creará nuevos directorios en la ruta *cavity*, cuyos nombres serán cada instante de tiempo de escritura, por ejemplo 0.1, 0.2, … 0.5. Dentro de estos directorios se encuentra el cálculo de todas las variables definidas en *0* para cada uno de esos instantes de tiempo en cada celda del mallado.

V.2.2.3 fvSchemes

El archivo *fvSchemes* especifica la elección de discretización de volumen finito. El usuario es libre de ver este archivo pero no es necesario conocer en detalle sus subdiccionarios, palabras claves y entradas dentro de esta guía básica de OpenFOAM®. La configuración del archivo es correcta tal como está y no se debe modificar.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear orthogonal;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         orthogonal;
}


// ************************************************************************* //
```

Figura 147. *fvSchemes de cavity.*

V.2.2.4 fvSolution

El archivo *fvSolution* especifica la ecuación linear del solucionador con su criterio de convergencia, además de las tolerancias, y otros algoritmos de control. El usuario es libre de ver este archivo pero no es necesario conocer en detalle sus subdiccionarios, palabras claves y entradas dentro de esta guía básica de OpenFOAM®. La configuración del archivo es correcta tal como está y no se debe modificar.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0.05;
    }

    pFinal
    {
        $p;
        relTol          0;
    }

    U
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-05;
        relTol          0;
    }
}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}


// ************************************************************************* //
```

Figura 148. *fvSolution de cavity.*

V.2.3 0

El directorio de tiempo 0 contiene los archivos que definen las condiciones iniciales de todas las magnitudes físicas variables del fluido. Dentro del caso *cavity*, las únicas variables en el tiempo serán la presión *p* y la velocidad *U*. Como se mostró previamente, este directorio debe estar referenciado en *controlDict*, para tener datos iniciales para resolver las ecuaciones.

V.2.3.1 p

El archivo *p* del directorio *0* define la presión inicial a lo largo del mallado. Como ahora se verá, esta presión es realmente presión partida por la densidad del fluido.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    movingWall
    {
        type            zeroGradient;
    }

    fixedWalls
    {
        type            zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }
}

// ************************************************************************* //
```

Figura 149. *p de cavity.*

Todos los archivos de las magnitudes físicas iniciales siguen la misma estructura. Por un lado dos palabras clave: *dimensions* e *internalField*. Y por otro lado el subdiccionario *boundaryField*.

En *dimensions* se especifica las dimensiones de la magnitud, tal como se explicó para la propiedad constante del fluido *nu*, pero omitiendo el valor escalar de la presión, que se especificará en *internalField* y en *boundaryField*. Como puede observarse esta magnitud tiene unidades de m^2/s^2 que evidentemente no son unidades de presión, son unidades de presión

partida por densidad. Sin embargo, por convenio de OpenFOAM® a esta variable se le llama presión *p* aunque no lo sea, ya que la densidad del agua (fluido habitual en CFD) es 1 kg/m^3.

En *internaField* se indica el valor inicial de la magnitud en el interior del mallado que puede ser uniforme o no uniforme (valores iniciales). En *cavity*, se ha configurado para una presión manométrica uniforme igual a *0* por conveniencia, aunque su valor no tiene mucha importancia debido a que esta presión es cinemática para un fluido incompresible. El solucionador calculará la presión real en los instantes sucesivos.

El subdiccionario *boundaryField* incluye las condiciones iniciales de los límites del mallado, llamados parches (conjuntos de caras). Para que OpenFOAM® no de error al simular, es estrictamente necesario que en este subdccionario se incluyan tantos subdccionarios como parches se definió en el archivo *blockMeshDict*, y con el mismo nombre exacto de esos parches. Dentro de la función de cada parche se indica el tipo de parche, que puede ser tipo primitivo o tipo derivado, y define las interacciones físicas de la magnitud con los parches, también conocidos como condiciones de frontera.

El caso *cavity*, está compuesto únicamente por muros, separados en dos parches llamados: *fixedWalls* para los muros laterales y base de la cavidad, y *movingWall* para el muro superior en movimiento. Al ser muros, el tipo de condición frontera para *p* es *zeroGradient*, que significa que el gradiente normal de presión es 0. Por otra parte, para el parche *frontAndBack*, se le asigna el tipo base *empty*, ya que representan las caras frontales y traseras de un caso de 2 dimensiones. Se puede ver una clasificación de los tipos más comunes de tipos (valga la redundancia) en la Tabla 8.

Tabla 8. *Tipos de tipos más comunes.*

Tipo Base	Tipo Primitivo	Tipo Derivado
patch	fixedValue	inlet
wall	fixedGradient	outlet
symmetry	zeroGradient	inletOutlet
empty	mixed	
wedge	directionMixed	
cyclic	calculated	
processor		

También se puede ver en la Tabla 9 una descripción de cada tipo primitivo, con los datos que se deben especificar en cada caso.

Tabla 9. *Descripción de cada Tipo Primitivo.*

Tipo Primitivo	Descripción para la magnitud física Φ	Datos a especificar
fixedValue	Valor de Φ es el especificado	*value*
fixedGradient	Gradiente normal de Φ es el especificado	*gradient*
zeroGradient	Gradiente normal de Φ es cero	-
mixed	Mixto entre *fixedValue* y *fixedGradient*, dependiendo de *valueFraction*	*refValue* *refGradient* *valueFraction* *value*
directionMixed	*mixed* siendo normal al parche con una condición *fixedGradient* tangencial al parche	*refValue* *refGradient* *valueFraction* *value*
calculated	El valor de Φ en esa frontera depende de otras fronteras	-

V.2.3.2 U

El archivo de *U* del directorio *0* define la velocidad inicial a lo largo del mallado. Como puede observarse, la estructura de palabras claves y subdirectorios es similar a la del archivo *p*, con algunas diferencias que se comentarán.

En *dimensions* se ha configurado la dimensión de la velocidad a m/s.

En *internalField* se ha configurado la velocidad como uniforme mediante tres componentes. Mientras que la presión es una magnitud escalar, la velocidad es una magnitud vectorial, y por eso debe definirse en sus tres componentes separadas por espacios y entre paréntesis. En el interior del mallado, estas componentes valen 0, ya que se supone que el fluido está inicialmente en reposo.

En *boundaryField* se dividen todos los parches de igual manera que en *p*, diferenciándose en los tipos de cada parche que son propios de la magnitud física velocidad.

El parche movingWall es de tipo *fixedValue*, que quiere decir valor fijo y que requiere una especificación de valor, tal como se mostró en la Tabla 9. Este se ha configurado como uniforme a 1 m/s en el sentido positivo del eje *X*. El parche *fixedWalls* es de tipo *noSlip*, que es la condición de frontera que define la interacción física del fluido con los muros.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    movingWall
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }

    fixedWalls
    {
        type            noSlip;
    }

    frontAndBack
    {
        type            empty;
    }
}

// ********************************************************************************** //
```

Figura 150. *U de cavity.*

V.3 Configuración del caso

Tal como se ha comentado en OpenFOAM® los casos se configuran editando archivos y guardando los cambios. Esta configuración no debe ser el cambio u omisión de parte de la estructura de los archivos, sino de sus entradas. De lo contrario OpenFOAM® sacará mensajes de error a la hora de ejecutar los comandos por la terminal, señalando la línea donde se cometió el error y el por qué.

En este tutorial se dejarán los archivos tal y como vienen configurados en el paquete de instalación de OpenFOAM 5.0, por lo que no se va a realizar ninguna configuración distinta de los mismos. Posibles ejemplos de configuración del caso sería ajustar *U* a 2 m/s o ajustar *writeInterval* a 10 en *controlDict*.

V.4 Preprocesado

El preprocesado consiste en realizar la configuración del caso, crear la geometría a importar (si la hay) y ejecutar los comandos que generen el mallado en el cual se van a resolver las ecuaciones y definir los parches del mismo.

En OpenFOAM® la generación del mallado se omite si se utilizan programas externos para realizarlo. Si esto ocurre, habría que ejecutar el comando correspondiente de conversión para convertir esos datos generados por el programa externo en cuestión, a datos entendibles por OpenFOAM®. En este tutorial no se explica la utilización programas externos de generación de mallado, por lo tanto se hace estrictamente necesario.

En OpenFOAM® existen muchos comandos de preprocesado, pero solo hay uno que es completamente imprescindible, el comando *blockMesh*. Este comando lee la configuración del archivo *blockMeshDict* y genera el mallado y define los parches.

Es importante señalar que este y cualquier comando de OpenFOAM® que se quiera ejecutar sobre un caso, se deben ejecutar en la ruta exacta del directorio del caso, por lo que no valdría por ejemplo sobre el directorio del escritorio o cualquiera de los subdirectorios contenidos en el caso. Entonces, desde la ruta del directorio *cavity*, simplemente hay que escribir el comando:

```
blockMesh
```

Para generar el mallado y además extraer los resultados del mismo que aparecen en la terminal de Ubuntu® habría que haber escrito el siguiente comando:

```
blockMesh > mallado
```

Este comando crea un archivo de nombre *mallado* con los resultados de la generación del mallado que habrían aparecido en la terminal y lo coloca en el directorio donde se ha ejecutado el comando. Esto se puede hacer para cualquier comando de OpenFOAM®.

Como puede observarse en la Figura 151 del archivo *mallado*, el comando a finalizado su ejecución sin errores, y ha extraído la información básica del mismo, como las dimensiones, coordenadas, cantidad de celdas, parches etc.

```
/*---------------------------------------------------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  5.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
Build  : 5.x-68e8507efb72
Exec   : blockMesh
Date   : Mar 29 2021
Time   : 18:20:55
Host   : "DESKTOP-FV6AU16"
PID    : 286
I/O    : uncollated
Case   : /mnt/c/Users/JJToscano/desktop/cavity
nProcs : 1
sigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).
fileModificationChecking : Monitoring run-time modified files using
timeStampMaster (fileModificationSkew 10)
allowSystemOperations : Allowing user-supplied system call operations

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
Create time

Creating block mesh from
    "/mnt/c/Users/JJToscano/desktop/cavity/system/blockMeshDict"
Creating block edges
No non-planar block faces defined
Creating topology blocks
Creating topology patches

Creating block mesh topology

Check topology

        Basic statistics
                Number of internal faces : 0
                Number of boundary faces : 6
                Number of defined boundary faces : 6
                Number of undefined boundary faces : 0
        Checking patch -> block consistency

Creating block offsets
Creating merge list .

Creating polyMesh from blockMesh
Creating patches
Creating cells
Creating points with scale 0.1
    Block 0 cell size :
        i : 0.005 .. 0.005
        j : 0.005 .. 0.005
        k : 0.01 .. 0.01

Writing polyMesh
----------------
Mesh Information
----------------
  boundingBox: (0 0 0) (0.1 0.1 0.01)
  nPoints: 882
  nCells: 400
  nFaces: 1640
  nInternalFaces: 760
----------------
Patches
----------------
  patch 0 (start: 760 size: 20) name: movingWall
  patch 1 (start: 780 size: 60) name: fixedWalls
  patch 2 (start: 840 size: 800) name: frontAndBack

End
```

Figura 151. *Extracción de datos del blockMesh. Archivo mallado.*

La correcta ejecución del comando *blockMesh*, genera el directorio *polyMesh* dentro del directorio *0* de *cavity*. En este directorio se han creado los archivos *boundary*, *faces*, *neighbour*, *owner* y *points*. Estos archivos recogen toda la información calculada del mallado y pueden visualizarse en la interfaz gráfica de ParaView®.

V.5 Visualización del mallado

Visualizar el mallado previamente a ejecutar el solucionador es muy recomendable para asegurarse de que este se ha definido sin errores, ya que la simulación puede tardar mucho más que la generación del mallado. Para ello, encontrándose en la ruta de *cavity*, se escribe el comando *paraFoam*, estando previamente ejecutado y conectado el X Server (Ver pasos 7 y 8 del Anexo III.3):

```
paraFoam
```

La ejecución crea el archivo temporal *cavity.OpenFOAM* que abre automáticamente ParaView 5.4.0 mediante el X Server. Esta ejecución continua en marcha hasta que se escriba en la terminal *CTRL+C* o se cierre la ventana de Windows de ParaView 5.4.0, momento en el cual desaparecerá el archivo temporal *cavity.OpenFOAM*.

El problema del comando anterior es que permanece en ejecución mientras esté abierto ParaView 5.4.0, por lo que no se podrán ejecutar otros comandos mientras tanto. Una posible solución a este problema es ejecutar *paraFoam* en segundo plano, añadiendo espacio y & después de escribir *paraFoam*:

```
paraFoam &
```

De este modo, no se cierra ParaView 5.4.0 mediante *CTRL+C* en la terminal, por lo que la única forma de cerrarlo es mediante la ventana de Windows.

ParaView 5.4.0 es la herramienta de postprocesado proporcionada por OpenFOAM 5.0. Por otra parte, mediante ParaView 5.8.1 se puede acceder a la pestaña "*Help*" → "*Getting started with ParaView*" para descargar una guía muy básica de la interfaz gráfica de usuario de ParaView®.

En la Figura 152, se muestra el entorno gráfico de ParaView 5.4.0:

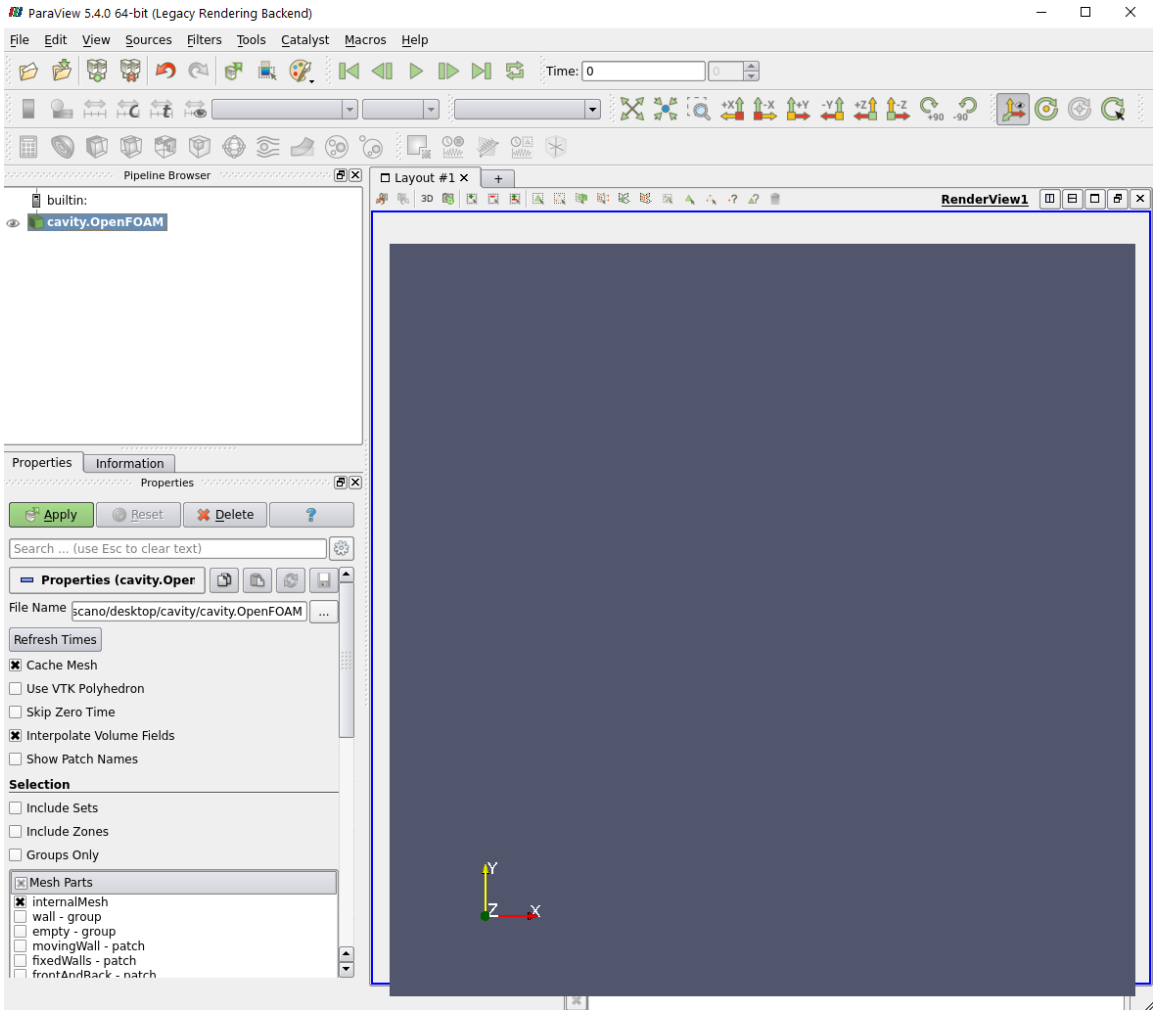


Figura 152. *Interfaz de ParaView 5.4.0.*

En la sección *"PipeLine Browser"* se encuentra el archivo *cavity.OpenFOAM* como módulo desactivado. Antes de activarlo, se debe seleccionar las partes del mallado que se quieren activar en *"Properties"* → *"Properties (cavity.OpenFOAM)"* → *"Mesh Parts"*. En este punto se marcan todas las partes del mallado y se pulsa *"Properties"* → *"Apply"* para activar el módulo *cavity.OpenFOAM*.

Muchas opciones de ParaView® requieren que se pulse *"Apply"* para aplicar la opción y así no saturar el programa con los cambios de visualización. Cada vez que se necesite pulsar *"Apply"*, su botón aparecerá como activable en un fondo verde, tal como se mostró en la Figura 152.

Se puede entonces minimizar el panel *"Properties (cavity.OpenFOAM)"* y acceder al panel *"Display"*, que controla la representación visual del módulo seleccionado. Dentro de *"Display"* se harán los siguientes ajustes:

1. en la sección *"Coloring"* seleccionar *"Solid Color"*.
2. en el menú "Representation" seleccionar "Wireframe".

Se puede rotar la vista de la geometría en cualquier momento mantenido pulsando el botón derecho del ratón sobre la ventana *"Layout #1"* mientras mueves el ratón.

Como *cavity* es un caso en 2 dimensiones, es muy recomendable cambiar la proyección del mallado de cónica a ortogonal. Para ello se puede proceder a minimizar el panel *"Display"* y desplegar el panel *"View"*. En este panel se debe marcar la opción *"Camara Parallel Projection"*. Puede apreciarse directamente como el mallado cambia de perspectiva cónica a isométrica. Además, para apreciar las dimensiones del mallado mediante ejes, se puede marcar la opción *"Axes Grid"* en el mismo panel de *"View"*.

Con todos los ajustes realizados la vista del mallado debe ser como se muestra en la Figura 153.

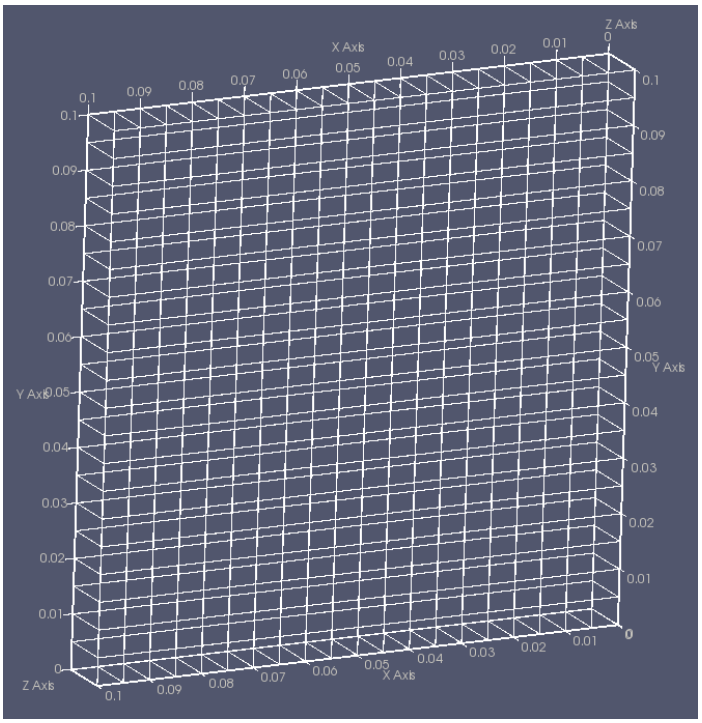


Figura 153. *Visualización del mallado en ParaView 5.4.0.*

Como puede apreciarse a primera vista, el mallado cumple los requisitos establecidos en el archivo *blockMeshDict*, sin ningún tipo de error en cuanto a sus dimensiones y cantidad de celdas.

También se puede comprobar que los parches fueron definidos correctamente dentro de *"Properties (cavity.OpenFOAM)"* → *"Mesh Parts"*, marcando y desmarcando parches y pulsando *"Apply"*. Por ejemplo, puede marcarse únicamente *"fixedWalls – pacth"* para visualizar únicamente los muros fijos de la geometría, tal como se muestra en la Figura 154.

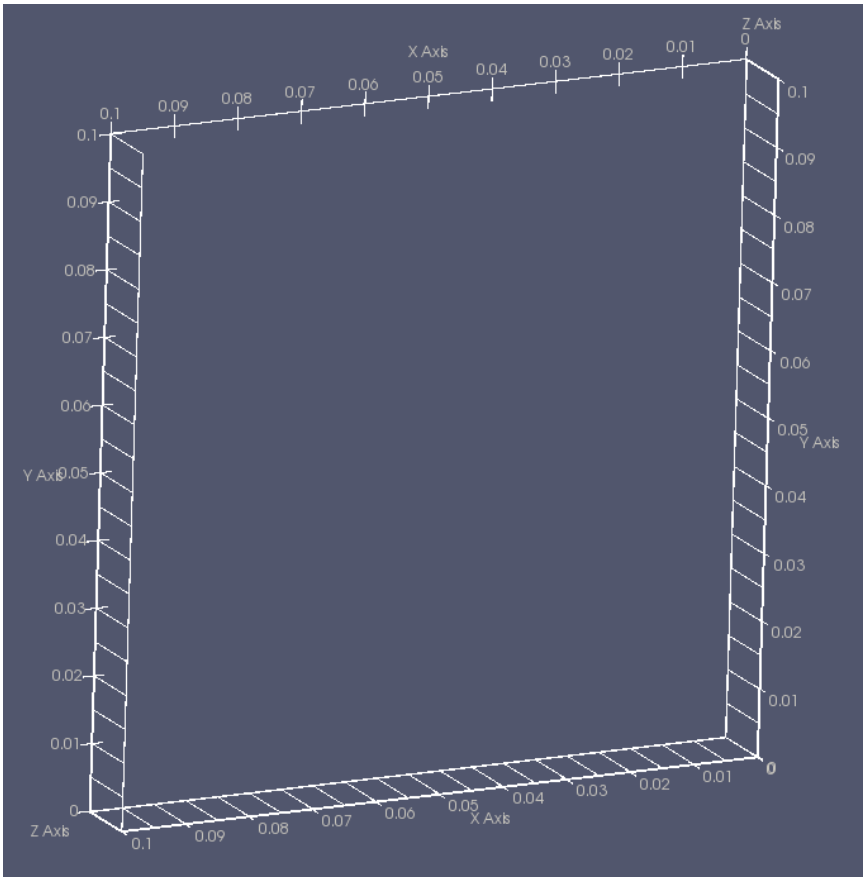


Figura 154. *Representación del parche fixedWalls en ParaView 5.4.0.*

Tras comprobar los parches, se puede verificar que también fueron definidos correctamente.

V.6 Simulación. Ejecución de icoFoam

Finalizado el preprocesado del caso y la visualización del mismo, se puede proceder a ejecutar el solucionador, que tal como se ha comentado, es un solucionador adecuado para flujos laminares de fluidos Newtonianos, isotérmicos e incompresibles llamado *icoFoam*.

El solucionador *icoFoam* resuelve las ecuaciones laminares incompresibles de Navier-Stokes utilizando el algoritmo *PISO*. El código es inherentemente transitorio y requiere una condición inicial y condiciones de contorno. También se puede utilizar para casos estacionarios añadiendo una instrucción que detenga la simulación cuando haya poca variación de los resultados entre saltos de tiempo.

El código *icoFoam* puede tener en cuenta la no ortogonalidad de la malla con iteraciones sucesivas de no ortogonalidad. El número de correcciones PISO y las correcciones de no ortogonalidad se controlan mediante la entrada del usuario. (OpenFOAM Wiki, 2009)

Para ejecutar la simulación hay que dirigirse previamente a la ruta de *cavity* y escribir:

```
icoFoam
```

De igual manera que se hizo para ejecutar el comando *blockMesh*, también se puede recopilar los resultados que se mostrarían por la terminal en un archivo llamado *simulación* mediante el siguiente comando:

```
icoFoam > simulacion
```

El proceso de simulación consiste primeramente en una lectura de datos del mallado recogido en el directorio *ployMesh*, de las magnitudes físicas iniciales definidas en el directorio *0*, además de la viscosidad cinemática del fluido *nu* y del modelo de turbulencia (si lo hubiese) del directorio *constant*. Por otra parte leerá los archivos *fvSchemes* y *fvSolution* para especificar la elección de discretización de volumen finito y la ecuación linear del solucionador.

Por último también se lee el archivo *controlDict*, que tal como se ha comentado, es el archivo más importante de la simulación porque define la cantidad de instantes que se calcularán, desde qué instante hasta cual, y cuáles de esos instantes se guardarán. Cualquier error de sintaxis o algún error concepto (no todos) lanzará un mensaje de error por la terminal que indique el archivo y número de la línea de código donde se cometió el error.

Como puede observarse en la Figura 155 de los resultados de la simulación, el comando a finalizado su ejecución sin errores. Esta Figura 155 solo expresa los dos primeros saltos de tiempo junto con el último para resumir.

```
/*---------------------------------------------------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  5.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
Build  : 5.x-68e8507efb72
Exec   : icoFoam
Date   : Mar 30 2021
Time   : 16:52:58
Host   : "DESKTOP-FV6AU16"
PID    : 309
I/O    : uncollated
Case   : /mnt/c/Users/JJToscano/desktop/cavity
nProcs : 1
sigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).
fileModificationChecking : Monitoring run-time modified files using timeStampMaster (fileModificationSkew 10)
allowSystemOperations : Allowing user-supplied system call operations

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
Create time

Create mesh for time = 0


PISO: Operating solver in PISO mode

Reading transportProperties

Reading field p

Reading field U

Reading/calculating face flux field phi


Starting time loop

Time = 0.005

Courant Number mean: 0 max: 0
smoothSolver:  Solving for Ux, Initial residual = 1, Final residual = 8.90511e-06, No Iterations 19
smoothSolver:  Solving for Uy, Initial residual = 0, Final residual = 0, No Iterations 0
DICPCG:  Solving for p, Initial residual = 1, Final residual = 0.0492854, No Iterations 12
time step continuity errors : sum local = 0.000466513, global = -1.79995e-19, cumulative = -1.79995e-19
DICPCG:  Solving for p, Initial residual = 0.590864, Final residual = 2.65225e-07, No Iterations 35
time step continuity errors : sum local = 2.74685e-09, global = -2.6445e-19, cumulative = -4.44444e-19
ExecutionTime = 0.01 s  ClockTime = 1 s

Time = 0.01

Courant Number mean: 0.0976825 max: 0.585607
smoothSolver:  Solving for Ux, Initial residual = 0.160686, Final residual = 6.83031e-06, No Iterations 19
smoothSolver:  Solving for Uy, Initial residual = 0.260828, Final residual = 9.65939e-06, No Iterations 18
DICPCG:  Solving for p, Initial residual = 0.428925, Final residual = 0.0103739, No Iterations 22
time step continuity errors : sum local = 0.000110788, global = 3.77194e-19, cumulative = -6.72498e-20
DICPCG:  Solving for p, Initial residual = 0.30209, Final residual = 5.26569e-07, No Iterations 33
time step continuity errors : sum local = 6.61987e-09, global = -2.74872e-19, cumulative = -3.42122e-19
ExecutionTime = 0.01 s  ClockTime = 1 s

                                                    ...

Time = 0.5

Courant Number mean: 0.222158 max: 0.852134
smoothSolver:  Solving for Ux, Initial residual = 2.3091e-07, Final residual = 2.3091e-07, No Iterations 0
smoothSolver:  Solving for Uy, Initial residual = 5.0684e-07, Final residual = 5.0684e-07, No Iterations 0
DICPCG:  Solving for p, Initial residual = 8.63844e-07, Final residual = 8.63844e-07, No Iterations 0
time step continuity errors : sum local = 8.8828e-09, global = 4.94571e-19, cumulative = 1.10417e-17
DICPCG:  Solving for p, Initial residual = 9.59103e-07, Final residual = 9.59103e-07, No Iterations 0
time step continuity errors : sum local = 9.66354e-09, global = 1.13175e-18, cumulative = 1.21735e-17
ExecutionTime = 0.16 s  ClockTime = 1 s

End
```

Figura 155. *Simulación del caso cavity. Archivo simulacion.*

La función del solucionador es resolver las ecuaciones Navier-Stokes para obtener las variables incógnitas, que son las definidas en la carpeta *0*, en este caso únicamente la velocidad *U* y la presión *p*. Este proceso se realiza en cada celda definida en *blockMeshDict*, y en cada salto de tiempo *deltaT* definido en *controlDict*. La solución de las variables incógnitas en cada salto de tiempo es a lo que se le llama instante. Para resolver cada instante se toman como condiciones iniciales los valores de variables incógnitas del instante inmediatamente anterior.

El caso *cavity* se ha resuelto en 100 instantes, uno cada 0,005 segundos. La cantidad de instantes se definió mediante entradas en *controlDict*, y se puede calcular mediante la siguiente ecuación:

$$nº\ instantes = \frac{endTime - \ startTime}{deltaT} = \frac{(0{,}5 - 0)\ s}{0{,}005\ \frac{s}{instante}} = 100\ instantes \qquad (36)$$

Dentro de *controlDict*, la palabra clave *writeInterval* define de esos 100 instantes cuales se escribirán, es decir, cuales guardarán los valores de las variables *p* y *U* de cada celda en directorios de tiempo. Según *controlDict*, *writeInterval* está configurado a *20*, lo cual quiere decir que guardará los instantes múltiplos de 20, es decir 20, 40, 60, 80 y 100. Esto en saltos de tiempo se corresponde con los instantes *0.1*, *0.2*, *0.3*, *0.4* y *0.5* segundos, por lo tanto estos son los nombres de los directorios de tiempo que se tienen que haber creado en la ruta *cavity*. El directorio *0.5* se hubiese creado igualmente independientemente de *writeInterval*, ya que se ha configurado *endTime* a *0.5* segundos. Sin embargo, si la solución hubiese convergido en algún instante anterior, sería ese instante el último directorio de tiempo en guardarse. El usuario podría hacerse la pregunta de, ¿por qué no se guardan todos los instantes configurando *writeInterval* a *1*? En un caso tan sencillo como *cavity*, donde existen pocas celdas y solo dos variables incógnitas, no tendría mucha importancia guardar todos los instantes. Sin embargo en casos más complejos de gran cantidad de celdas en las tres dimensiones, con superficies geométricas complejas en el interior del mallado y con aún más cantidad de variables, ajustar *writeInterval* a *1* puede suponer ralentizar muchísimo la simulación por tener que guardar directorios de tiempo de gran cantidad de datos, con el espacio que ello requiere.

En la Figura 155 se muestra que la simulación del caso *cavity* se ha resuelto en 0,16 segundos tal como pone en el *ExecutionTime* para $T = 0.5$, con un número Courant máximo de 0,852134, lo cual es inferior a 1, por lo que se asegura precisión temporal y estabilidad numérica.

V.7 Estructura general de un caso CFD simulado

Tras finalizar la simulación, la estructura básica general de un caso CFD debe ser tal y como se muestra en la Figura 156.

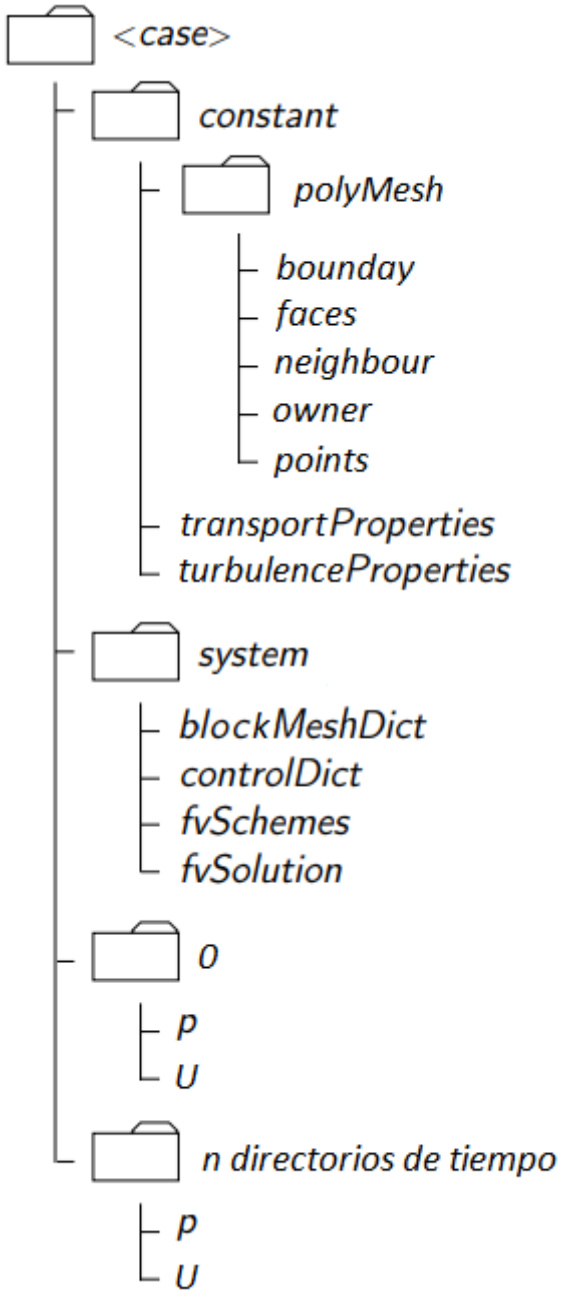


Figura 156. *Estructura básica de un caso CFD simulado.*

Por una parte se creó el directorio *polyMesh*, dentro del directorio *constant*, con todos los archivos generados mediante el comando *blockMesh* y que recogen la definición del mallado en cuanto a sus dimensiones, celdas, parches etc.

Por otra parte, durante la ejecución del comando *icoFoam*, se generaron tantos directorios de tiempo como fueron definidos en el archivo *controlDict*, que recogen cada uno de ellos el valor de cada celda para cada variable definida en el directorio *0*. Tal como se comentó, en el caso *cavity*, se debieron haber generado los directos de tiempo *0.1*, *0.2*, *0.3*, *0.4* y *0.5*, con los correspondientes archivos *p* y *U* que definen los valores de cada una de esas variables en cada celda en cada instante.

La Figura 157 muestra los primeros cinco y los últimos cinco valores de presión de las 400 celdas en una lista no uniforme del *internalField* de *p*, dentro del directorio *0.1*.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   nonuniform List<scalar>
400
(
-2.23126e-08
-0.00573238
-0.0127361
-0.018058
-0.0199809

                                    ...

0.442608
0.718597
1.22594
2.41113
4.84851
)
;

boundaryField
{
    movingWall
    {
        type            zeroGradient;
    }
    fixedWalls
    {
        type            zeroGradient;
    }
    frontAndBack
    {
        type            empty;
    }
}

// ************************************************************************* //
```

Figura 157. *p en 0.1 de cavity.*

Para cada variable, la definición de *dimensions* y *boundaryField* es exactamente la misma a como se definió para en el directorio *0*.

De igual manera, la Figura 158 muestra los primeros cinco y los últimos cinco valores de velocidad de las 400 celdas en una lista no uniforme del *internalField* de *U*, dentro del directorio *0.1*. A diferencia de la variable escalar presión, cada valor de velocidad se ha calculado por sus tres componentes, al ser una variable vectorial. En *boundaryField*, la velocidad del *movingWall* es 1 m/s constante durante toda la simulación.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   nonuniform List<vector>
400
(
(0.000249136 -0.000245641 0)
(0.000137297 0.000111667 0)
(-0.00116243 0.000558043 0)
(-0.00343171 0.000872759 0)
(-0.00628217 0.00103011 0)

                                        ...

(0.763481 -0.021755 0)
(0.699021 -0.039267 0)
(0.583912 -0.0726478 0)
(0.415819 -0.127862 0)
(0.308822 -0.14947 0)
)
;

boundaryField
{
    movingWall
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }
    fixedWalls
    {
        type            noSlip;
    }
    frontAndBack
    {
        type            empty;
    }
}

// ************************************************************************* //
```

Figura 158. *U en 0.1 de cavity.*

V.8 Postprocesado en ParaView®

En este punto se puede proceder a postprocesar los resultados de las variables calculadas en OpenFOAM® mediante ParaView®.

Entre las funcionalidades de ParaView que se verán este apartado, se encuentran representar magnitudes escalares o vectoriales, vectores mediante glifos, mallados, líneas de corrientes, contornos, cortes y largo etcétera, todo dentro de una escala de colores personalizable para cada variable. Además permite cálculos de otras variables a partir de las variables definidas en OpenFOAM® y la represión gráfica en 2 o 3 dimensiones de todas ellas.

Para abrir ParaView 5.4.0, se procede de igual manera a como se realizó para visualizar el mallado, que es mediante el comando *paraFoam*, estando previamente ejecutado y conectado el X Server (Ver pasos 7 y 8 del Anexo III.3):

```
paraFoam
```

Tal como se ha comentado en apartados anteriores, la ejecución crea el archivo temporal *cavity.OpenFOAM* que abre automáticamente ParaView 5.4.0 mediante el X Server. Esta ejecución continua en marcha hasta que se escriba en la terminal *CTRL+C* o se cierre la ventana de Windows de ParaView 5.4.0, momento en el cual desaparecerá el archivo temporal *cavity.OpenFOAM*.

A diferencia del archivo temporal *cavity.OpenFOAM* que se creó para visualizar el mallado y desapareció, este archivo *cavity.OpenFOAM* cuenta con los valores recogidos de todas las variables en los directorios de tiempo generados y guardados en la ejecución del solucionador.

El motivo principal de instalar alguna versión de ParaView® en Windows consiste precisamente en que este archivo *cavity.OpenFOAM* no sea un archivo temporal. Para ello, mientras este archivo está abierto mediante ParaView 5.4.0, se abre desde Windows el mismo archivo cuya ruta es *cavity*, mediante ParaView 5.8.1.

En este aparecerá una ventana que preguntará como se quieren abrir los datos. Simplemente seleccionar *OpenFOAMReader* y pulsar *OK*, tal como puede verse en la Figura 159.

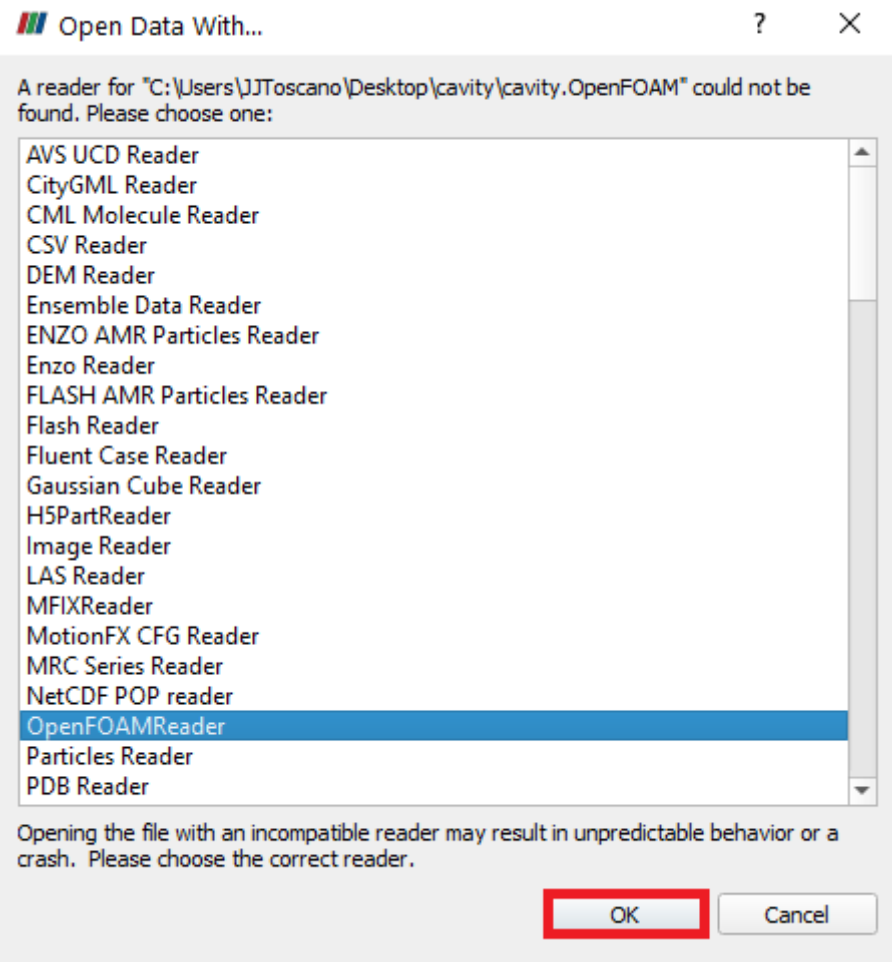


Figura 159. *Ventana para abrir cavity.OpenFOAM en ParaView 5.8.1.*

Una vez se pulse *OK*, se debe proceder primero a cerrar la ventana de ParaView 5.4.0. Si en este punto también se cierra ParaView 5.8.1 se puede observar que el archivo *cavity.OpenFOAM* no desaparece de la ruta *cavity*, por lo que se puede abrir el archivo en cualquier momento mediante ParaView 5.8.1 sin necesidad de crearlo temporalmente mediante la terminal.

Se vuelve abrir entonces el archivo *cavity.OpenFOAM* mediante ParaView 5.8.1 y en la pestaña *"Properties"* → *"Properties (cavity.OpenFOAM)"* debe estar marcado *"internalMesh"* dentro de *"Mesh Regions"*, y por otra parte *"U"* y *"p"* dentro de *"Cell Arrays"*. Una vez aseguradas estas selecciones, se puede pulsar *"Apply"* para aplicar el mallado y los valores de *p* y *U* en cada celda del mallado para cada instante de tiempo guardado en los directorios de tiempo.

V.8.1 Conceptos básicos de visualización de datos

Si los pasos previos se hicieron correctamente, se puede minimizar las pestañas *"Properties (cavity.OpenFOAM)"* y *"Display"*, y marcar *"View"* → "Camara Parallel Projection". Entonces se minimiza *"View"* y se maximiza *"Display"* para que la ventana de ParaView 5.8.1 se corresponda con la Figura 160.

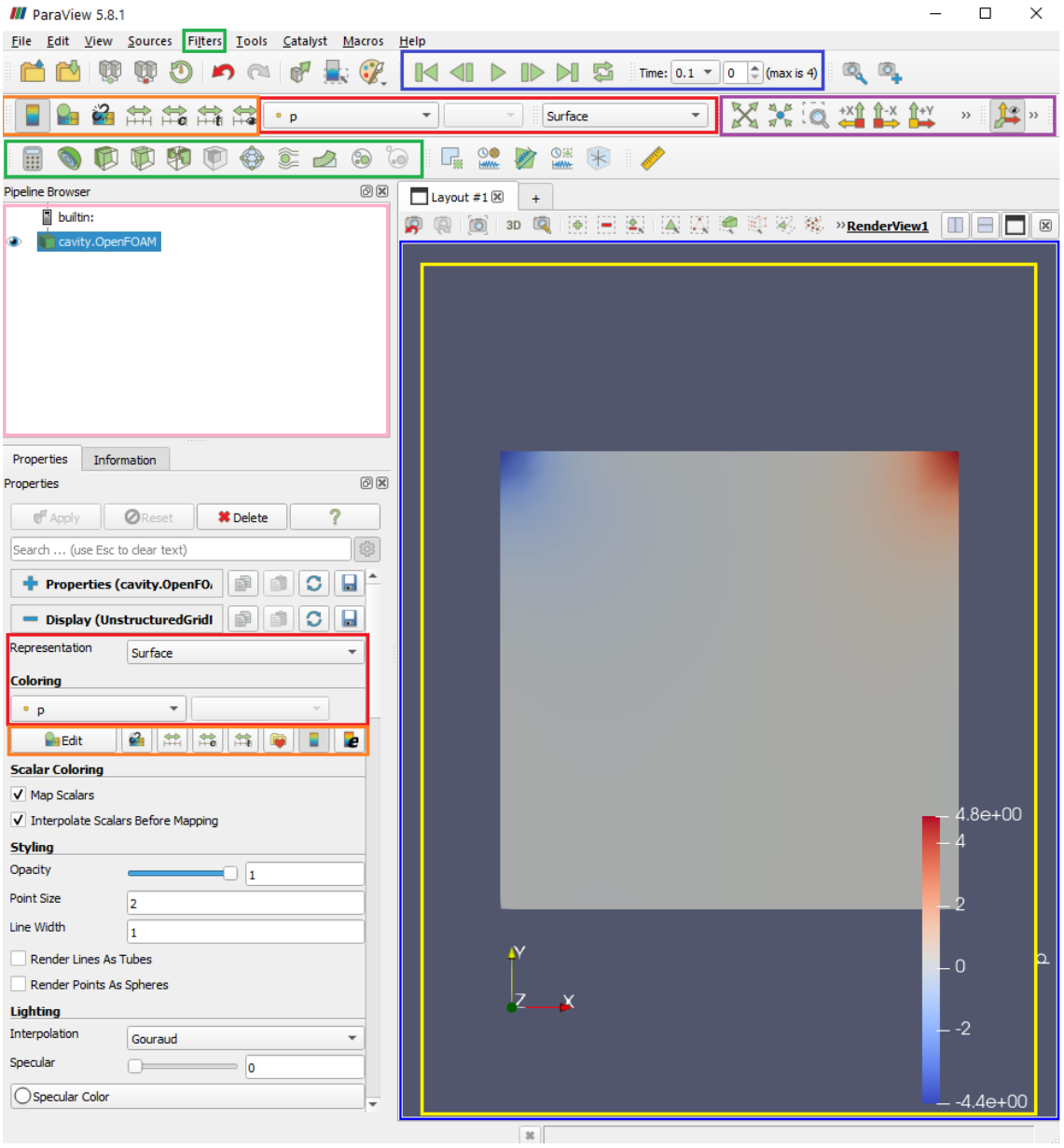


Figura 160. *Interfaz de visualización de datos en ParaView®.*

Entre las opciones más básicas y típicas de visualización de datos en ParaView®, están las señaladas en los cuadros de colores de la Figura 160, y se procederá al explicarlos uno a uno en este apartado.

El cuadro amarillo se refiere al *"Layout #1"* donde se representará la visualización de datos. Por defecto se representa cualquier variable con una escala de colores *Rojo-Blanco-Azul* llamado *"Cool to Warm"* que va desde 4.8 a - 4.4 m^2/s^2. Además de poder pinchar sobre el *"Layout #1"* para rotar la geometría, se puede arrastrar la escala para colocarla donde más convenga y en horizontal o vertical. En la geometría, se puede apreciar las mayores presiones en la esquina superior derecha y menores presiones en la esquina superior izquierda. Para cambiar el color del fondo del *"Layout #1"* se acceder a la pestaña *"Properties"* → *"View"* → *"Background"* → *"Color"* y seleccionar *blanco*. El icono de guardar (a la derecha de *"View"*) se pulsa para guardar esta configuración de *"View"* como por defecto.

El cuadro rojo permite elegir la variable que se quiere representar en el *"Layout #1"* y estilo de visualización de los datos. La selección por defecto de *p* está puesta en modo interpolación (icono del punto), con lo cual los colores degradados de la geometría son interpolaciones entre las celdas para dar una apariencia de continuidad. La lista desplegable *"Coloring"* permite en cualquier momento cambiar a la variable *U*, o incluso seleccionar *p* o *U* en modo celda (icono de celda), que representará el valor real de cada celda con un único color sin degradado. Si se selecciona *U*, la lista desplegable derecha permite elegir entre *"Magnitude"* que representa la suma vectorial de las componentes de *U*, o bien *"X"*, *"Y" o "Z"* que permite seleccionar una componente en concreto de *U* a representar. La Figura 161 representa la componente *Y* de *U* en modo celda:

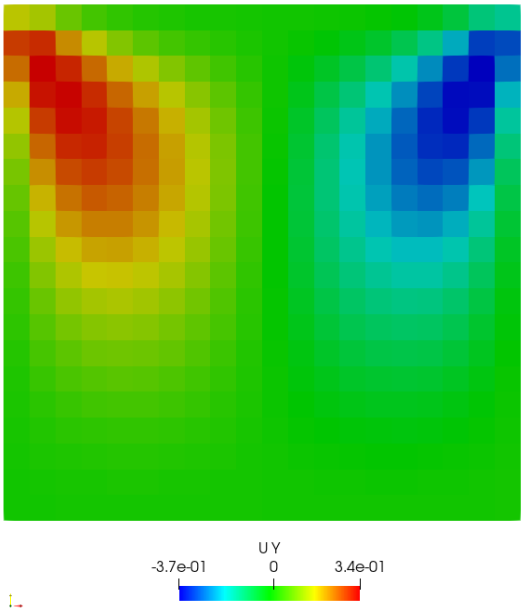


Figura 161. *UY en formato celda de cavity.*

Por otra parte, la lista desplegable *"Representation"* permite elegir el estilo de representación. Los típicamente usados son: *"Surface"*, *"Wireframe"* y *"Surface With Edges"*. *"Surface"* es el más típico, ya que permite visualizar la superficie de la geometría en colores definidos por la variable sin mallado. *"Wireframe"* permite visualizar el mallado con los colores de la variable. Por otra parte está *"Surface With Edges"*, que es una combinación de *"Surface"* con *"Wireframe"*.

El cuadro naranja se utiliza para redimensionar o para personalizar la apariencia la escala y la geometría según la escala. Refiriéndose al cuadro naranja de la pestaña *"Display"*, el tercer y cuarto botón por la izquierda, permiten redimensionar y definir los límites de la escala respectivamente.

El primer botón por la izquierda Edit abre la ventana *"Color Map Editor"*. En esta ventana se encuentran todas las opciones personalización de la apariencia la escala y la geometría según la escala. Dentro, el quinto botón de la línea de botones vertical , llamado *"Choose preset"*, permite elegir la escala de colores. En casos CFD como *cavity* es muy recomendable elegir escalas de colores de alto contraste como las de tipo arcoíris. En la ventana *"Choose preset"* se escribe *"rain"* en el buscador, se selecciona *"Blue to Red Rainbow"* y se pulsa *"Apply"*. Por otra parte, el botón superior derecho de *"Color Map Editor"* llamado *"Edit color legend properties"*, permite personalizar la leyenda en cuanto a nombres, tipografías, colores, etc. En este se debe cambiar el color de *"Title Font Properties"* y *"Text/Annotation Font Properties"* de blanco a negro. El icono de guardar (a la izquierda y abajo en la ventana *"Edit color legend properties"*) se pulsa para guardar esta configuración de la leyenda como por defecto. Una vez se han definido las configuraciones de visualización en *"Color Map Editor"* lo mejor es guardar estos ajustes para que se apliquen en todas las variables, y para que cada vez que se abra ParaView 5.8.1 se carguen directamente y no se tengan que volver a seleccionar una y otra vez. Para ello simplemente abría que pulsar el tercer botón de abajo en la misma ventana *"Color Map Editor"* que guarda los ajustes realizados como por defecto. Los botones pulsados en *"Color Map Editor"* se pueden ver en su ventana en la Figura 162.

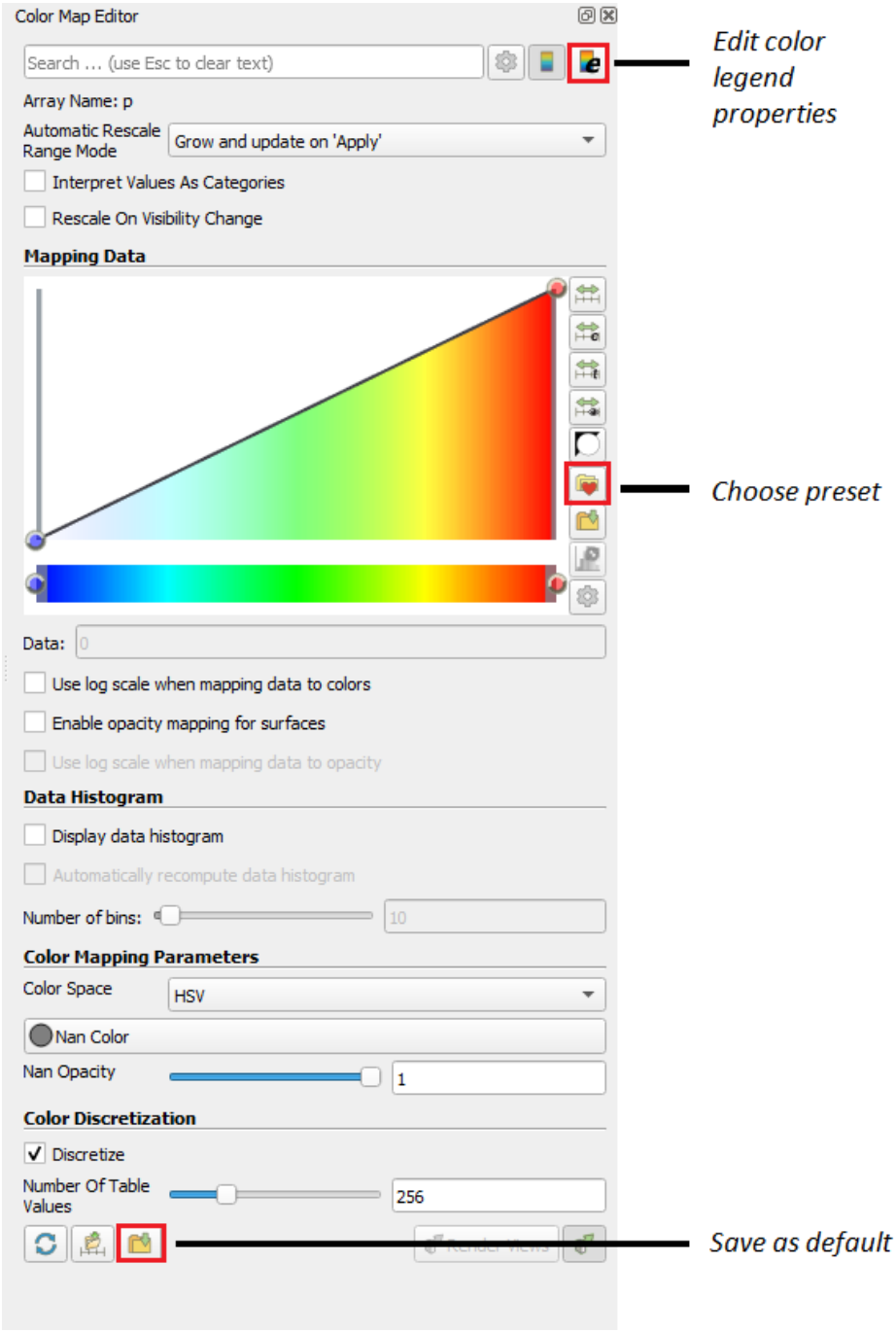


Figura 162. *Configuración en Color Map Editor.*

Cambiando de cuadro, el cuadro azul permite seleccionar un instante concreto o reproducir todos instantes mediante iconos de botones típicos de vídeo .

El cuadro morado permite elegir la vista o proyección que más convenga en cada caso. Para un caso en 2 dimensiones como *cavity*, con los ejes definidos tal como están, la vista más conveniente es la *-Z* . Entre otras opciones, también se permite rotar 90 grados las vistas en sentido horario o antihorario .

El cuadro rosa se refiere al llamado *Pipeline Browser*. En este se representan todos los módulos en formato jerárquico. Los módulos son selecciones de datos, que pueden ser archivos de cualquier formato compatible que se han abierto en ParaView® o filtros aplicados a esos archivos o a otros filtros. Para poder activar o desactivar la visualización de un módulo, simplemente hay que pulsar en el icono del ojo a la izquierda de cada módulo .

El cuadro verde representa los iconos de los filtros más comunes. Los filtros son aplicaciones ParaView® que procesan los datos de los módulos para generar, extraer o derivar características de ellos. Para poder aplicar un filtro a un módulo, primero se debe seleccionar este módulo dentro del *Pipeline Browser*. Se puede apreciar en la Figura 160 como el módulo *cavity.OpenFOAM* está seleccionado porque se encuentra en un fondo azul. Si en cambio no estuviese seleccionando ningún módulo (fondo de todos los módulos en celeste), los botones de filtros aparecerán como desactivados. Algunos de los filtros más utilizados para representar datos CFD se mostrarán en los siguientes apartados.

La Figura 163 muestra cómo debe quedar el *"Layout #1"* si los pasos previos se realizaron correctamente.

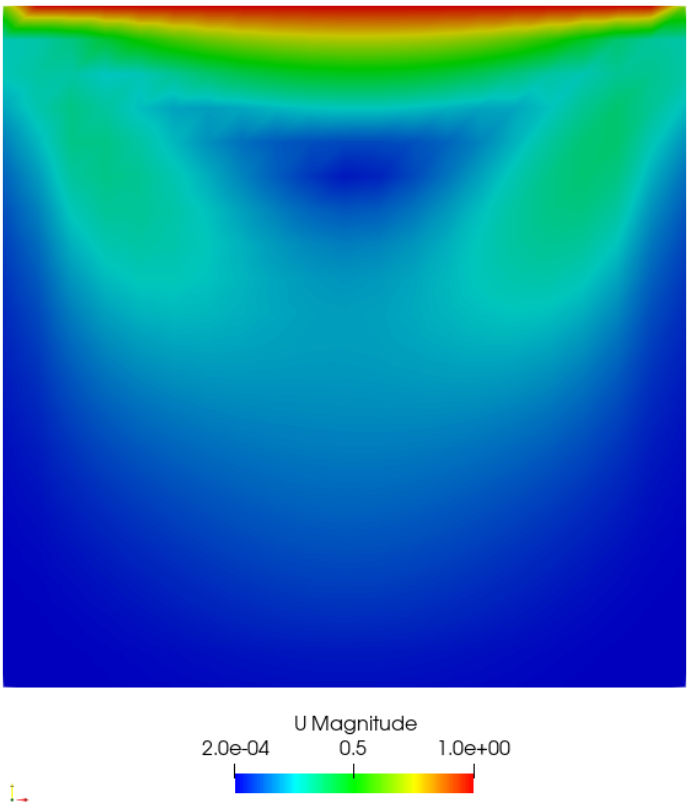


Figura 163. *"Layout #1" sin filtros.*

V.8.2 Slice

El filtro *Slice* se utiliza para obtener planos a partir de geometrías en 3 dimensiones conservándose los valores de las variables en la intersección del plano con la geometría. Para ello, con el módulo *cavity.OpenFOAM* previamente seleccionado, se pincha en el icono dentro de la barra de filtros comunes, o bien se selecciona mediante *"Filters"* → *"Alphabetical"* → *"Slice"*. En *"Alphabetical"* se recogen todos los filtros de ParaView® en orden alfabético, que además muestra cuales están o no están disponibles según el módulo seleccionado (iconos activados o desactivados).

Una vez seleccionado *"Slice"*, en *"Properties (Slice1)"* → *"Plane Parameters"* → *"Origin"* se sitúa el origen en (*0.05*, *0.05*, *0.005*), se selecciona *"Z Normal"*, se desmarca *"Show Plane"* para que no se desplace el plano por error si se rota la geometría, y se pulsa *"Apply"*. Como se puede apreciar rotando *"Layout #1"*, se ha realizado una rebanada (*slice*) a partir del volumen geométrico mediante un plano que pasa por el centro de la geometría y es normal al eje *Z*.

Se puede comprobar que en el *Pipeline Browser* debe haber aparecido un módulo justo debajo del módulo *cavity.OpenFOAM* llamado *Slice1* con la visualización activada, mientras que el módulo *cavity.OpenFOAM* tiene ahora la visualización desactivada. La visualización de estos módulos se puede combinar o cambiar en cualquier momento pulsando el icono del ojo, sin que esto afecte a sus datos, que siguen estando ahí aunque no se vean. En la Figura 164 se combina *Slice* con estilo *"Surface"* y *cavity.OpenFOAM* con estilo *"Wireframe"*.

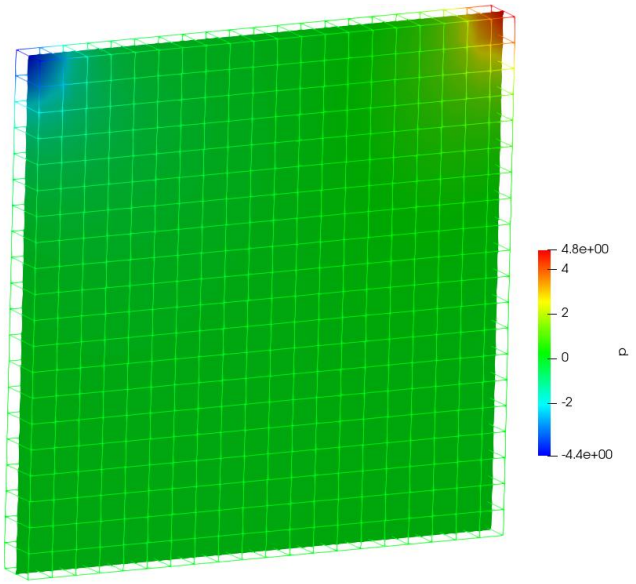


Figura 164. *Combinación de Slice con cavity.OpenFOAM*

V.8.3 Contour

El filtro *Contour* genera contornos de isovalores de una magnitud seleccionada. Se pueden definir tantos isovalores como se pretenda dentro de la escala de valores de la magnitud seleccionada y de 2 o 3 dimensiones dependiendo del módulo al que se le aplique el filtro.

Para ello, con el módulo *Slice* únicamente activado y seleccionado, se pincha en el icono dentro de la barra de filtros comunes, o bien se selecciona mediante *"Filters"* → *"Alphabetical"* → *"Contour"*. Entonces en *"Properties (Contour1)"* → *"Isosurfaces"* se pincha en el botón para eliminar la selección de contorno por defecto y se pincha en el botón para seleccionar los contornos.

En la ventana que aparece, se muestra el rango de valores de presión en que se mostrarán los contornos de tipo lineal. Se configura *Number of Samples* como *50* y se pulsa en *"Generate"*. Entonces se pulsa *"Apply"* para generar el contorno. En este caso, no se permiten hacer contornos de velocidad, pero si se permite combinar variables haciendo contornos de presión con la escala de colores de velocidad mediante *"View"* → *"Coloring"* → *"U"*, tal como se muestra en la Figura 165.

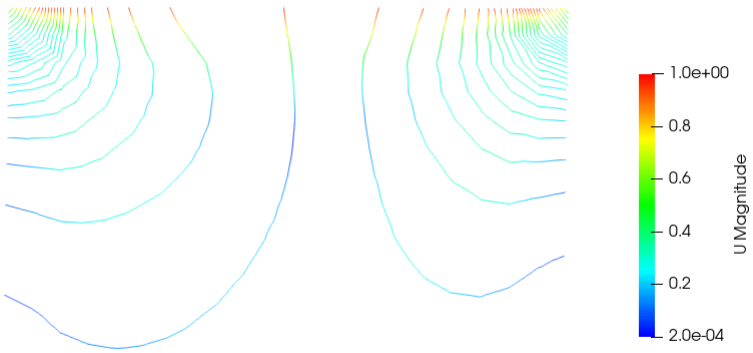


Figura 165. *Contorno de p con escala de colores de U.*

En ocasiones, combinar filtros según la escala de valores de una variable, con la escala de colores de otra variable es una forma muy aclaratoria de representar la relación entre ambas variables.

V.8.4 Gráfica de vectores

La gráfica de vectores permite visualizar una o más variables mediante flechas de colores que indican el sentido según su orientación y la intensidad según su escala de color y/o escala de tamaño.

Para ello, con el módulo *Slice1* únicamente activado y seleccionado, se selecciona mediante *"Filters"* → *"Alphabetical"* → *"Cell Centers"* y se pulsa *"Apply"*. Entonces al filtro *Cell Centers* se le aplica el filtro *Glifo*, pinchando en el icono dentro de la barra de filtros comunes, o bien se selecciona mediante *"Filters"* → *"Alphabetical"* → *"Glyph"* y se pulsa *"Apply"*.

En *"Properties (Glyph1)"* se configura la escala en *"Scale"* cambiando *"Scale Array"* como *"No scale array"* para que no aplique escalas según la variable, y por otra parte, *"Scale Factor"* en *0.004*. Se pulsa *"Apply"* y el resultado puede verse en la Figura 166.

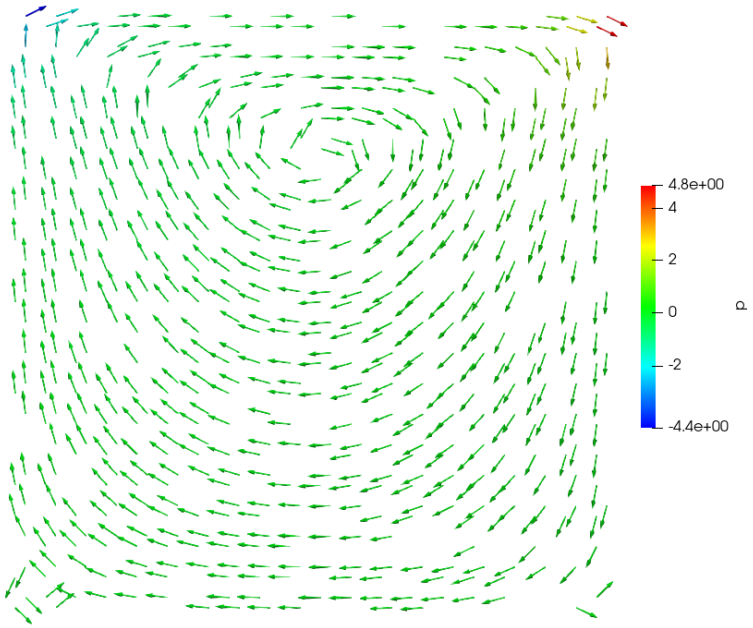


Figura 166. *Gráfico de vectores con sentido de U y color de p.*

La Figura 166 muestra el vórtice formado en la cavidad por el movimiento del muro superior representado mediante el sentido de los vectores según *U*, con una escala de colores según *p* y sin escala de tamaño para los vectores. Además se muestra los pequeños vórtices que se forman en las esquinas inferiores.

V.8.5 StreamTracer

El filtro *StreamTracer* dibuja las líneas de corriente que se cortan con una recta definida, o que atraviesan una esfera definida.

Para ello, con el módulo *Slice1* únicamente activado y seleccionado, se pincha en el icono dentro de la barra de filtros comunes, o bien se o se selecciona mediante *"Filters"* → *"Alphabetical"* → *"StreamTracer"* y se pulsa *"Apply"*. En la pestaña *"Properties (StreamTracer1)"* → *"Line Parameters"* se define la recta que corta las líneas de corriente mediante 2 puntos. Esta recta debe atravesar todo el vórtice para que se dibujen todas las líneas de corriente. Se ajusta *"Resolution"* a *25*, *"Point1"* a *(0.05,0,0.005)* y *"Point2"* a *(0.05,0.1,0.005)*, donde se recomienda previamente marcar *"Axes Grid"* en *"View"* para situarse en las coordenadas que se están estableciendo. Se pulsa *"Apply"* y se selecciona *"View* → *"Coloring"* → *"U"*. El resultado debe ser semejante a la Figura 167 con *"Axes Grid"* desmarcado.

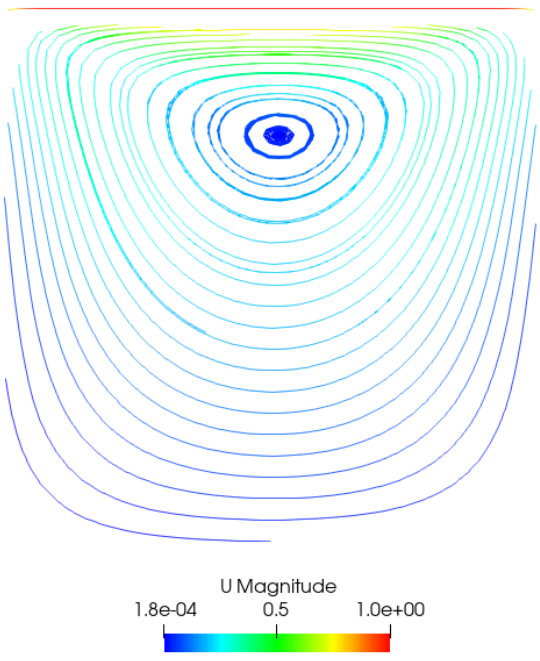


Figura 167. *StreamTracer de U.*

La Figura 167 representa la trayectoria de *25* de las líneas de corriente que forman el vórtice de *cavity* con una escala de color según *U*.

Es habitual utilizar el filtro *Tube* sobre las líneas de corriente para aumentar su grosor. Para ello, con el módulo *StreamTracer1* seleccionado, se selecciona mediante *"Filters"* →

"Alphabetical" → *"Tube"* y se pulsa *"Apply"*. En la pestaña *"Properties (Tube1)"* se ajusta *"Radius"* a *0.0004* y se pulsa *"Apply"* y se selecciona *"View"* → *"Coloring"* → *"U"*. El resultado debe ser semejante a la Figura 168.

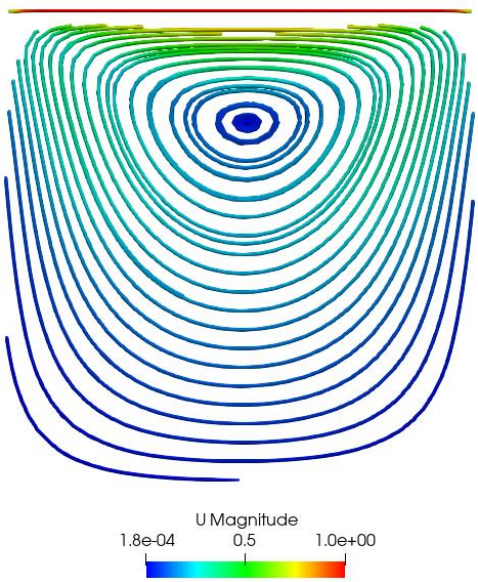


Figura 168. *Tube al StreamTracer de U.*

Con todos los filtros aplicados hasta ahora, la jerarquía de módulos del *Pipeline Browser* deber ser como se muestra en la Figura 169.

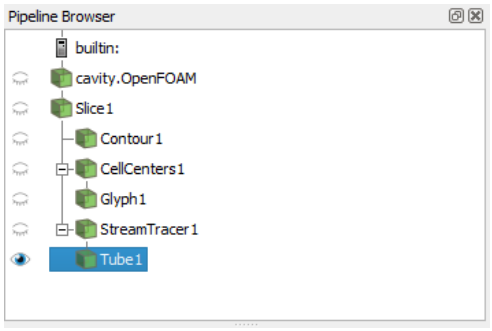


Figura 169. *Jerarquía de módulos usados.*

En este punto estarían vistos todos los filtros que típicamente son aplicables en problemas CFD, por lo que se contaría con los conocimientos básicos necesarios para postprocesar datos simulados en OpenFOAM® mediante ParaView® e interpretar los resultados.

Anexo VI

Guía de comandos de importación de geometría en OpenFOAM®

En este anexo se describe los archivos necesarios y ejecución de los comandos de OpenFOAM® *surfaceFeatureExtract* y *snappyHexMesh* necesarios para la importación de la geometría en el caso *SibiuPro*. Estos comandos no fueron utilizados durante la guía básica del tutorial *Lid-driven cavity flow* descrita en el anterior anexo porque carecía de geometría interna. La guía de ambos comandos está basada en el punto 5.4 del documento de OpenFOAM® llamado *OpenFOAM v5 User Guide* (The OpenFOAM Foundation, 2016).

Para seguir este anexo adecuadamente se entiende que el lector cuenta con conocimientos básicos de OpenFOAM® del Anexo V comentado anteriormente.

OpenFOAM® cuenta con el comando de preprocesado *snappyHexMesh* para generar geometrías malladas en 3 dimensiones internas al mallado externo predefinido mediante *blockMesh*. La utilidad *snappyHexMesh* necesita para ello extraer previamente la superficie geométrica 3D interna desde archivos en formato *.stl* (STereoLithography) o *.obj* (Wavefront Object) mediante el comando *surfaceFeatureExtract* y también utilidad de preprocesado.

Los archivos en formato *.stl* y *.obj* son archivos comunes de exportación desde herramientas informáticas de modelado 3D como por ejemplo SolidWorks®.

El mallado interno se ajusta aproximadamente a la superficie extraída mediante el refinado iterativo según el mallado externo. La especificación del nivel de refinamiento de la malla es muy flexible, con un manejo de la superficie robusto, resultado una calidad del mallado final bastante aceptable aunque dependiente en gran medida de la complejidad de la geometría a extraer.

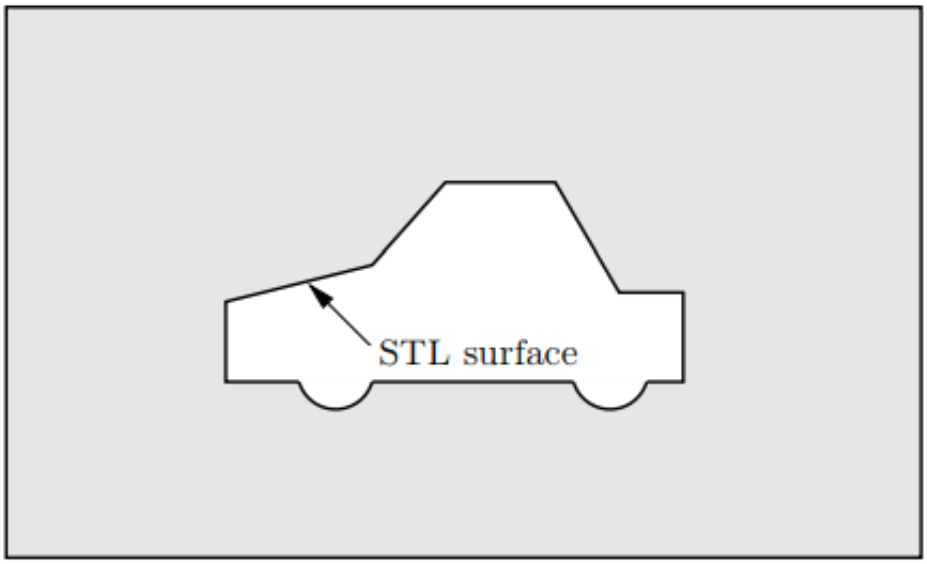


Figura 170. *Esquema 2D de un problema de mallado para snappyHexMesh.*

El proceso de generación del mallado usando *snappyHexMesh* se describirá a partir del esquema de la Figura 170, que aunque no describe el problema en 3 dimensiones, es completamente extrapolable y didácticamente más sencillo de entender. Es necesario recordar que *snappyHexMesh* es realmente una utilidad de mallado 3D.

VI.1 Generación del mallado mediante snappyHexMesh

El objetivo consiste en insertar el mallado 3D extraído sobre el volumen definido mediante el archivo *blockMeshDict*, que, como puede verse en la Figura 170, es un problema típico de simulación aerodinámica externa.

Para ejecutar *snappyHexMesh* se requiere lo siguiente:

- Al menos un archivo que describa la superficie 3D a insertar, en formato *.stl* o *.obj* localizado en el directorio *<caso>/constant/triSurface*.
- Un mallado hexagonal externo previamente creado ejecutando *blockMesh*.
- Un archivo *snappyHexMeshDict*, con las entradas correctamente definidas y localizado en el directorio *<caso>/system*.

El archivo *snappyHexMeshDict* incluye interruptores en las primeras líneas de código para activar o desactivar etapas del proceso del mallado. La activación de los interruptores conlleva que se ejecuten los subdiccionarios y las funciones que estos incluyen. Sus palabras claves son las siguientes:

- *castellatedMesh*. Interruptor para activar el mallado interno almenado.
- *snap*. Interruptor para activar el mallado interno de ajuste superficial.
- *addLayers*. Interruptor para activar la adición de capa superficial.
- *mergeTolerance*. Fusiona la tolerancia como fracción de la frontera del mallado externo.
- *geometry*. Subdiccionario que incluye las superficies geométricas del mallado interno.
- *castellatedMeshControls*. Subdiccionario que recoge los subdiccionarios y funciones incluidos en *castellatedMesh*
- *snapControls*. Subdiccionario que recoge los subdiccionarios y funciones incluidos en *snap*.
- *addLayersControls*. Subdiccionario que recoge los subdiccionarios y funciones incluidos en *addLayers*.
- *meshQualityControls*. Subdiccionario de control de la calidad del mallado interno.

Toda la geometría usada por *snappyHexMesh* es especifica en el subdiccionario *geometry* dentro de *snappyHexMeshDict*. La geometría puede ser definida a través de archivos *.stl* o *.obj*, o mediante una geometría de entidades de frontera de OpenFOAM®. Un ejemplo puede verse en la siguiente captura.

```
geometry
{
    sphere1     // User defined region name
    {
        type    triSurfaceMesh;
        file    "sphere1.obj";  // surface geometry OBJ file
        regions
        {
            secondSolid             // Named region in the OBJ file
            {
                name mySecondPatch; // User-defined patch name
            }                       // otherwise given sphere1_secondSolid
        }
    }

    box1x1x1    // User defined region name
    {
        type    searchableBox;       // region defined by bounding box
        min     (1.5 1 -0.5);
        max     (3.5 2 0.5);
    }

    sphere2     // User defined region name
    {
        type    searchableSphere;    // region defined by bounding sphere
        centre  (1.5 1.5 1.5);
        radius  1.03;
    }
};
```

Figura 171. *Ejemplo de definición de geometría interna en snappyHexMeshDict.*

En la Figura 171, se define la geometría *sphere1* mediante un archivo *.obj* mientras que, por otra parte se definen las geometrías *box1x1x1* y *sphere2* mediante su tipo y sus palabras clave y entradas de definiciones geométricas de OpenFOAM®.

VI.2 Creación del mallado externo

Antes de ejecutar *snappyHexMesh* el usuario debe crear el mallado externo de celdas hexaédricas que incluirá el mallado interno, tal como se muestra en la Figura 172.

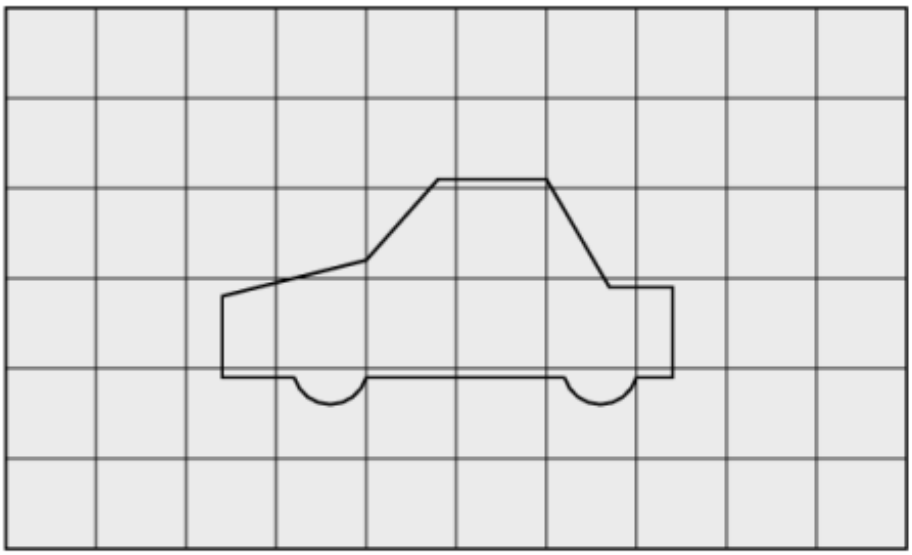


Figura 172. *Mallado externo. snappyHexMesh sin ejecutar.*

El mallado externo se puede crear simplemente ejecutando el comando *blockMesh*. El mallado definido en el archivo *blockMeshDict* debe seguir el siguiente criterio para que sea compatible con la ejecución posterior de *snappyHexMesh*:

- El mallado debe consistirse únicamente de celdas hexaédricas.
- Se debe definir celdas perfectamente o casi puramente cúbicas. Si no es así, la convergencia del procedimiento de ajuste será lenta, posiblemente hasta el punto de fallar la ejecución del comando.
- Debe existir al menos una intersección de las caras de alguna celda con la superficie de la geometría interna. Por ejemplo, un mallado externo de una sola celda no funcionará porque las caras de la celda no cortaran la superficie de la geometría si esta es completamente interna.

Antes de ejecutar *snappyHexMesh* y con el mallado externo creado, es muy recomendable hacerse una idea de la posición exacta de la geométrica interna de la cual se quiere realizar el mallado interno a partir de archivos *.stl* o *.obj*. Para ello simplemente habría que abrir el mallado externo en ParaView®, y en este importar el archivo *.stl* o *.obj*. El mallado debe definirse conforme a la posición y volumen que ocupa la geometría interna, de tal forma

que simplemente reconfigurando el archivo *blockMeshDict*, se construya una mallado externo adecuado. Puede observarse el ejemplo de visualización de *cylinder.stl* en la Figura 173.

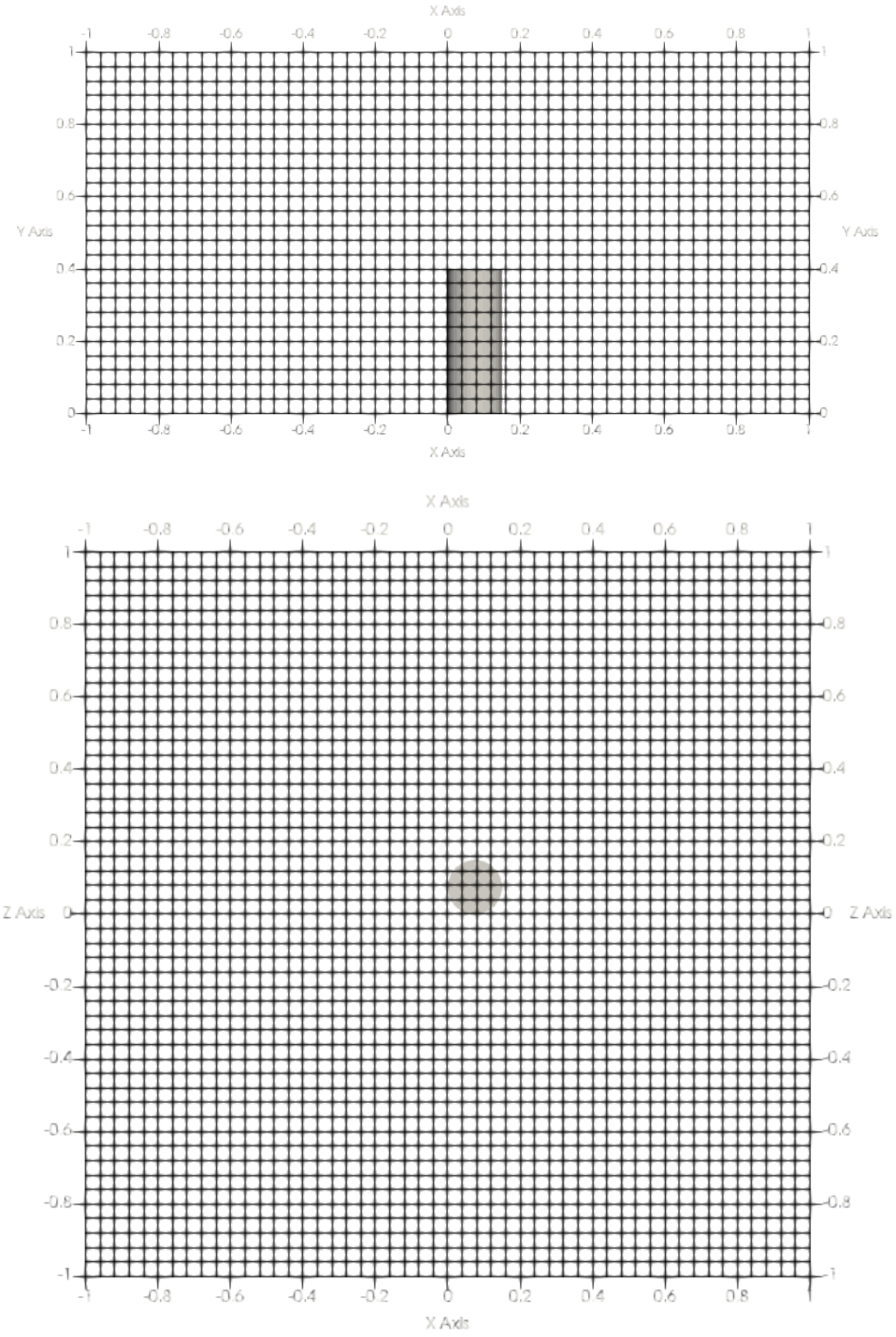


Figura 173. *Alzado y planta del ejemplo cylinder.stl contenido en un mallado externo.*

VI.3 Refinado de celdas con castellatedMesh

La división de celdas se realiza de acuerdo con la *especificación* definida por el usuario en el subdiccionario *castellatedMeshControls* de snappyHexMeshDict. Las palabras clave para los controles de *castellatedMesh* se presentan a continuación.

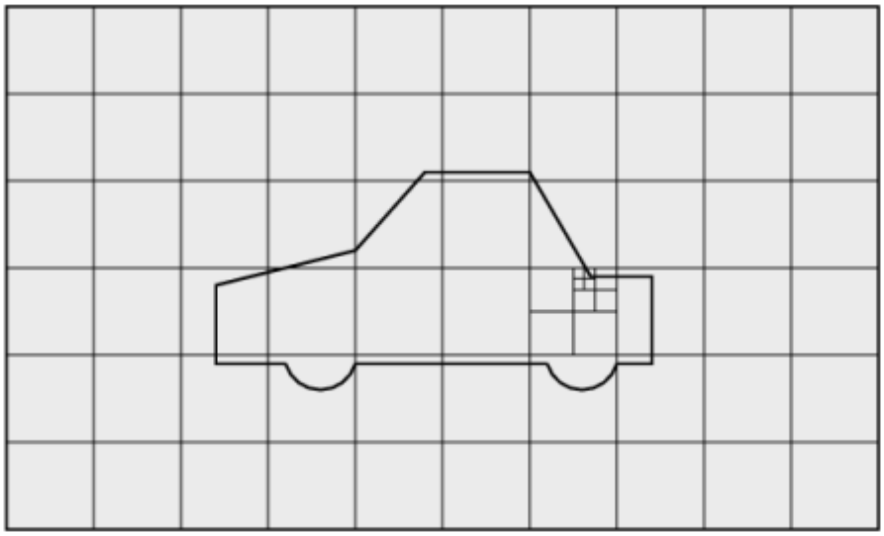


Figura 174. *Refinado de una celda de castellatedMesh.*

- *locationInMesh*. Vector de posición dentro del llamado interno. Este no puede coincidir con la cara de una celda antes y durante el refinado.
- *maxLocalCells*. Máximo número de celdas durante el refinado.
- *maxGlobalCells*. Límite total de celdas durante el refinado (antes del borrado).
- *minRefinementCells*. si *minRefinementCells* ≥ número de celdas a refinar, termina el refinado superficial.
- *nCellsBetweenLevels*. Número de capas de celdas entre niveles sucesivos de refinado (normalmente se configura a 3).
- *resolveFeatureAngle*. Aplica el máximo nivel de refinado a las celdas de aristas cuyos ángulos sobrepasen la entrada configurada (normalmente se configura a 30).
- *features*. Función que define el refinado mínimo
- *refinementSurfaces*. Subdiccionario de nivel refinado de superficies.
- *refinementRegions*. Subdiccionario de nivel de refinado de regiones.

El proceso de refinado comienza seleccionando las celdas cuyas caras que se cortan con la geometría interna en formato *.eMesh* (archivo *edgeMesh*). Obtener la geometría interna en

formato *.eMesh* conlleva la ejecución de un comando previo a *snappyHexMesh*, llamado *surfaceFeatureExtract*. Para que su ejecución se realice sin errores, debe crearse el archivo *surfaceFeatureExtractDict*, con las entradas correctamente definidas y localizado en el directorio *<caso>/system*.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

cylinder.stl
{
    extractionMethod    extractFromSurface;

    extractFromSurfaceCoeffs
    {
        includedAngle   150;
    }

    writeObj                yes;
}

// ************************************************************************** //
```

Figura 175. *surfaceFeatureExtractDict de cylinder.stl.*

La Figura 175 recoge la definición de extracción de la geometría del ejemplo *cylinder.stl*. Primero se debe definir el nombre del subdiccionario de extracción, que debe coincidir exactamente con el nombre del archivo de extensión *.stl* o *.obj* (EXTENSIÓN EN MINÚSCULAS) previamente incluido en el directorio *<caso>/constant/triSurface*. En este archivo se pueden configurar dos entradas. Por una parte se configura *includedAngle* para especificar la calidad de la extracción superficial: configurarlo a *0* no extraerá ninguna arista (ya que todas tendrán un ángulo), mientras que configurarlo a *180* extraerá todas las aristas (ya que cualquier arista se define entre *0* y *180* grados). Por otra parte se configura *writeObj* a *yes* o *no* para especificar si se quiere extraer archivos *.stl* o *.obj* de regiones de la geometría interna que pueden ser útiles para el postprocesado en ParaView®.

Una vez definido y ubicado *surfaceFeatureExtractDict*, se ejecuta el comando.

```
surfaceFeatureExtract
```

La ejecución sin errores crea el directorio *<caso>/constant/extendedFeatureEdgeMesh*, dónde se ubica el archivo de mismo nombre que la geometría interna pero en formato *.eMesh*, además de los archivos *.stl* o *.obj* de regiones de la geometría si se configuró *writeObj* a *yes* previamente a la ejecución de *surfaceFeatureExtract*.

Siguiendo con el proceso de refinado de celdas de *snappyHexMesh*, primero se comprueba qué celdas del mallado externo intersecan con la geometría interna extraída en el archivo *.eMesh*. Cada una de las celdas intersecadas se divide en 8 celdas de iguales dimensiones y se vuelve a comprobar cuales de estas celdas dividas intersecan con la geometría interna extraída. *snappyHexMesh* repite el proceso y vuelve a dividir las celdas dividas tantas veces como mínimamente se definió en el nivel de refinado configurado en la entrada de *features*. La división de celdas puede verse en la Figura 176 para el ejemplo visto en 2 dimensiones.

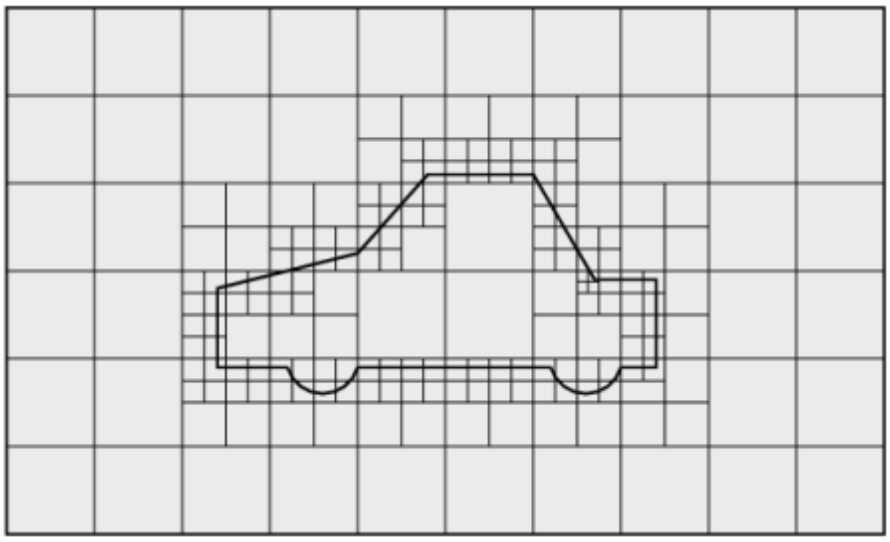


Figura 176. *Refinado de celdas de varios niveles de castellatedMesh.*

En el subdiccionario *refinementSurfaces* se definen entradas para especificar el nivel de refinado mínimo y máximo (<min> <max>) de geometrías internas concretas si es que se extrajo más de un archivo *.stl* o *.obj*. El refinado máximo se aplicará únicamente en las aristas más críticas de la geometría interna, es decir, las de menor ángulo y que sean superiores al definido en *resolveFeatureAngle*.

El refinado puede ser desactivado opcionalmente en una o más regiones específicas de la geometría interna. Las distintas subdiccionarios de las distintas superficies de la geometría interna se recopilan en un subdiccionario llamado *regions*. En cada subdiccionario de regiones se especifica el nivel de refinado concreto que define el usuario. Un ejemplo se muestra en la Figura 177.

```
refinementSurfaces
{
    sphere1
    {
        level (2 2); // default (min max) refinement for whole surface
        regions
        {
            secondSolid
            {
                level (3 3); // optional refinement for secondSolid region
            }
        }
    }
}
```

Figura 177. *Refinado nivel 3 de secondSolid dentro de la geometría shpere1 nivel 2.*

VI.4 Borrado de celdas con castellatedMesh

Una vez se ha completado el proceso de refinado de celdas, comienza el borrado de todas aquellas celdas interiores al mallado interno. La eliminación de celdas requiere de uno o más regiones volumétricas encerradas por completo por una superficie 3D delimitadora. Cada región se identifica mediante un vector de posición dentro de la región, definido mediante la palabra clave de *locationInMesh*. La condición para el borrado de celdas es verificar si el 50% o más del volumen de la celda está contenido en la región, independientemente de su tamaño. El resultado del borrado se muestra en la Figura 178.

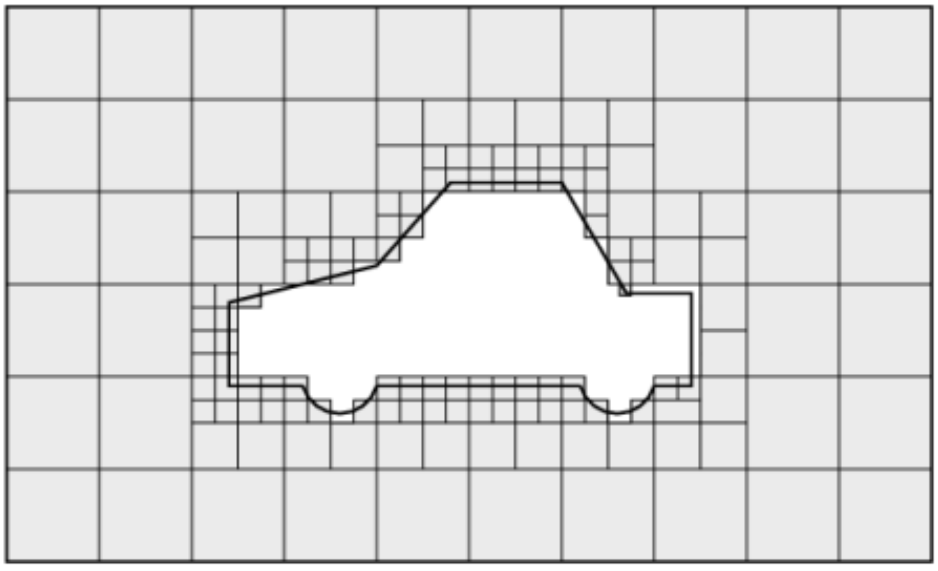


Figura 178. *Borrado de celdas de castellatedMesh.*

VI.5 Refinado en regiones específicas con castellatedMesh

Los problemas típicos de aerodinámica externa requieren muchas veces de un mallado refinado no solo en la superficie objeto, sino también en sus proximidades y parte trasera, para así estudiar adecuadamente la evolución de presiones, velocidades y turbulencias que sufre. El subdiccionario *refinementRegions* permite seleccionar regiones prismáticas rectangulares definidas en el subdccionario *geometry* para que las celdas que se encuentran dentro se puedan refinar a un nivel definido en la entrada *levels*. El modo de refinado se define de tres formas:

- *inside*. Refina dentro de la región volumétrica.
- *outside*. Refina fuera de la región volumétrica.
- *distance*. Refina según la distancia a la superficie. Puede definir diferentes niveles a múltiples distancias con la entrada de *levels*.

```
refinementRegions
{
    box1x1x1
    {
        mode inside;
        levels ((1.0 4));         // refinement level 4 (1.0 entry ignored)
    }

    sphere1
    {                             // refinement level 5 within 1.0 m
        mode distance;            // refinement level 3 within 2.0 m
        levels ((1.0 5) (2.0 3)); // levels must be ordered nearest first
    }
}
```

Figura 179. *Ejemplo de tipos de refinado de regiones.*

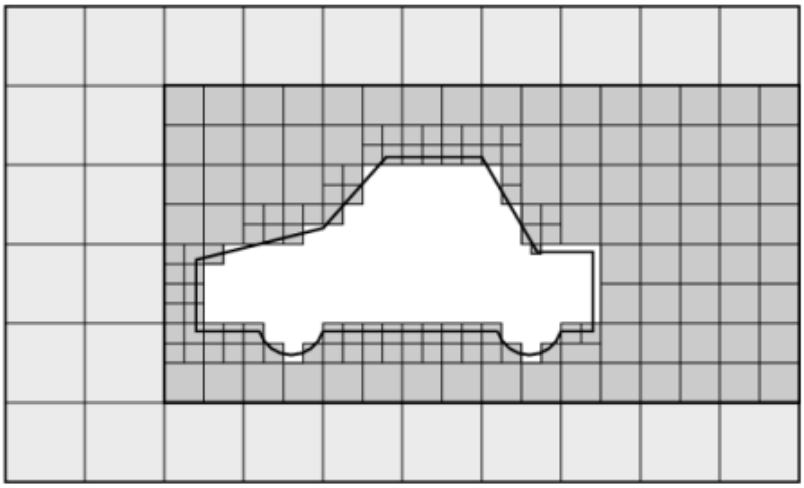


Figura 180. *Refinado de nivel 1 en una región rectangular.*

VI.6 Suavizado de superficies con snap

El siguiente paso del mallado mediante *snappyHexMesh* es el de suavizado de superficies, que consiste en mover los vértices de las celdas refinadas que cortan la superficie de la geometría interna para que se adapten lo máximo posible a esa geometría. De esta forma se elimina la superficie irregular almenada con *castellatedMesh* para esas celdas. El proceso consiste en los siguientes pasos:

1. Desplazar los vértices de la frontera interna almenada hacia la superficie del *.stl* o *.obj*.
2. Resolver la relajación del mallado interno con los vértices desplazados.
3. Encontrar los vértices que incumplen los parámetros de calidad del mallado.
4. Reducir el desplazamiento de esos vértices de su valor inicial en el punto 1 y repetir el punto 2 hasta que se satisfaga que ningún vértice se encuentra en el punto 3.

El método usado se ajusta en el subdiccionario *snapControls*, cuyas palabras claves en las siguientes:

- *nSmoothPatch*. Número de iteraciones de suavizado de parches (normalmente 3).
- *tolerance*. Ratio de distancia para que los puntos sean atraídos por el punto o borde de la superficie *.stl* o *.obj*., con la longitud máxima del borde local (normalmente 2.0).
- *nSolverIter*. Número de iteraciones de relajación del desplazamiento del mallado (normalmente 30-100).
- *nRelaxIter*. Máximo número de iteraciones relajación suavizada (normalmente 5).

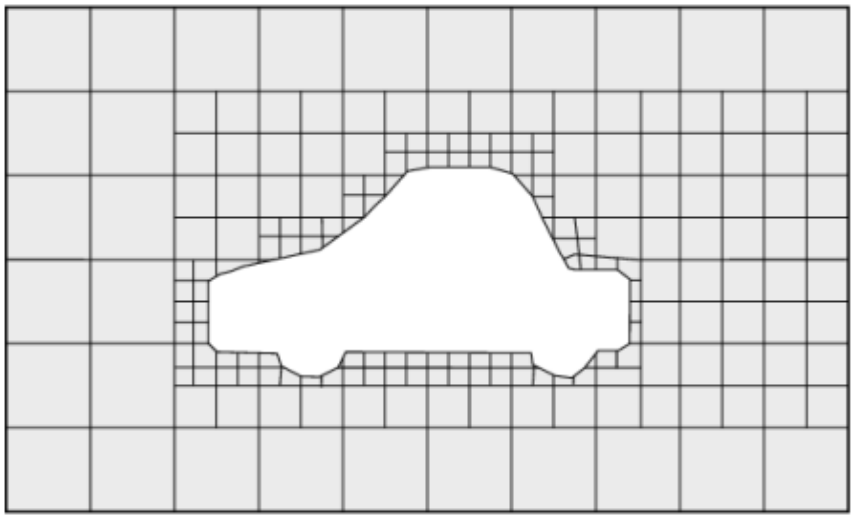


Figura 181. *Suavizado de superficies con snap.*

VI.7 Adición de capas con addLayers

El resultado del suavizado de superficies con *snap* puede adecuado según la calidad del mallado que se pretende, aunque puede producir irregulares a lo largo de las superficies de frontera. Existe una etapa opcional del mallado de *snappyHexMesh* que introduce capas adicionales de celdas hexaédricas alineadas con la superficie límite. Esto se muestra en las celdas sombreadas de la Figura 182.

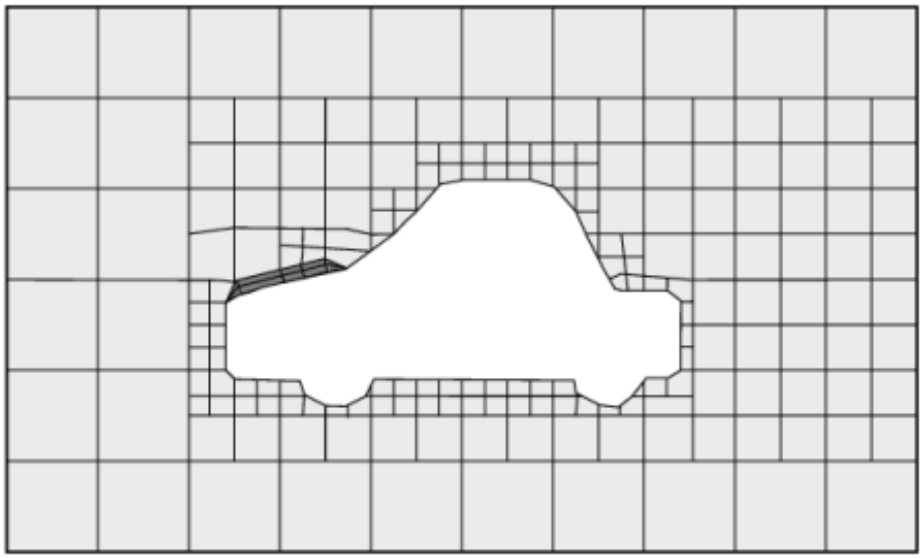


Figura 182. *Adición de capas con addLayers.*

El proceso de adición de capas consiste en encoger el mallado suavizado con *snap* desde la frontera e insertar capas de celdas de la siguiente manera:

1. El mallado se proyecta hacia atrás desde la superficie con un espesor especificado en la dirección normal a la superficie.
2. Se resuelve la relajación del mallado interno con los últimos vértices de la frontera proyectados.
3. Se comprueba si se cumplen los criterios de validación. De lo contrario se reduce el espesor especificado y se vuelve a 2. Si la validación no puede satisfacerse para espesores reducidos, no se añaden capas.
4. Si el criterio de validación se satisface, se añade la capa.
5. El mallado se revisa otra vez. Si las comprobaciones fallan, las capas añadidas se borran y se vuelve al punto 2.

El método usado se ajusta en el subdiccionario *addLayersControls*. El usuario tiene la opción elegir dos de los cuatro diferentes parámetros de espesor de capa: *expansionRatio*, *finalLayerThickness*, *firstLayerThickness* y *thickness*. Las palabras claves son las siguientes:

- *layers*. Subdiccionario que define las capas a insertar.
- *realtiveSizes*. Interruptor que establece si los espesores de capa especificados son relativos al tamaño de celda no distorsionado fuera de la capa o absolutos.
- *expansionRatio*. Factor de expansión de la capa adicionada, incrementada en tamaño desde una capa a la siguiente.
- *finalLayerThickness*. Espesor de la última capa adicionada del mallado interno (alejada), normalmente se combina con tamaños relativos de acuerdo con la entrada *relativeSizes*.
- *firstLayerThickness*. Espesor de la primera capa adicionada del mallado interno (cercana), normalmente se combina con tamaños relativos de acuerdo con la entrada *relativeSizes*.
- *thickness*. Espesor total de todas las capas adicionadas, normalmente se combina con tamaños relativos de acuerdo con la entrada *relativeSizes*.
- *minThickness*. Mínimo espesor de capa de celdas añadida, ya se relativa y/o absoluta.
- *nGrow*. número de capas de caras conectadas que no crecen si los puntos no se extruyen; ayuda a la convergencia de la adición de capas cerca de los parches.
- *featureAngle*. Ángulo por encima del cual la superficie no se extruye.
- *nRelaxIter*. Máximo número de iteraciones de relajación de suavizado (normalmente 5).
- *nSmoothSurfaceNormals*. Número de iteraciones de superficie suavizada (“ “ 1).
- *nSmoothNormals*. número de iteraciones de suavizado de la dirección del movimiento del mallado interior (“ “ 3).
- *nSmoothTickness*. espesor de capa suavizada sobre la superficie de los parches (“ “ 10).
- *maxFaceThicknessRatio*. Detiene el crecimiento de la capa en celdas deformadas (“ “ 3).
- *maxThicknessToMediaRatio*. Reduce el crecimiento de la capa cuando la relación entre el grosor y la distancia media es grande (“ “ 0,3).
- *minMedianAxisAngle*. Ángulo utilizado para captar puntos del eje medio (“ “ 90).
- *nBufferCellsNoExtrude*. Crea una región amortiguada para nuevos acabados de capa (“ “ 0).
- *nLayersIter*. Número máximo total de iteraciones de adición de capa (“ “ 50).
- *nRelaxIter*. Número máximo de iteraciones después de las cuales se usan los controles en el subdiccionario relajado de *meshQuality* (“ “ 20).

El subdiccionario *layers* contiene entradas para cada parche en el que se aplicarán las capas y el número de capas de superficie necesarias. El nombre del parche es necesario para relacionar la adición de capas con el mallado suavizado con *snap*, no con la geometría *.stl* o *.obj*, por lo que se aplica a un parche, no a una región de una superficie. Un ejemplo de subdiccionario *layers* es el que se muestra en la Figura 183.

```
layers
{
    sphere1_firstSolid
    {
        nSurfaceLayers 1;
    }
    maxY
    {
        nSurfaceLayers 1;
    }
}
```

Figura 183. *Ejemplo de layers.*

VI.8 Control de calidad del mallado con meshQualityControls

La calidad del mallado se controla mediante las entradas del subdiccionario *meshQualityControls*. Las palabras clave de cada entrada son las siguientes:

- *maxNonOrtho*. Máximo ángulo no ortogonal permitida (normalmente 65).
- *maxBoundarySkewness*. Máxima asimetría de la cara frontera permitida (" " 20).
- *maxInternalSkewness*. Máxima asimetría de la cara frontera permitida (" " 4).
- *maxConcave*. Máxima asimetría de la cara interna permitida (" " 80).
- *minFlatness*. Ratio entre el área mínima proyectada y el área real (" " 0.5).
- *minTetQuality*. Calidad mínima de las celdas tetraédricas por descomposición de celdas (normalmente desactivado).
- *minVol*. Mínimo volumen de celda piramidal (" " 1e-13).
- *minArea*. Mínimo área de cara (" " -1).
- *minTwist*. (" " 0.05).
- *minDetermiant*. Determinante de celda mínimo normalizado; 1 = hexaédrico; ≤ 0 = celda ilegal (" " 0.001).
- *minFaceWeight*. 0 → 0.5 (" " 0.05).
- *minVolRatio*. 0 → 1.0 (" " 0.01).
- *minTriangleTwist*. > 0 para compatibilidad *Fluent* (" " -1).
- *nSmoothScale*. Número de errores de distribución de errores (" " 4).
- *errorReduction*. Cantidad para reducir el desplazamiento en el punto de error (" " 0.05).
- *relaxed*. Subdiccionario que puede incluir valores modificados para las entradas de las palabras clave de *meshQualityControls* que se utilizan cuando se exceda *nRelaxedIter* durante la adición de capas.

Anexo VII

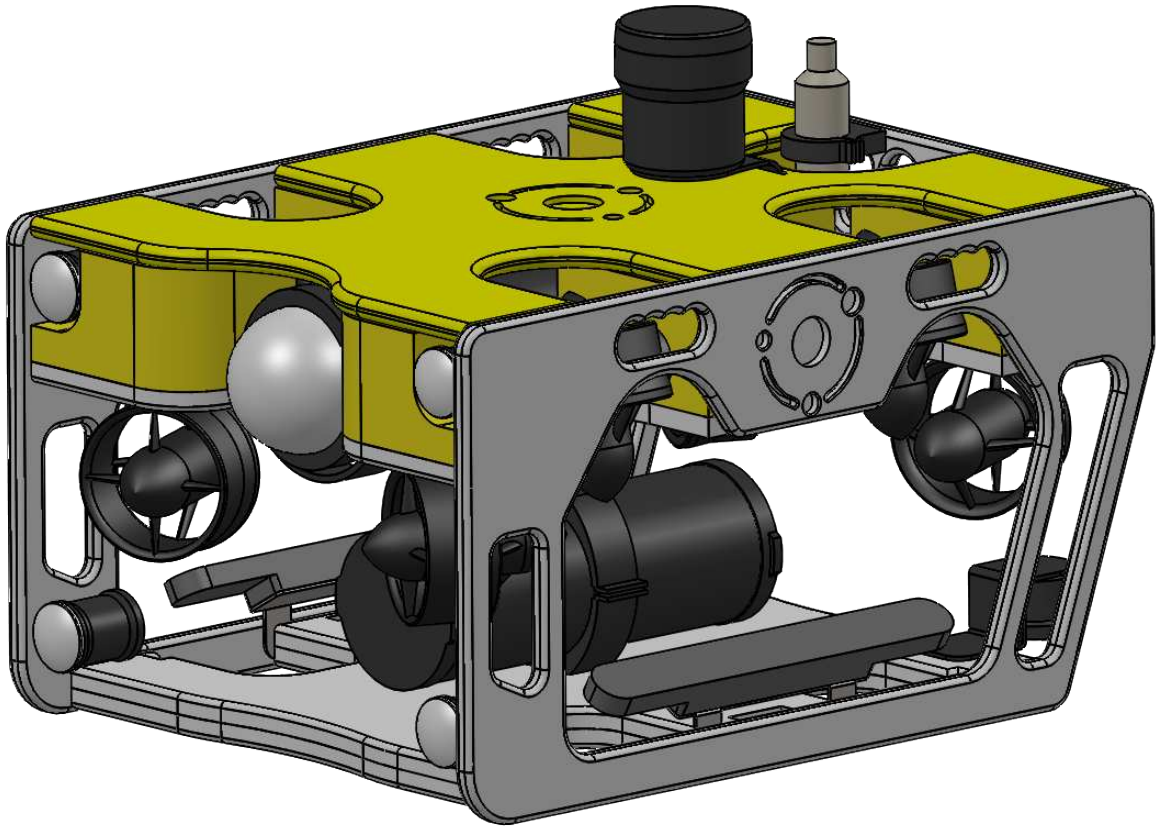
Planos

VII.1 Ensamblaje del ROV

VII.2 Explosionado del ROV

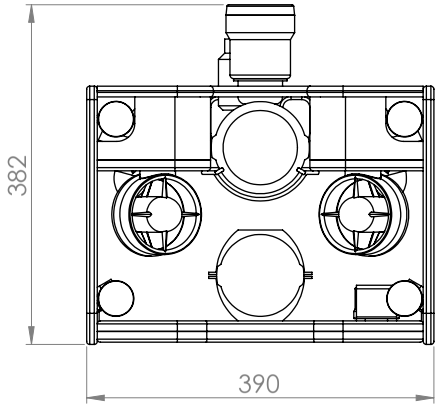
VII.3 Piezas colector

VII.4 Conjunto colector

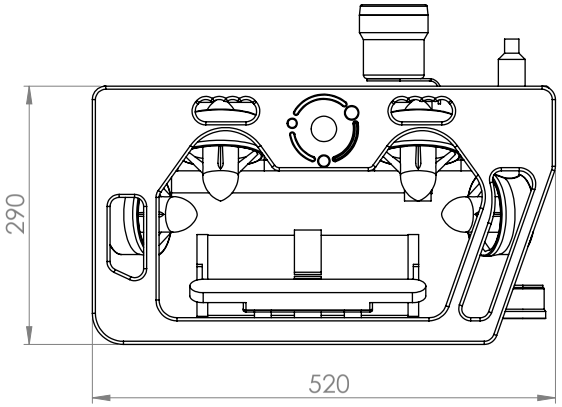


ID	PIEZA	UDS
1	Tapa superior	1
2	Chasis inferior	2
3	Chasis posterior	1
4	Chasis lateral izquierdo	1
5	Chasis lateral derecho	1
6	Chasis medio derecho	1
7	Flotador delantero derecho	1
8	Separador transversal	4
9	Separador longitudinal	2
10	Flotador medio	2
11	Flotador trasero	2
12	Chasis medio izquierdo	1
13	Flotador delantero izquierdo	1
14	Soporte SBL	2
15	Sonar de barrido lateral	2
16	Faro	4
17	Soporte ecosonda	1
18	Ecosonda	1
19	Soporte localizador S1	1
20	Localizador S1	1
21	Soporte sonar 360	1
22	Sonar 360	1
23	Soporte batería	2
24	Soporte cilindro de electrónica	2
25	Cilindro de electrónica	1
26	Protección cámara	1
27	Propulsor	8
28	Batería	1

VISTA FRONTAL



VISTA LATERAL



E = 1:12

TFM.

Simulación hidrodinámica y validación experimental de un vehículo submarino operado remotamente

ENSAMBLAJE DEL ROV

Modelo. Sibiu Pro

Plano nº. 1

Escala. 1:6

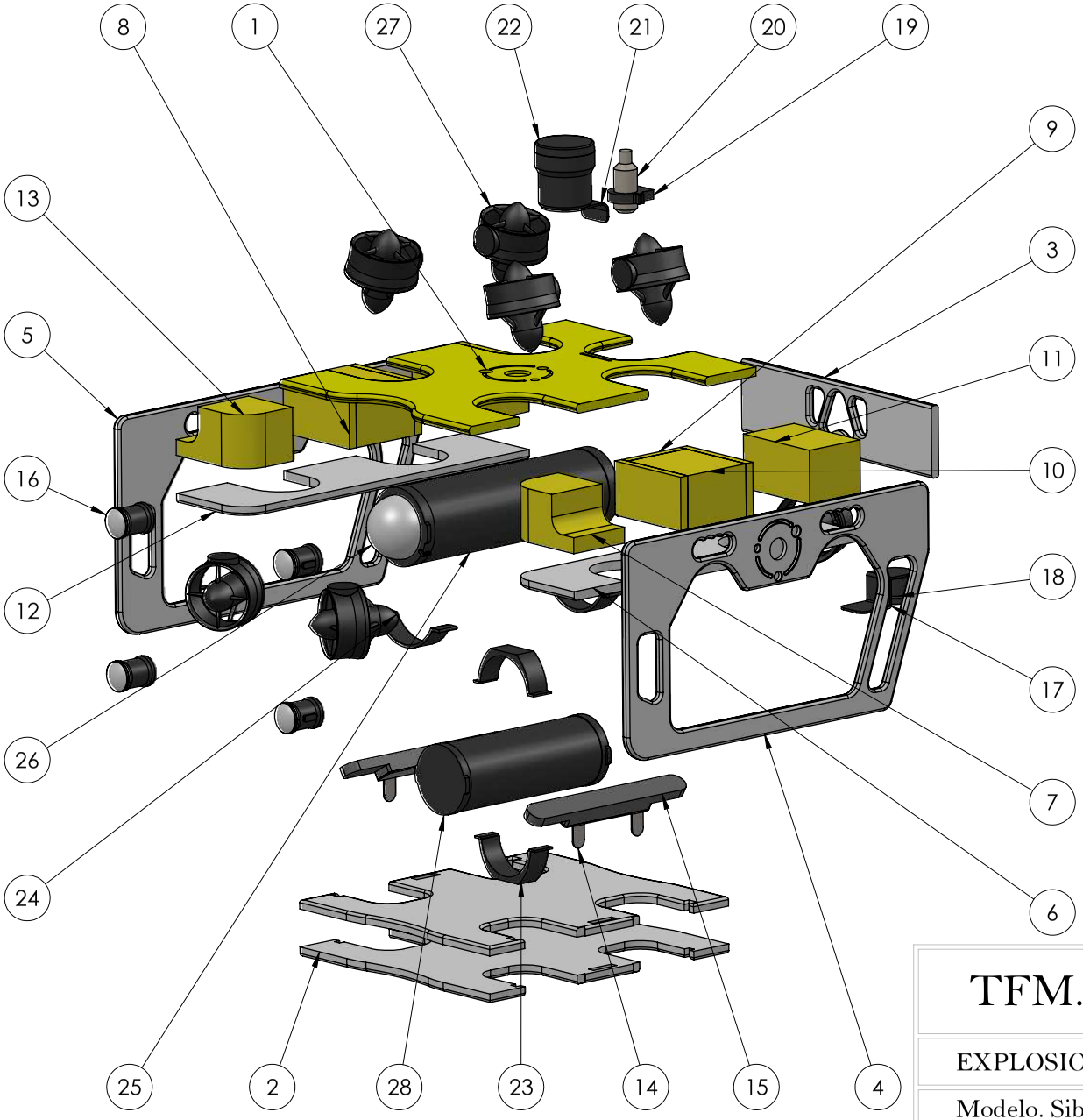
Realizado por:
Juan José Toscano Angulo

Comprobado por:
Inmaculada Pulido Calvo
Juan Carlos Gutiérrez Estrada

Máster en Ingeniería Industrial

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
Universidad de Huelva

ETSI ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA



ID	PIEZA	UDS
1	Tapa superior	1
2	Chasis inferior	2
3	Chasis posterior	1
4	Chasis lateral izquierdo	1
5	Chasis lateral derecho	1
6	Chasis medio derecho	1
7	Flotador delantero derecho	1
8	Separador transversal	4
9	Separador longitudinal	2
10	Flotador medio	2
11	Flotador trasero	2
12	Chasis medio izquierdo	1
13	Flotador delantero izquierdo	1
14	Soporte SBL	2
15	Sonar de barrido lateral	2
16	Faro	4
17	Soporte ecosonda	1
18	Ecosonda	1
19	Soporte localizador S1	1
20	Localizador S1	1
21	Soporte sonar 360	1
22	Sonar 360	1
23	Soporte batería	2
24	Soporte cilindro de electrónica	2
25	Cilindro de electrónica	1
26	Protección cámara	1
27	Propulsor	8
28	Batería	1

TFM.

Simulación hidrodinámica y validación experimental de un vehículo submarino operado remotamente

EXPLOSIONADO DEL ROV

Modelo. Sibiu Pro

Plano nº. 2

Escala. 1:12

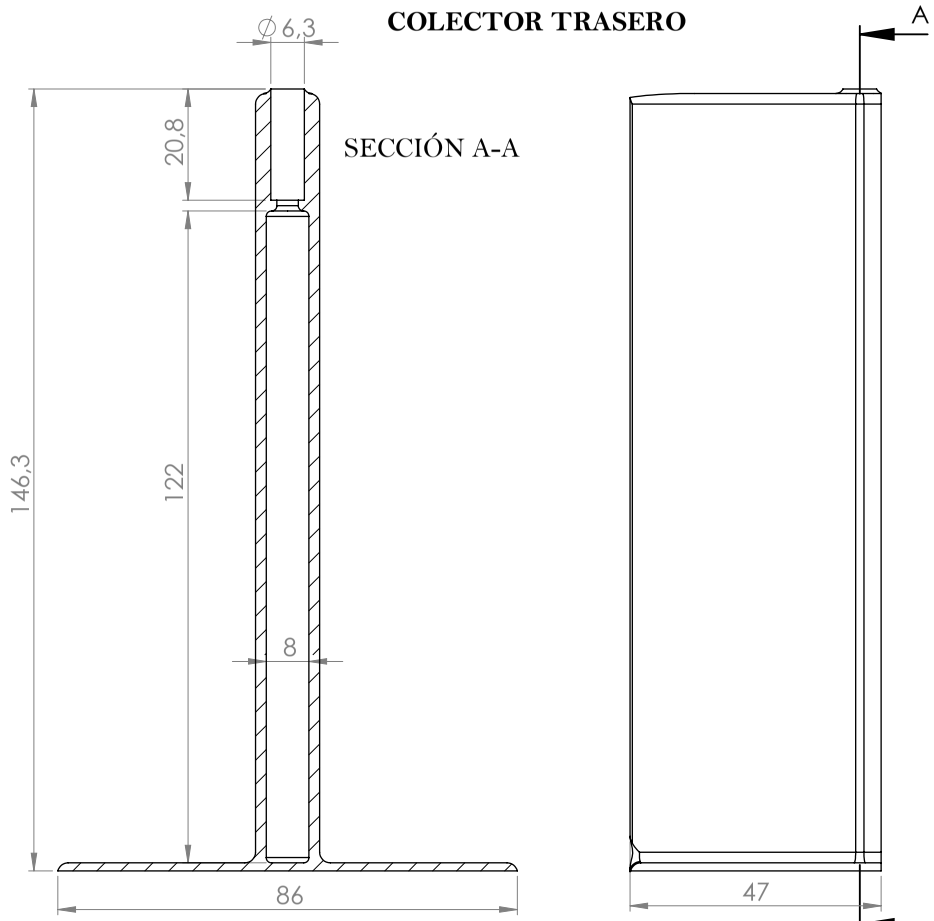
Realizado por:
Juan José Toscano Angulo
Comprobado por:
Inmaculada Pulido Calvo
Juan Carlos Gutiérrez Estrada

Máster en Ingeniería Industrial

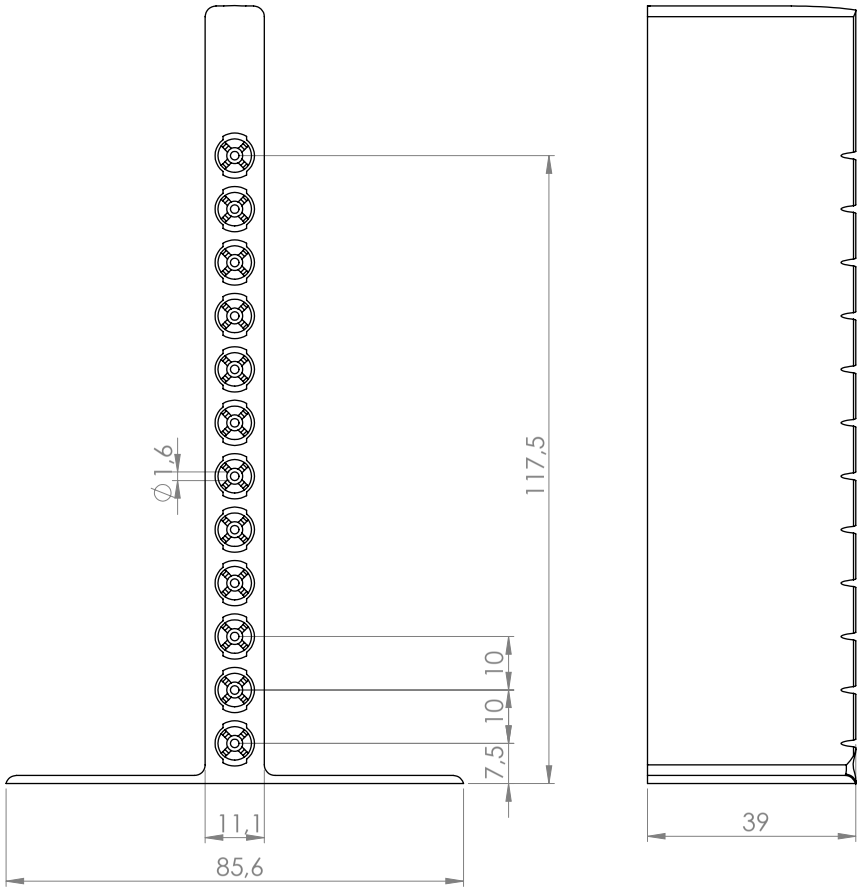
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
Universidad de Huelva

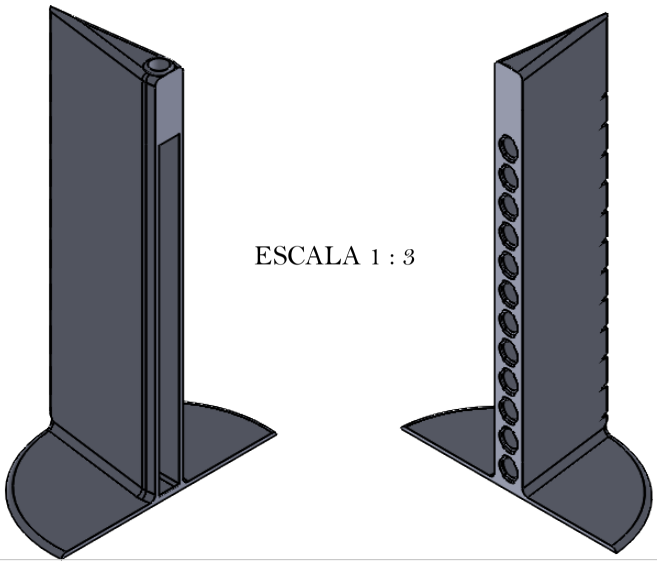
ETSI ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA

COLECTOR TRASERO



COLECTOR FRONTAL





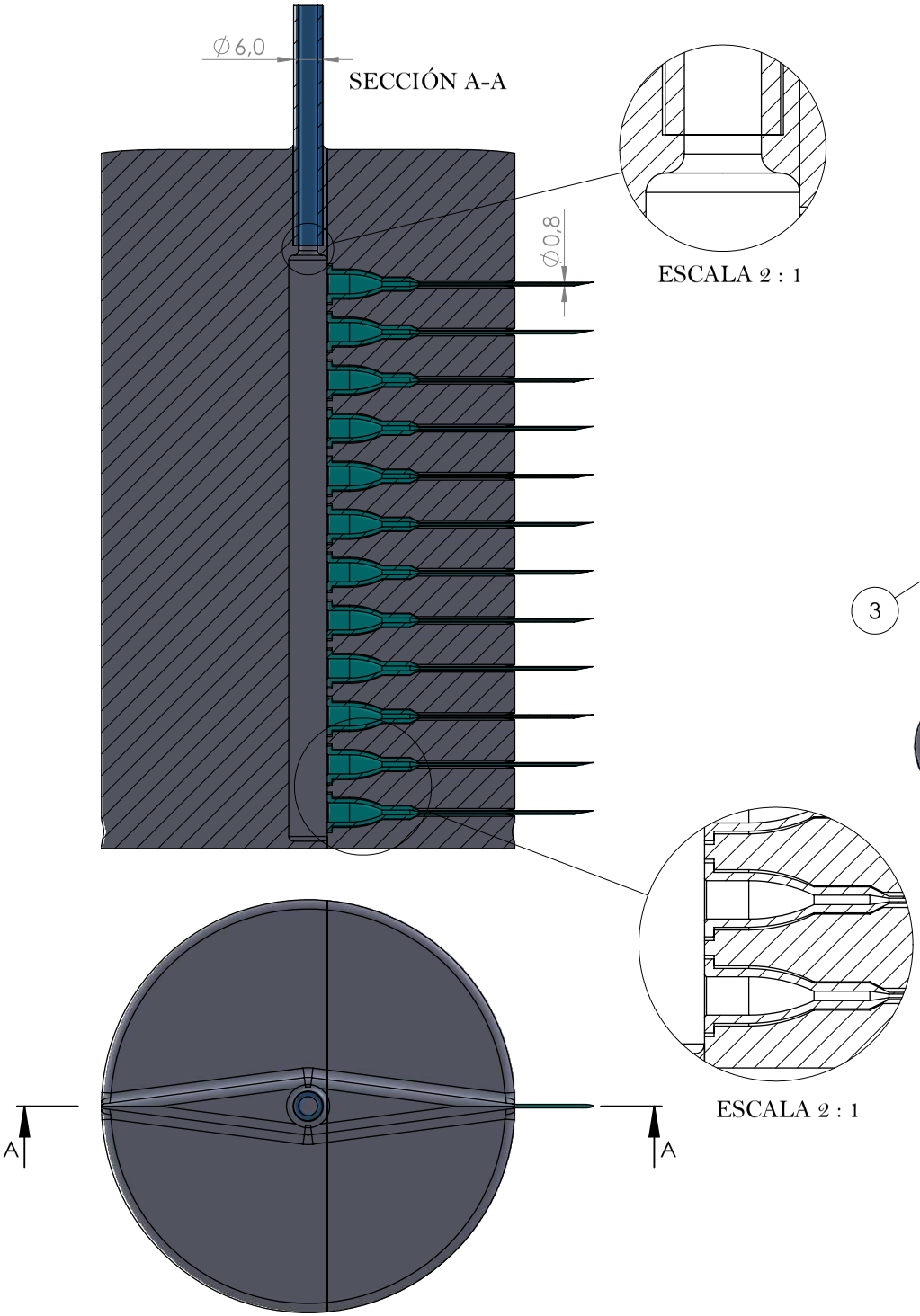
ESCALA 1 : 3

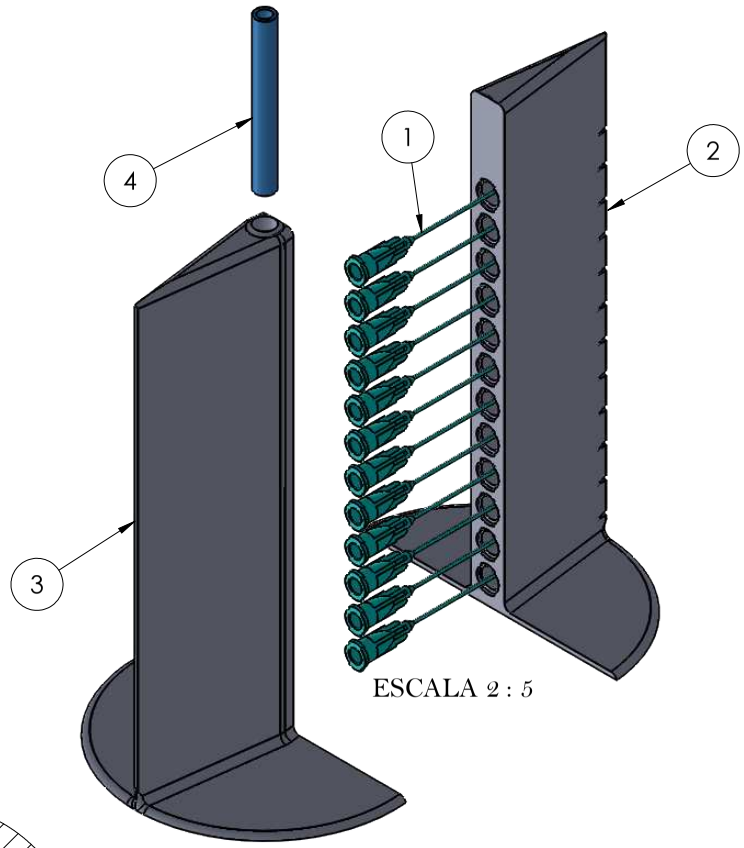
TFM.

Simulación hidrodinámica y validación experimental de un vehículo submarino operado remotamente

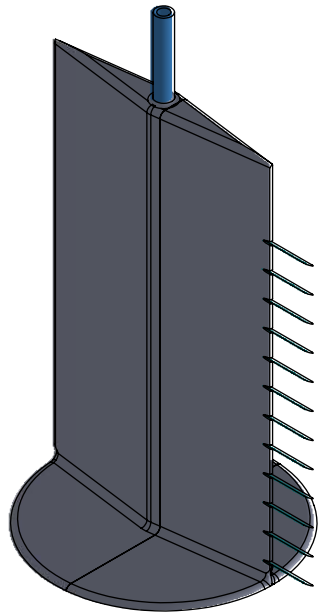
PIEZAS COLECTOR	Realizado por: Juan José Toscano Angulo
Modelo.	Comprobado por: Inmaculada Pulido Calvo, Juan Carlos Gutiérrez Estrada
Plano nº. 3 — Escala. 1:2	
Máster en Ingeniería Industrial	ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Universidad de Huelva

ID	N.º DE PIEZA	UDS
1	Aguja hipodérmica	12
2	Colector frontal	1
3	Colector trasero	1
4	Tubo flexible	1





ESCALA 2 : 5



ESCALA 1 : 3

TFM.

Simulación hidrodinámica y validación experimental de un vehículo submarino operado remotamente

CONJUNTO COLECTOR

Modelo.

Plano nº. 4 | Escala. 1:2

Realizado por:
Juan José Toscano Angulo
Comprobado por:
Inmaculada Pulido Calvo
Juan Carlos Gutiérrez Estrada

Máster en Ingeniería Industrial

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
Universidad de Huelva

ETSI ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA

Bibliografía

Anderson, J. D. (1995). *Computational Fluid Dynamics: The Basics With Applications.* McGraw-Hill. Obtenido de https://www.airloads.net/Downloads/Textbooks/Computational-Fluid-Dynamics-the-Basics-With-Applications-Anderson-J-D.pdf

Bravo-Córdoba, F. J., Fuentes-Pérez , J. F., Valbuena-Castro, J., Martínez de Azagra-Paredes, A., & Sanz-Ronda, F. J. (2021). *Turning Pools in Stepped Fishways: Biological Assessment via Fish Response and CFD Models.* Water, 13, 1186. Obtenido de https://doi.org/10.3390/w13091186

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control* (1st ed.). Wiley-Blackwell.

Gabl, R., Davey, T., Cao, Y., Li, Q., Li, B., Walker, K.L., Giorgio-Serchi, F., Aracri, S., Kiprakis, A., Sotkes, A.A. & Ingram, D.M. (2020). *Experimental Force Data of a Restrained ROV under Waves and Current.* Data, 5, 57. Obtenido de https://www.mdpi.com/2306-5729/5/3/57

Garrido Pellicer, A. (S.F.). *Estimación de los coeficientes hidrodinámicos de vehículos autonomos submarinos mediante CFD.* Proyecto Fin de Carrera, Universidad Politécnica de Cartagena, Cartagena.

Gutiérrez Estrada, J. C., Peregrín Rubio, A., Gómez Bravo, F., Pulido Calvo, I., & González Cabrera, M. (2019). *Informe introductorio sobre desarrollo y evaluación de sistemas fijos (boyas), pequeños vehículos marinos (ROVs) y análisis de datos e imágenes en zonas estuáricas y explotaciones acuícolas. Informe técnico de la Acción 1.3 del Proyecto KTTSeaDrones cofinanciado por el Fondo Europeo de Desarrollo Regional FEDER a través del Programa Interreg V-A España-Portugal (POCTEP) 2014-2020.* Universidad de Huelva, Ciencias Agroforestales, Huelva. Obtenido de https://kttseadrones.wixsite.com/kttseadrones

Gutiérrez-Estrada, J. C., Gómez-Bravo, F., Peregrín, A., Pulido-Calvo, I., Bethencourt, M., Barbero, L., Semião, J., Martínez-Ceada, I. & Escobar-zamora, M. 2020a. *Proyecto KTTSeaDrones: Drones aéreos y acuáticos para el sector pesquero-acuícola.* Industrias Pesqueras, noviembre 2020, nº 2163: 68-69.

Gutiérrez-Estrada, J. C., Gómez-Bravo, F., Peregrín, A., Pulido-Calvo, I., Bethencourt, M., Barbero, L., Semião, J., Martínez-Ceada, I. & Escobar-zamora, M. 2020b. *Proyecto KTTSeaDrones: Conocimiento y soluciones tecnológicas para el sector pesquero-acuícola.* Ruta Pesquera y Naval, noviembre-diciembre 2020, nº 143: 27.

Katsui, T., Kajikawa, S., & Inoue, T. (2012). *Numerical investigation of flow around a ROV with crawler based.* ASME 2012 31st International Conference on Ocean, Offshore and Arctic Engineering. 1-6. Rio de Janeiro: OMAE.

Li, Q., Cao, Y., Li, B., Ingram, D. M., & Kiprakis, A. (2020). *Numerical Modelling and Experimental Testing of the Hydrodynamic Characteristics for an Open-Frame Remotely Operated Vehicle.* Journal of Marine Science and Engineering, 8, 688. Obtenido de https://www.mdpi.com/2077-1312/8/9/688/pdf

Menter, F. R. (1993). *Zonal Two Equation k-ω Turbulence Models for Aerodynamic Flows.* 23rd Fluid Dynamics, Plasmadynamics and Laser Conference, Orlando, FL, USA.

Moreno, H., Saltarén, R., Puglisi, L., Carrera, I., Cárdenas, P., & Álvarez, C. (2014). *Robótica Submarina: Conceptos, Elementos, Modelado y Control.* Revista Iberoamericana de Automática e Informática Industrial, 11, 3-19.

NidoRobotics. (s.f.). *Sibiu Pro Technical Details.* Obtenido de https://www.nidorobotics.com/sibiu-pro

OpenFOAM Foundation. (2011). openfoam.org. Obtenido de https://openfoam.org/

OpenFOAM *User Guide v2012. (2012). k-epsilon.* Obtenido de https://www.openfoam.com/documentation/guides/latest/doc/guide-turbulence-ras-k-epsilon.html

OpenFOAM Wiki. (2009). openfoamwiki.net. Obtenido de https://openfoamwiki.net/index.php/IcoFoam

OpenFOAM Wiki. (2020). *OpenFOAM Wiki - SimpleFoam.* Obtenido de https://openfoamwiki.net/index.php/SimpleFoam

OpenFOAM Wiki. (2021). openfoamwiki.net. Obtenido de https://wiki.openfoam.com/Main_Page

Paraview. (2007). Paraview. Obtenido de https://www.paraview.org/

Ramírez-Macías, J., Brongers, P., Rúa, S., & Vásquez, R. (2016). *Hydrodynamic modelling for the remotely operated vehicle Visor3 using CFD.* IFAC-PapersOnLine, 49-23, 187-192.

Satria, D., Wiryadinata, R., Esiswitoyo, D., Adji, M., Rosyadi, I., Listijorini, E., & Sunardi. (2014). *Hydrodynamic analysis of Remotely Operated Vehicle (ROV).* IOP Conf. Ser.: Mater. Sci. Eng. 645 012014

The OpenFOAM Foundation. (2016). CFD Direct. Obtenido de User Guide v5: https://cfd.direct/openfoam/user-guide-v5/

Torres Parish, R. (2017). *Comparación de Resultados de Análisis en CFD entre el Módulo FloEFD de CATIA V5 y Fluent.* Proyecto Fin de Máster, ETSI, Universidad de Sevilla, Sevilla.

Vélez Bermejo, C. (2020). *Modelación hidrodinámica de un vehículo submarino operado remotamente (ROV) usando CFD*. Trabajo Fin de Grado, ETSI, Universidad de Huelva, Huelva.

Wilcox, D. C. (1998). *Turbulence Modeling for CFD* (3ª edición). DCW Industries. Obtenido de https://cfd.spbstu.ru/agarbaruk/doc/2006_Wilcox_Turbulence-modeling-for-CFD.pdf