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

ANÁLISIS COMPARATIVO ENTRE HERRAMIENTAS DE SIMULACIÓN FLUIDODINÁMICA (CFD) PARA LA CARACTERIZACIÓN DE SISTEMAS DE VENTILACIÓN

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MECÁNICO

PABLO ANDRÉS POGORELOW MORALES

PROFESOR GUÍA:
DR. GABRIEL ARÉVALO GONZÁLEZ

MIEMBROS DE LA COMISIÓN:
DR. ÁLVARO VALENCIA MUSALEM
MSC. JORGE MELLA DÍAZ

SANTIAGO DE CHILE
2021

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL MECÁNICO
POR: PABLO ANDRÉS POGORELOW MORALES
FECHA: 2021
PROF. GUÍA: GABRIEL ARÉVALO

ANÁLISIS COMPARATIVO ENTRE HERRAMIENTAS DE SIMULACIÓN FLUIDODINÁMICA (CFD) PARA LA CARACTERIZACIÓN DE SISTEMAS DE VENTILACIÓN

La Dinámica de Fluidos Computacional (CFD) cumple un rol fundamental en la ingeniería moderna, permitiendo simular y estudiar numéricamente una serie de fenómenos. Existe en la actualidad una industria dedicada a los programas computacionales que permiten realizar este tipo de simulaciones, siendo de gran interés estudiar las diferencias existentes entre dos de las herramientas más conocidas en simulación CFD (ANSYS Fluent y OpenFOAM) mediante un análisis aplicado a un sistema de ventilación industrial real. Con este fin, se buscan establecer las principales diferencias metodológicas de ambos programas, realizar una comparación libre de sesgo de mallado y poder establecer rangos de operación equivalentes entre ambas herramientas.

Mediante información otorgada por SAME S.A. se construye un caso de estudio de ventilación simplificado a partir del sistema de ventilación de un laboratorio metalúrgico. Empleando Autodesk Inventor 2019 y ANSYS Meshing 2018 se realiza un mallado de una geometría simplificada desde donde se desarrollan simulaciones en estado transiente equivalentes en OpenFOAM v7 y ANSYS Fluent 2018. Los resultados de las simulaciones resultan ser similares y comparables, pudiendo hacer un pormenorizado análisis de las principales variables simuladas. Se encuentra que OpenFOAM presenta resultados menos conservadores que A. Fluent, teniendo mayores rangos de valores y dispersión de datos. El análisis comparativo de temperaturas evidencia que la inicialización del caso de estudio en estado estacionario determina solamente los primeros segundos de simulación, lográndose luego evoluciones temporales de temperatura similares en ambos programas.

Las mayores diferencias numéricas se encuentran al analizar la dinámica del flujo mediante velocidad. En puntos de medición equivalentes, OpenFOAM llega a obtener una diferencia de 27 % respecto a la simulación realizada en A. Fluent. Un análisis en detalle muestra visualmente un desarrollo de la dinámica interna de flujo similar entre ambas simulaciones finales pero no idéntico.

Finalmente, se analiza el flujo másico simulado presentando leves diferencias entre ambas herramientas y se describe una metodología de uso de OpenFOAM, en cuyo detalle se busca brindar una base de conocimiento para el empleo del programa en esta área de la ingeniería mecánica.

Dedicado a mis abuelos y abuelas,
ejemplos vívidos de que el esfuerzo trae las más grandes recompensas
Sinceramente

Agradecimientos

Me gustaría partir agradeciendo a mi profesor guía Gabriel Arévalo y a SAME S.A. por darme la oportunidad de realizar un tema de memoria que sea un aporte para la comunidad y ojalá no otro empaste jamás abierto, espero este sea un punto de inicio más que uno de término.

Quiero agradecer a mi familia, a mis padres y mis hermanas. Son todo lo que un testarudo y volátil estudiante pudiera desear para sacar adelante cualquier tarea y meta que me proponga. Me han apoyado siempre y los he sentido conmigo en todo este camino.

Me gustaría mencionar a Eduardo Ortega en representación de un gran número de profesores que con sus consejos y enseñanzas me formaron y me guiaron hasta esta Facultad, a mis entrenadores y amigos que conocí a través del deporte, muchos de mis días más felices los pasé con un balón en la mano y gritando en una cancha.

A mis cucarachos y mis amigos de mecánica, amigos y amigas por los cuales haría todo, ustedes lo saben. Sé que soy malo para pedir ayuda pero llegado el momento también se que estarán ahí como han estado en el pasado.

A ti Pau, fuiste mi compañera todos estos años y mi gran amiga. Aprendí tanto contigo como pude y he mantenido mi promesa de seguir haciéndolo, tú y tu familia significarán siempre mucho para mi.

Finalmente a ti, persona que estás leyendo esto y no haces de este un conocimiento vacío y olvidado.

Desde lo más profundo de mi corazón, muchas gracias.

Tabla de Contenido

Índice de Tablas

Índice de Figuras

Capítulo 1

Introducción

La Dinámica de Fluidos Computacional (CFD por sus siglas en inglés), es el análisis de sistemas que involucran flujo de fluidos, transferencia de calor y fenómenos asociados, tales como reacciones químicas mediante simulaciones realizadas en computadores. Dentro del amplio abanico de aplicaciones CFD se listan por ejemplo investigaciones científicas, herramientas educacionales, aplicaciones aeroespaciales, desarrollo vehicular, generación de energía y un sinnúmero de aplicaciones ingenieriles (Tu, 2019). Dentro de estas últimas, se encuentra el diseño de sistemas de ventilación natural y forzada.

El constante desarrollo de la sociedad y sus necesidades demanda una permanente mejora en cuanto a la calidad del aire, confort térmico y eficiencia energética de los sistemas de ventilación (Jiang, Allocca, y Chen, 2004). En cuanto a la calidad del aire, es el método de ventilación mecánica aquel principalmente utilizado para lograr niveles que cumplan con las normas establecidas (Posner, Buchanan, y Dunn-Rankin, 2003). El empleo de la mecánica de fluidos computacional permite caracterizar y analizar complejos patrones de flujo presentes en el desarrollo de esta especialidad.

La validez de los resultados generados por CFD ha sido un problema constante. Sin embargo, el desarrollo computacional y de métodos numéricos mejoran la precisión de las predicciones y muestran un potencial enorme para el análisis de flujos complejos de aire. Esto permite que gran parte del diseño de sistemas de ventilación contemple una etapa de simulación computacional (Hong et al., 2017).

Existe una amplia gama de programas empleados en el desarrollo de esta disciplina, una fácil distinción entre ellos es el carácter privado de una gran cantidad de ellos. El programa Fluent es quizás el software comercial CFD más utilizado en la industria local. Adquirido el año 2006 por ANSYS Inc., ha permitido que este programa contenga un considerable respaldo y desarrollo. Por otra parte, OpenFOAM es un programa de código libre cuya propiedad intelectual si bien pertenece a OpenFOAM Foundation, presenta una licencia GLPv3. Esta licencia implica que el

usuario puede usar, crear y compartir su trabajo realizado sin restricciones. Su estructura se basa en el lenguaje de programación $C++$ y ha estado disponible de forma gratuita desde el año 2004.

El principal objetivo de este trabajo, es estudiar las diferencias entre herramientas de simulación CFD para aplicaciones en sistemas de ventilación industrial. Para ello, se contempla la utilización de dos programas especializados, ANSYS Fluent 2018 y OpenFOAM v7 (2019). Para este fin se resumen los siguientes objetivos específicos:

- Realizar un estudio acerca de los principales componentes, características y diferencias de los programas Ansys Fluent 2018 y OpenFOAMv7.
- Obtener una representación computacional fidedigna mediante CFD de un caso de estudio real de ventilación industrial.
- Analizar de forma cuantitativa simulaciones equivalentes realizadas al caso de estudio para cada software.
- Consolidar el contenido generado en una metodología de uso de OpenFOAM que permita replicaciones similares en el futuro.

Ya que el universo de posibilidades de simulación es muy amplio, se restringen los casos de estudio a un campo de trabajo permanente en los proyectos de ventilación industrial. Se consideran situaciones de análisis de flujos de aire y transferencia de calor al medio desde una fuente con temperatura constante.

Para la formulación del caso de estudio se utilizan planos reales proporcionados por la empresa SAME S.A. A partir de ellos se realiza una simplificación geométrica que dé cuenta de los principales fenómenos de ventilación. De la misma forma se extrae la información de operación utilizada en determinar las condiciones de borde del problema. El volumen de control es construido mediante el software Autodesk Inventor 2019 y se realiza el mallado de dicha geometría utilizando ANSYS Meshing 2018. La realización de simulaciones posteriores se desarrollan utilizando ANSYS Fluent 2018 y OpenFOAM v7 (2019). Finalmente, se realiza el post procesamiento de datos utilizando ParaView 5.4, ANSYS Fluent y Microsoft Excel. Todo lo anterior se realiza en un total de 4 computadores utilizando sistemas operativos de Microsoft Windows y Linux.

Capítulo 2

Antecedentes generales

2.1. Mecánica de fluidos computacional - Teoría

La Dinámica de Fluidos Computacional representa un área temática de suma importancia en la mecánica de fluidos avanzada, donde se debe tener mucho cuidado en el proceso de obtención de soluciones numéricas (Munson, Young, y Okiishi, 2002). La rigurosidad del proceso es producto de la validación final de este, donde los datos obtenidos son contrastados con datos experimentales, siendo generalmente aceptado que modelos de turbulencia requieren de estudios de validación y verificación para evaluar el desempeño de dicho modelo en cada caso particular (Blocken, 2018). El proceso de simulación consta de numerosos pasos, donde se parte por determinar la geometría y el modelo físico-matemático a resolver. A continuación, se presentan los principales aspectos relacionados con la caracterización física y aproximación computacional de un problema CFD.

2.1.1. Ecuaciones gobernantes

Las ecuaciones de Navier Stokes son aquellas que rigen el movimiento de un fluido en el espacio tridimensional. Provienen de aplicar la segunda ley de Newton a una porción infinitesimal de dicho fluido (Munson et al., 2002). Como resultado de esto se tienen las ecuaciones 2.1 y 2.2. La primera expresión corresponde a la condición de continuidad del fluido, que como se verá más adelante, actuará como restricción del modelo matemático a resolver.

En la segunda expresión se presentan una serie de términos. Al lado izquierdo de la igualdad se tiene el equivalente a la masa por la aceleración de una porción del fluido, donde el análisis es realizado sobre el supuesto de un fluido cuyo volumen permanece constante. Al lado derecho de la ecuación se tienen las fuerzas internas expresadas por el gradiente de presión, la componente de viscosidad o fricción y luego los términos que resumen las fuerzas de cuerpo y externas aplicadas al fluido.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \tag{2.1}$$

$$\frac{\partial}{\partial t}(\rho\vec{v}) + \nabla \cdot (\rho\vec{v}\vec{v}) = -\nabla p + \nabla \cdot (\bar{\bar{\tau}}) + \rho\vec{g} + \vec{F} \tag{2.2}$$

Donde p es la presión estática y $\bar{\bar{\tau}}$ el tensor de esfuerzos descrito por 2.3. Aquí, μ es la viscosidad dinámica e I el tensor unitario.

$$\bar{\bar{\tau}} = \mu\left[\left(\nabla\vec{v} + \nabla\vec{v}^T\right) - \frac{2}{3}\nabla \cdot \vec{v}I\right] \tag{2.3}$$

Con el fin de caracterizar por completo un problema, muchas veces se requieren ecuaciones adicionales capaces de dar cuenta de otros fenómenos. La ecuación de energía 2.4 da cuenta del intercambio de calor en el medio de trabajo. Los términos de la derecha de esta expresión representan: la transferencia de energía mediante conducción (donde k_{eff} será la conductividad efectiva producto de la adición entre la conductividad del medio y el término de conductividad turbulento definido por el modelo de turbulencia empleado); el término difusivo de transporte de especies con h_j la entalpía y J_j el flujo de la especie j y la disipación viscosa. Además, se define S_h como calor externo producto de una fuente volumétrica o reacción química.

$$\frac{\partial}{\partial t}(\rho E) + \nabla \cdot (\vec{v}(\rho E + p)) = \nabla \cdot \left(k_{eff}\nabla T - \sum_j h_j\vec{J_j} + (\bar{\tau}_{eff} \cdot \vec{v})\right) + S_h \tag{2.4}$$

A la izquierda de la igualdad se plantea la variación interna de energía en el volumen de control. Aquí, E está dada por la ecuación 2.5, donde h corresponde a la entalpía sensible cuyo cálculo variará dependiendo de las condiciones del fluido de trabajo.

$$E = h - \frac{p}{\rho} + \frac{v^2}{2} \tag{2.5}$$

En ocasiones donde no existen grandes variaciones de temperatura en el fluido la aproximación de Boussinesq es una simplificación recurrente. Sostiene que la variación en densidad de un fluido se asume constante para los términos temporales y convectivos de la ecuación 2.2, y que la dependencia del fluido con la densidad solo considera variaciones asociadas a la fuerza de empuje en problemas donde está presente el fenómeno de convección natural. Esta aseveración equivale a no sobrepasar $Ma < 0.3$, con Ma el número de Mach (Zhang, Liang, Ren, Wang, y Wang, 2016; Joel H. Ferziger, 2001).

Para caracterizar esta aproximación se emplea la relación 2.6, en donde ρ_0 es la densidad de referencia del fluido a una temperatura T_0 y β es el coeficiente de expansión termal. En orden de obtener errores de cálculo menores al 1 % se debe cumplir la condición 2.7, que en términos prácticos pone una condición sobre la temperatura T al ser reemplazada en la ecuación 2.6 dando origen a la expresión 2.8. Si se emplea aire como fluido de trabajo, la condición de temperatura limita a una variación máxima de 15° la validez de la aproximación de Boussinesq (Joel H. Ferziger, 2001; Wimshurst, 2018).

$$\rho = \rho_0 \left[1 - \beta \left(T - T_0\right)\right] \tag{2.6}$$

$$\frac{\rho - \rho_0}{\rho} << 1 \tag{2.7}$$

$$\beta \left[T - T_0\right] << 1 \tag{2.8}$$

Las principales ventajas de la implementación de la aproximación de Boussinesq se asocian a la simplificación del sistema de ecuaciones a solucionar. Obviando el cálculo de la variable de presión ρ se logran mayores estabilidades en la resolución numérica y un menor gasto computacional asociado al almacenamiento de información y tiempo de cómputo.

2.1.2. Método de volúmenes de control

La resolución numérica del sistema de ecuaciones gobernantes mostradas anteriormente, requiere de un proceso de discretización acorde al conjunto de ecuaciones que se intenta resolver. En este sentido, no cualquier método de diferencias finitas logra el cometido. Dado que la discretización de los términos convectivos implican un valor de transporte hacia las caras del volumen de control, junto con un balance de flujos convectivos, es que se debe emplear un método que logre caracterizar dicho fenómeno. El método de los volúmenes de control, divide el espacio en celdas que cuentan con caras colindantes con sus vecinas y nodos centrales. En la figura 2.1 presenta una simplificación para el caso de una dimensión.

Luego de realizada la discretización del espacio, el método de cálculo debe ser determinado. Se definen como “esquemas” las metodologías empleadas para realizar el cálculo de los valores en cada nodo. Se destacan: Diferencias centrales, aguas arriba (upwind), exponencial, híbrido y ley de potencia.

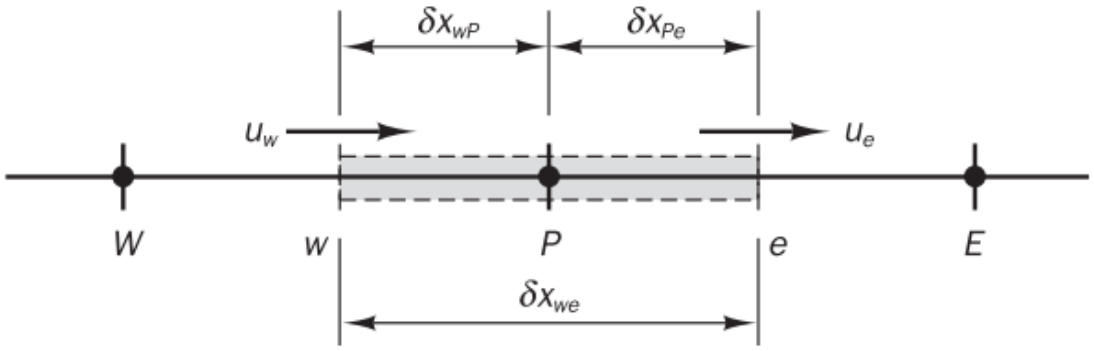


Figura 2.1: Volumen de control entorno al punto P para 1D (Versteeg y Malalasekera, 2016)

Para este estudio se contempla principalmente el uso del esquema aguas arriba y para su entendimiento es necesario observar la figura 2.2. Aquí, el esquema contempla la dirección del flujo (de izquierda a derecha) al momento de determinar el cálculo de los puntos en torno a P. Así, si el fluido se mantiene en la dirección "a favor del viento"los valores de la cara de la celda descrita son determinados por los puntos a la izquierda ($\phi_w = \phi_W$ y $\phi_e = \phi_p$), sin considerar los valores a la derecha.

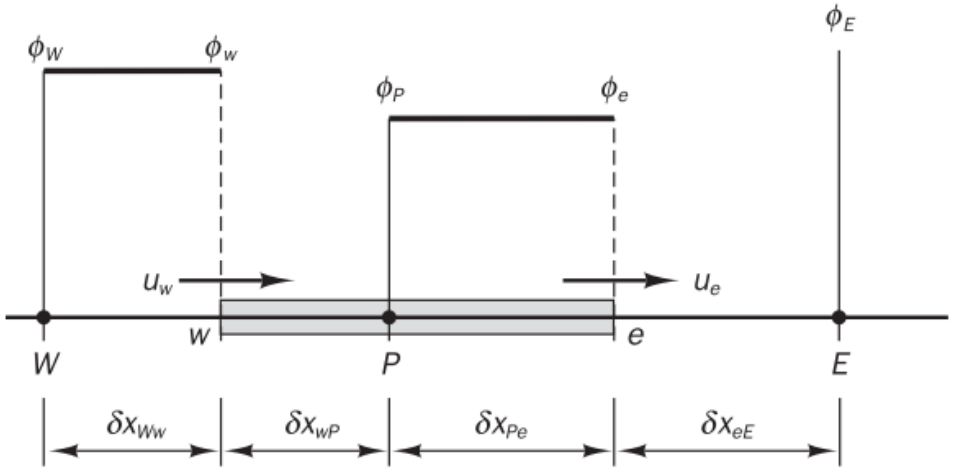


Figura 2.2: Volumen de control entorno al punto P esquema upwind (Versteeg y Malalasekera, 2016)

La principal razón del empleo de este tipo de esquemas en este trabajo, es la modelación del fenómeno de advección y transporte de variables, el cual intrínsecamente se desarrolla en la dirección del flujo (Versteeg y Malalasekera, 2016). A pesar de tener presente este aspecto, dependiendo del esquema empleado en la discretización existen otros factores a tener en consideración. A continuación se presentan cuatro aspectos a tener en cuenta en la elección de un esquema de discretización:

1. Conservación: Tendrá relación con la capacidad del esquema de evaluar la condición de continuidad de flujo a través de un número finito de volúmenes de control. Para asegurar la conservación de una propiedad ϕ para todo el dominio, el flujo de ϕ que sale de una cara de una celda, debe ser idéntico al flujo de la misma propiedad que entra a través de las caras adyacentes de la celda en cuestión.

   Para asegurar esta condición, cada esquema define una forma constante de calcular el balance de flujos a través de cada celda. Las inconsistencias en la interpolación del cálculo del flujo, conllevan a que ciertos esquemas no cumplan de forma global la condición de conservación.

2. Delimitación/acotación: Desde un punto de vista numérico, existen criterios sobre los valores de los coeficientes de las ecuaciones discretizadas, mediante los cuales se asegura de forma suficiente la convergencia de la solución[1]. Garantizando dicha condición, la solución satisface la acotación del problema.

   Desde el punto de vista físico, que los valores de una propiedad cualquiera ϕ estén acotados, significa que no se encontrarán valores fuera de los máximos y mínimos impuestos por las condiciones de borde de la variable. En la aplicación de esta condición se encuentra por ejemplo la imposibilidad de exceder un valor de temperatura en el medio por sobre el de una fuente de calor. Llevándolo a una discretización 1D como la expuesta en 2.2, el valor de ϕ_P estará dentro del intervalo $[\phi_W, \phi_E]$ sin importar el método de interpolación empleado para calcular ϕ_P.

3. Transportividad: Tendrá relación con la capacidad del fluido de transportar a través de la advección los fenómenos presentes en este. El numero de Peclet Pe (definido mediante la ecuación 2.9) dirá relación entre los efectos de advección (F) y difusión (D) presentes en el fluido. Con un valor de Pe alto ($u \to \infty$), el transporte mediante advección será predominante sobre el efecto de la difusión. Dependiendo de la condición del fluido a modelar, el esquema de discretización debe ser capaz de captar este fenómeno.

$$Pe = \frac{F}{D} = \frac{\rho u}{\Gamma/\delta x} < 2 \tag{2.9}$$

   Esta formulación se realiza sobre la discretización mediante el método de diferencias centrales. Aquí Γ corresponde al coeficiente de difusividad y δx el largo característico de la discretización. Se impone en este caso que el número de Peclet permanezca estrictamente menor a 2 para asegurar la positividad en los coeficientes de la discretización.

4. Precisión: Dependiendo del esquema de discretización numérico empleado, es posible lograr una aproximación numérica que, siendo cercana al valor buscado, no llegue a este. En la aplicación del método de volúmenes de control, la precisión estará asociada con la cantidad de

[1] Específicamente el criterio de Scarborough. Ver: Patankar, 1980, p.64 y Versteeg y Malalasekera, 2016, p.143

términos matemáticos empleados al realizar la ponderación numérica (Versteeg y Malalasekera, 2016).

Si se observa la composición del número de Peclet en la ecuación 2.9 se aprecia que tanto Γ como ρ son propiedades del fluido, u condición de flujo y δx parámetro del mallado. En este sentido, esquemas como el de diferencias centrales se ven limitados en la capacidad de modelar flujos de carácter advectivo (alto Pe) dada la naturaleza del cálculo (condición 2.9), por lo que la elección del método de diferenciación debe estar asociado a la naturaleza del fenómeno y a la precisión numérica deseada.

El conjunto de aspectos anteriores junto con la matemática empleada en calcular los valores de las variables según las definiciones de cada esquema, determinan las principales características y su grado. Aquellos de mayor orden o grado requerirán de un mayor gasto computacional en pos de alcanzar mayores precisiones (Ansys, 2013; Ramponi y Blocken, 2012).

Otro aspecto a considerar relacionado con la discretización e implementación del método de cálculo será el número de Courant. El número de Courant o relación Courant-Friedrichs-Lewy, es la relación entre el intervalo de tiempo Δt y la escala de tiempo de convección $u/\Delta x$. Siguiendo lo mostrado por las ecuaciones 2.10, 2.11 y 2.12 se aprecia que el promedio de flujos determinados entre las celdas E y W (siguiendo la figura 2.2) definen la misma escala advectiva, un importante factor de estabilidad numérica (ANSYS, 2018).

$$Co = \frac{u\Delta t}{\Delta x} \tag{2.10}$$

$$Co = \frac{u\Delta t}{\Delta x} = \frac{u\Delta t}{\Delta x}\frac{A}{A} = \frac{\phi \Delta t}{\Delta V} \tag{2.11}$$

$$Co = \frac{\frac{|\phi_E| - |\phi_W|}{2}\Delta t}{\Delta V} = \frac{1}{2}\frac{\left(|\phi_E| - |\phi_W|\right)\Delta t}{\Delta V} \tag{2.12}$$

Para asegurar precisión temporal y estabilidad numérica el número de Courant debe ser menor a la unidad para ciertos solver como icoFoam (Greenshields, 2014) (descripción de icoFoam en C.6). En relación a programas comerciales se pueden encontrar por defecto valores de relajación de esta condición. Un incremento del número de Courant es plausible en aras de disminuir el tiempo de cálculo, sin embargo números excesivamente altos pueden tender a inestabilidades indeseadas (Tu, 2019).

Para problemas sencillos de fluidos compresibles ANSYS Fluent utiliza un valor por defecto de 5 y sugiere no exceder valores de 10, 20 y 100 para la condición de CFL. Se aconseja iniciar con

Courant bajo e ir modificando el paso tiempo progresivamente acorde a la estabilidad numérica de la solución. Manifestada esta sugerencia es el usuario quien debe implementar este parámetro bajo la advertencia de posibles inestabilidades numéricas en la relajación de este parámetro, aconsejándose la reducción del paso de tiempo en dichos casos.

2.1.3. Aproximación computacional

La solución numérica de una ecuación diferencial consiste en un conjunto de números a partir del cual se puede construir la distribución de una variable dependiente (Patankar, 1980). En los problemas de la mecánica de fluidos computacional se emplea la discretización de las ecuaciones gobernantes para obtener valores de aquellas variables de interés en todo el dominio deseado.

La turbulencia es un fenómeno presente en la mecánica de fluidos que genera vórtices en un amplio rango de escalas de longitud y tiempo que interactúan de forma dinámicamente compleja (Versteeg y Malalasekera, 2016). Dada la importancia de este fenómeno en las aplicaciones de la mecánica de fluidos, la discretización de las ecuaciones gobernantes concentran gran esfuerzo en capturar este fenómeno.

Existen tres categorías de clasificación para los métodos de discretización actualmente utilizados: Modelos de turbulencia para *Reynolds-averaged Navier Stokes* abreviado como RANS, *Large Eddy Simulation* abreviado como LES y *Direct Numerical Simulation* o DNS. El último método es el único capaz de calcular todas las fluctuaciones turbulentas resolviendo las ecuaciones de Navier Stokes en términos de espacios de escala y tiempo. Como en las aplicaciones ingenieriles se hace uso de un número promedio o acotado de parámetros, el costo computacional asociado a este método no justifica su uso práctico, siendo descartado en la industria. A continuación se exponen brevemente los métodos RANS y LES, cuyas formulaciones se tienen en consideración para la realización de este trabajo.

- RANS: Esta metodología consiste en discretizar las ecuaciones gobernantes en términos de componentes promedio y componentes fluctuantes. De esta forma se obtiene un nuevo set de ecuaciones que resuelven el comportamiento general del fluido, mientras que los términos asociados a la turbulencia se obtienen de forma aproximada.

  Las aproximaciones realizadas sobre los términos turbulentos generan a su vez variables desconocidas que serán aproximadas nuevamente por el modelo de turbulencia empleado. Dentro de los modelos más conocidos se encuentra $k-\epsilon$, $RNG\ k-\epsilon$, $k-\omega$, $SST\ k-\omega$, RSM, entre otras variaciones y modelos. Para aplicaciones de ventilación, el análisis bibliográfico ha mostrado que comúnmente se emplean los tres o cuatro primeros modelos, cuyos resultados han sido ampliamente validados por la literatura y la experimentación. Ejemplo de esto se

encuentra el trabajo de (Tian, Tu, Yeoh, y Yuen, 2006), quien contrasta los modelos $k-\epsilon$, $RNGk-\epsilon$ y LES con datos experimentales.

- LES: Esta metodología tiene la particularidad de realizar una diferenciación en la escala de turbulencia, diferenciando aquella mayormente generada por la geometría del problema (vórtices de mayor tamaño) y la asociada a pequeños vórtices incapaces de resolver dada la resolución de la malla empleada. Esta diferenciación es realizada mediante una función de filtrado, donde las escalas más pequeñas que la distancia de dicho filtro son removidas de las variables a resolver por el sistema de ecuaciones (Blocken, 2018).

  Se tiene entonces un tamaño de malla que permite resolver la mayor parte del fenómeno turbulento deseado y otra turbulencia de menor escala que es estimada. Esta última tiende a estar presente a lo largo de todo el dominio, por lo que su estimación en el cálculo no perjudica mayormente los resultados (Blocken, 2018). Se resuelven entonces las ecuaciones de Navier Stokes para los vórtices de mayor tamaño (y energía), en conjunto con las propiedades del fluido en general así como en RANS.

  Si bien este método no requiere un nivel de detalle en mallado como el necesario para DNS, si requiere un mallado considerablemente más fino que los métodos RANS, así como un mayor tiempo de procesamiento para los casos que requieren pequeños pasos de tiempo (Tu, 2019). Además del gasto computacional que esto significa, LES presenta una complejidad asociada a la alta resolución de mallado cercano a las capas de condiciones de borde, puesto que los remolinos 'grandes' en este espacio se vuelven pequeños, requiriendo una resolución asociada al número de Reynolds. Es por esto que LES se ve limitado a fluidos con números de Reynolds bajos, del orden de 10^4 y 10^5 (Ansys, 2013).

2.1.4. RANS: k-epsilon

Las modelación de las ecuaciones de momentum según RANS son una versión aproximada en el tiempo, descartando el detalle de las fluctuaciones instantáneas del fluido. La descomposición de Reynolds consiste en modelar la presión y la velocidad de forma de tener un valor promedio y un valor fluctuante en el tiempo de la forma de las ecuaciones 2.22 y 2.14 (Tu, 2019; Fridolin, 2012). Estas definiciones se expanden también a la variable T, donde los términos t' serán aquellos valores fluctuantes.

$$u = U + u' \tag{2.13}$$

$$p = P + p' \tag{2.14}$$

$$t = T + t' \tag{2.15}$$

Al ingresar la descomposición de Reynolds en la ecuación 2.2 se obtiene una serie de términos adicionales $-\rho\overline{u_i' u_j'}$ definidos como esfuerzos de Reynolds. Ya que el número de ecuaciones gobernantes no ha sido aumentado, es necesaria la implementación de otros métodos para poder resolver estas nuevas incógnitas.

El sistema es capaz de resolverse al relacionar estos nuevos términos con las propiedades del flujo de forma promedio. La ecuación 2.16 resume los valores del tensor de esfuerzos de Reynolds, proposición realizada por Boussinesq el año 1869. Aquí el término δ_{ij} es el delta de Kronecker y k la energía cinética turbulenta descrita por la ecuación 2.17 equivalente a la expresión 2.18.

$$-\rho\overline{u_i' u_j'} = \mu_t \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij} \right) - \frac{2}{3}k\delta_{ij} \tag{2.16}$$

$$k = \frac{1}{2}\left(u'^2 + v'^2 + w'^2 \right) \tag{2.17}$$

Existen varias formas de obtener el valor de k y de los otros parámetros turbulentos, cada implementación dará lugar a una variación distinta de modelo de turbulencia con aproximaciones de una y dos ecuaciones además de formulaciones algebraicas. Para este estudio se emplea la formulación de dos ecuaciones $k - \epsilon$ descrita por Launder y Spalding el año 1974.

Se define la energía cinética turbulenta k según la ecuación 2.18 y la tasa de disipación turbulenta ε como la ecuación 2.19, con $i, j = 1, 2, 3$. Con estas expresiones de k y ε se define el parámetro de viscosidad turbulenta μ_T definido según la ecuación 2.20. La viscosidad cinemática se puede expresar de la forma $\nu_T = \mu_T/\rho$.

$$k = \frac{1}{2}\left(u_i' u_i' \right) \tag{2.18}$$

$$\varepsilon = \nu_T \overline{\left(\frac{\partial u_i'}{\partial x_j} \right)\left(\frac{\partial u_i'}{\partial x_j} \right)} \tag{2.19}$$

$$\mu_T = \frac{C_\mu \rho k^2}{\varepsilon} \tag{2.20}$$

Empleando estas definiciones, se muestran a continuación las ecuaciones de continuidad y momento en dos dimensiones bajo el modelo $k - \varepsilon$.

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{2.21}$$

$$\frac{\partial u}{\partial t}+\frac{\partial (uu)}{\partial x}+\frac{\partial (vu)}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\frac{\partial}{\partial x}\left[(v+v_T)\frac{\partial u}{\partial x}\right] +\frac{\partial}{\partial y}\left[(v+v_T)\frac{\partial v}{\partial y}\right]+\frac{\partial}{\partial x}\left[(v+v_T)\frac{\partial u}{\partial y}\right]+\frac{\partial}{\partial y}\left[(v+v_T)\frac{\partial v}{\partial x}\right] \tag{2.22}$$

$$\frac{\partial v}{\partial t}+\frac{\partial (uv)}{\partial x}+\frac{\partial (vv)}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial y}+\frac{\partial}{\partial x}\left[(v+v_T)\frac{\partial v}{\partial x}\right] +\frac{\partial}{\partial y}\left[(v+v_T)\frac{\partial v}{\partial y}\right]+\frac{\partial}{\partial x}\left[(v+v_T)\frac{\partial u}{\partial y}\right]+\frac{\partial}{\partial y}\left[(v+v_T)\frac{\partial v}{\partial y}\right] \tag{2.23}$$

Con $Pr_T=\dfrac{\mu_T}{\Gamma_T}$, se tiene la ecuación de transporte de temperatura 2.24

$$\frac{\partial T}{\partial t}+\frac{\partial (uT)}{\partial x}+\frac{\partial (vT)}{\partial y}=\frac{\partial}{\partial x}\left[\left(\frac{\nu}{Pr}\frac{\nu_T}{Pr_T}\right)\frac{\partial T}{\partial x}\right]+\frac{\partial}{\partial y}\left[\left(\frac{\nu}{Pr}\frac{\nu_T}{Pr_T}\right)\frac{\partial T}{\partial y}\right] \tag{2.24}$$

Para los términos de energía cinética turbulenta k y la tasa de disipación turbulenta ε se tienen las ecuaciones 2.26 y 2.27.

$$\frac{\partial k}{\partial t}+u\frac{\partial k}{\partial x}+v\frac{\partial k}{\partial y}=\frac{\partial}{\partial x}\left(\frac{v_T}{\sigma_k}\frac{\partial k}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{v_T}{\sigma_k}\frac{\partial k}{\partial y}\right)+P-D \tag{2.25}$$

$$\frac{\partial \varepsilon}{\partial t}+u\frac{\partial \varepsilon}{\partial x}+v\frac{\partial \varepsilon}{\partial y}=\frac{\partial}{\partial x}\left(\frac{v_T}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{v_T}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial y}\right)+\frac{\varepsilon}{k}\left(C_{\varepsilon 1}P-C_{\varepsilon 2}D\right) \tag{2.26}$$

$$P=2v_T\left[\left(\frac{\partial u}{\partial x}\right)^2+\left(\frac{\partial v}{\partial y}\right)^2\right]+v_T\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)^2 \tag{2.27}$$

Este modelo emplea cinco constantes ajustables descritas en la tabla 2.1. Estas constantes son obtenidas mediante el ajuste del modelo a una amplia serie de experimentaciones realizadas por Launder y Spalding para problemas de flujo turbulento (Tu, 2019).

Tabla 2.1: Constantes modelo k-epsilon

Constante	C_μ	$C_{\varepsilon 1}$	$C_{\varepsilon 2}$	σ_k	σ_ε
Valor	0.09	1.44	1.92	1.0	1.3

2.1.5. Solvers

Un solver es un programa o un algoritmo de resolución que resuelve un set de ecuaciones generadas producto de un proceso de discretización. En cuanto a los solvers CFD, existe una particularidad en la resolución, dado que el conjunto de ecuaciones gobernantes suele ser un sistema de ecuaciones no lineales. El método de resolución entonces es realizado de forma iterativa, donde el algoritmo es aplicado la cantidad de veces requerida para llegar a un nivel de convergencia estipulado por el usuario, sin la necesidad de una solución exacta (Moukalled, Mangani, y Darwish, 2016).

Los programas de simulación CFD presentan una gran amplia variedad de solvers para la elección del usuario. En ANSYS Fluent la elección se centra en cuatro métodos de resolución principales (SIMPLE, SIMPLEC, PISO y COUPLED), presentando leves variaciones dependiendo del modelo a resolver. Por otra parte, OpenFOAM presenta cerca de un centenar de opciones dependiendo del problema a modelar por el usuario. Si bien se presentan los mismos métodos que en ANSYS, cada solver tendrá un alto grado de especificidad, como podrá apreciarse más adelante al momento de abordar la resolución de la ecuación de energía y el fenómeno de transferencia de calor.

Un aspecto transversal al proceso de modelación es la dependencia del tiempo de un modelo. Esta, dirá relación con formulaciones estacionarias o transientes en el aspecto de resolución numérica. En el caso del estado transiente, varias simulaciones son realizadas por un paso de tiempo, no avanzando hasta el siguiente de no cumplir las condiciones de convergencia impuestas (Versteeg y Malalasekera, 2016). Es común encontrar diversas variaciones de métodos de resolución estacionarios para estados transientes como se verá en esta sección.

En las ecuaciones de momento el fluido se desplaza mediante diferencias de presión denotadas por el primer término del lado derecho de la ecuación 2.2. Si bien adicionando la ecuación de continuidad (ecuación 2.1) se tiene un sistema de cuatro ecuaciones y cuatro incógnitas, al no existir una ecuación de transporte independiente para la presión, se tiene un sistema de ecuaciones acoplado. La dificultad intrínseca de resolución de este sistema, se debe a que la ecuación de continuidad corresponde a una restricción sobre el campo de velocidades generado a partir de las ecuaciones de momento, siendo la solución de estas obligadas a satisfacer la continuidad del fluido y no de otra forma.

A continuación se presentan de manera resumida los principales métodos de resolución empleados por los programas CFD para caracterizar fluidos.

- SIMPLE: Este algoritmo creado el año 1972 por Patankar y Spalding, es uno de los algoritmos

más utilizados en la resolución del acomplamiento de presión-velocidad en fluidos incompresibles en estado estacionario. Este esquema se crea con la finalidad de obtener soluciones prácticas para la ingeniería (Tu, 2019). Este método de resolución se basa en un proceso iterativo donde se comienza por *estimar* un valor para el campo de presiones. Luego, se procede con una discretización de las ecuaciones de momento y la resolución de estas empleando el campo de valores estimado. Por último, se extrae un factor de corrección para el campo de presiones inicial a raíz de correcciones sobre el campo de velocidades. En este procedimiento se realizan simplificaciones sobre la suma de ciertos términos en la derivación de las correcciones de velocidad, siendo la principal característica del método (Versteeg y Malalasekera, 2016).

A partir de este algoritmo se crean un sinnúmero de variantes al realizar diversas suposiciones sobre los factores de corrección. Es importante recordar que al ser este un procedimiento iterativo de corrección, no existe razón por la cual la fórmula de obtener los factores de corrección, deban ser correctos desde un punto de vista físico (Tu, 2019).

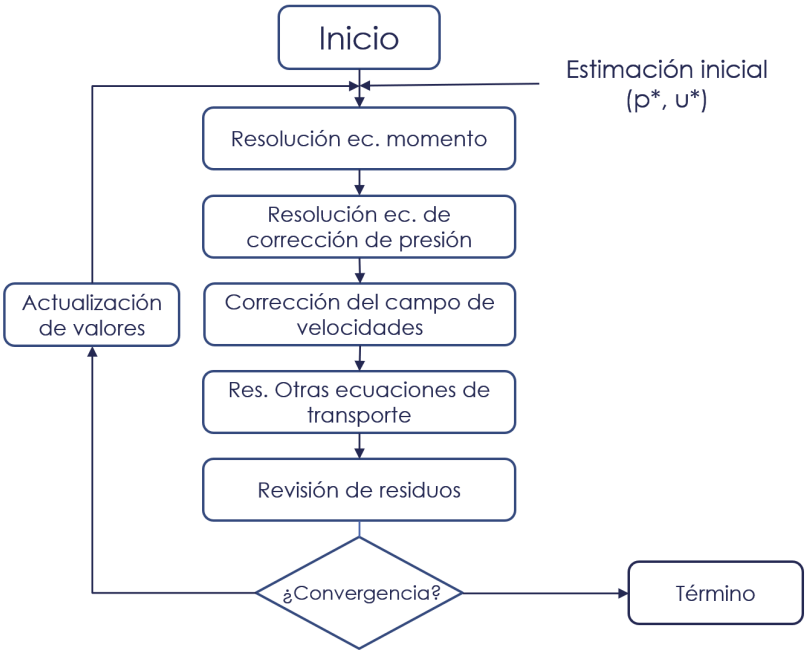


Figura 2.3: Esquema SIMPLE

- SIMPLEC: Esta formulación basada en el algoritmo SIMPLE, es realizada de tal forma que en la manipulación de las ecuaciones de momento, se omiten términos de menor importancia en las correcciones de la velocidad (Versteeg y Malalasekera, 2016). Este algoritmo no requiere de consideraciones adicionales sobre la presión (relajaciones) y mantiene el mismo procedimiento que su algoritmo base. Al emplear este algoritmo se obtienen típicamente resultados más robustos y en un tiempo de convergencia menor (OpenCFD, 2019).

- PISO: Presión implícita con operadores divididos es el significado del acrónimo PISO por sus siglas en inglés. Este algoritmo propuesto por (ISSA, 1986), además de ser desarrollado para el estado transiente, contiene un paso extra de corrección en comparación con SIMPLE (Welahettige y Vaagsaether, 2018). En este caso, la discretización de la ecuación de presión previo a la ecuación de momento provee de un campo de presiones sobre el cual es calculada la cinética del fluido. El corregir dos veces los valores del sistema de ecuaciones acoplado significa un mayor gasto computacional. Sin embargo, empleando factores de relajación, el algoritmo puede llegar a ser el doble de rápido en comparación al SIMPLE para un mismo problema, esto debido a que PISO no estipula el cálculo de sucesivas iteraciones sobre un mismo paso de tiempo como lo realiza SIMPLE (Versteeg y Malalasekera, 2016).

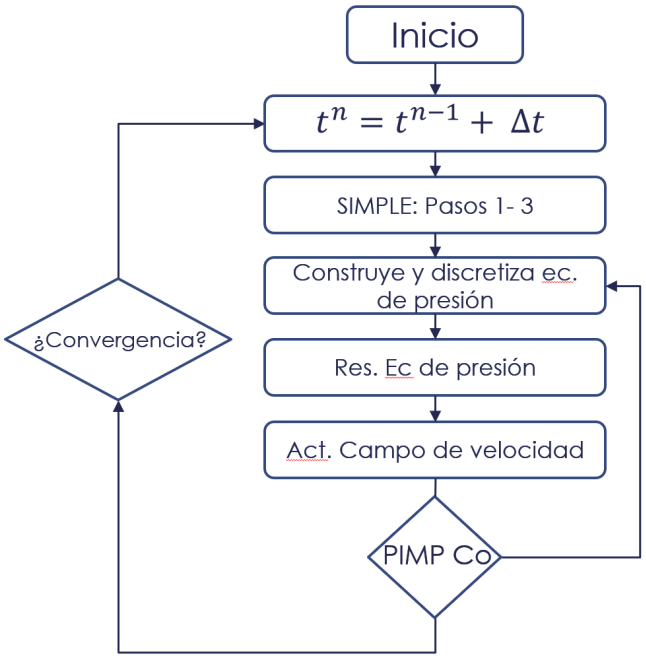


Figura 2.4: Esquema PIMPLE

- PIMPLE: Este algoritmo nace a partir de la mezcla entre los algoritmos PISO Y SIMPLE. Si lo que se define como los correctores no-ortogonales[2] son mayores a la unidad, entonces opera bajo el algoritmo PISO (Welahettige y Vaagsaether, 2018).

  Este algoritmo es ampliamente utilizado en solvers de OpenFOAM, presentando ventajas en comparación a PISO ante en su capacidad de operar en presencia de mallas móviles y cambios de topología en el mallado (Greenshields, 2014).

[2] Un corrector no-ortogonal especifica la cantidad de repeticiones sobre la resolución de las ecuaciones de presión usadas para obtener el factor de corrección no-ortogonal (Greenshields, 2014)

Tanto ANSYS Fluent como OpenFOAM permiten el cálculo de la ecuación de energía (2.4) para problemas que involucren transferencia de calor. Si bien en ambos programas la implementación de las condiciones de borde necesarias para la resolución de este tipo de problemas es sencilla, las modificaciones sobre las configuraciones de los algoritmos distan bastante. A continuación se desglosa la implementación del cálculo de transferencia de calor para ambos software.

En ANSYS Fluent la activación de la resolución de la ecuación de energía recae el habilitar directamente dicha opción dentro de las opciones presentes para el usuario. Este procedimiento dejará disponible la selección del esquema de resolución de la ecuación de energía en esta herramienta. A diferencia de OpenFOAM, la selección del solver se mantiene constante sin importar el tipo de transferencia de calor (radiación, convección y conducción) o cantidad de superficies involucradas en el caso.

OpenFOAM requiere considerables modificaciones en el caso de trabajo para implementar la resolución de la ecuación de energía. Primero se determina el solver a utilizar según si la condición del flujo es transiente o estacionaria. Luego, se analiza el tipo de transferencia de calor presente en el caso y la cantidad de elementos que transfieren calor. Si por ejemplo se desea estudiar el enfriamiento de un elemento en un medio, deben considerar todas las propiedades físicas de aquel elemento en el caso. A continuación se enuncian los tres tipos de solver más comunes para efectos de transferencia de calor[3].

- Buoyant Boussinesq SIMPLE/PIMPLE Foam: Estos métodos emplean la aproximación de Boussinesq descrita en la ecuación 2.6. Serán idóneos para fluidos turbulentos en estado estacionario (SIMPLE) y transiente (PIMPLE) y bajo la suposición de incompresibilidad. Se sugiere el uso de este método ante inestabilidades en el desarrollo de la simulación empleando métodos más complejos.
- Buoyant SIMPLE/PIMPLE Foam: Es por defecto el método a utilizar para ventilación ya que tanto para fluidos transientes como estacionarios se asume compresibilidad en el fluido, incluyendo los efectos de convección generados por las diferencias de temperatura y densidad. La implementación del método opera para un mismo fluido, cuyas propiedades termo-físicas deben ser entregadas como información en el modelo al igual que en el método anterior.
- chtMultiRegion Foam: Es el mecanismo de resolución más capaz para la modelación de transferencia de calor sencilla en OpenFOAM. Permite la interacción de diversos medios y fases entre sólidos y fluidos. Opera bajo la lógica de subdividir el cálculo de la dinámica de cada fluido, teniendo que el usuario especificar parámetros de simulación sobre cada elemento distinto que componga el caso.

[3] Esta selección puede ampliarse a fluidos con más de una fase presente en el caso o a resoluciones basadas en la densidad y no en la presión.

2.1.6. Funciones de pared

Los modelos de turbulencia son ampliamente usados en las simulaciones CFD, sin embargo no todos los modelos son válidos en todo el dominio. Sin ir más lejos, el modelo $K - \varepsilon$ es solo válido en el volumen donde la turbulencia está completamente desarrollada y no se comporta bien en las distancias cercanas a la pared (Liu, 2016). Generalmente existen dos formas de solventar esta situación.

La primera forma consiste en emplear el modelo de turbulencia de forma de que resuelva este fenómeno en las cercanías de la pared. Para este enfoque, es necesario incluir mediante mallado elementos dentro a la capa de desarrollo laminar, teniendo que mallar con gran resolución cerca de las paredes sin importar la extensión que pueda presentar el dominio, lo que conlleva una gran destinación de recursos computacionales.

Por otra parte, las funciones de pared son funciones que buscan modelar la zona cercana a la pared sin necesidad de presentar un mallado tan refinado. Estas funciones son ecuaciones empíricas usadas para satisfacer la física del fluido en la zona cercana a la superficie. Para asegurar precisión en el análisis se debe cumplir una condición de mallado más laxa que en al caso anterior. A continuación se desarrollan ciertos términos necesarios para entender las aproximaciones realizadas por los programas de simulación CFD (basado en (Versteeg y Malalasekera, 2016)).

Se define la altura adimensional y^+ mediante la ecuación 2.28, siendo u_τ un término denominado velocidad de fricción. Se define además, una velocidad adimensional u^+ mediante la expresión 2.30. Exceptuando otras condiciones o propiedades se debiese notar una relación entre la altura y la velocidad adimensional, siendo interesante de identificar la relación $u^+(y^+)$. Esta función se conoce como el perfil universal de velocidad para un flujo turbulento cercano a una pared (Earls Brennen, 1995).

$$y^+ = \frac{y \cdot u_\tau}{\nu} \tag{2.28}$$

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} \tag{2.29}$$

$$u^+ = \frac{u}{u_\tau} \tag{2.30}$$

El perfil universal de velocidad se muestra a continuación en donde se distinguen tres secciones en función de la distancia y^+. La subcapa viscosa es aquella dominada por los efectos viscosos (y^+

< 5), donde se asume que el esfuerzo de corte del fluido es igual al de la pared, de forma que el perfil de velocidad se considera lineal. Sin importar la característica del fluido, las fluctuaciones turbulentas tienden a cero en contacto con la pared, por lo que siempre existe una capa de perfil laminar que cumple con esta condición.

$$u^+ = y^+ \tag{2.31}$$

El área de transición o "zona buffer" se extiende para el rango $5 < y^+ < 30$. Aquí, los esfuerzos viscosos y turbulentos presentan similares magnitudes y dado que el perfil de velocidad no está bien definido, las aproximaciones originales de funciones de pared omiten este rango de distancia. Sin embargo, las funciones de pared mejoradas permiten que el centroide de la primera celda de la malla se encuentre en este rango de altura.

Por último, se define el área logarítmica para valores comprendidos entre $30 < y^+ < 200$. En esta zona dominan los esfuerzos turbulentos y el perfil de velocidad varía de forma logarítmica siguiendo la siguiente expresión. Aquí κ es la constante de von Kármán igual a 0.4187 y E es una constante que varía dependiendo de la rugosidad de la superficie. Para paredes lisas se considera $E = 9.793$.

$$u^+ = \frac{1}{\kappa} \cdot ln(E * y^+) \tag{2.32}$$

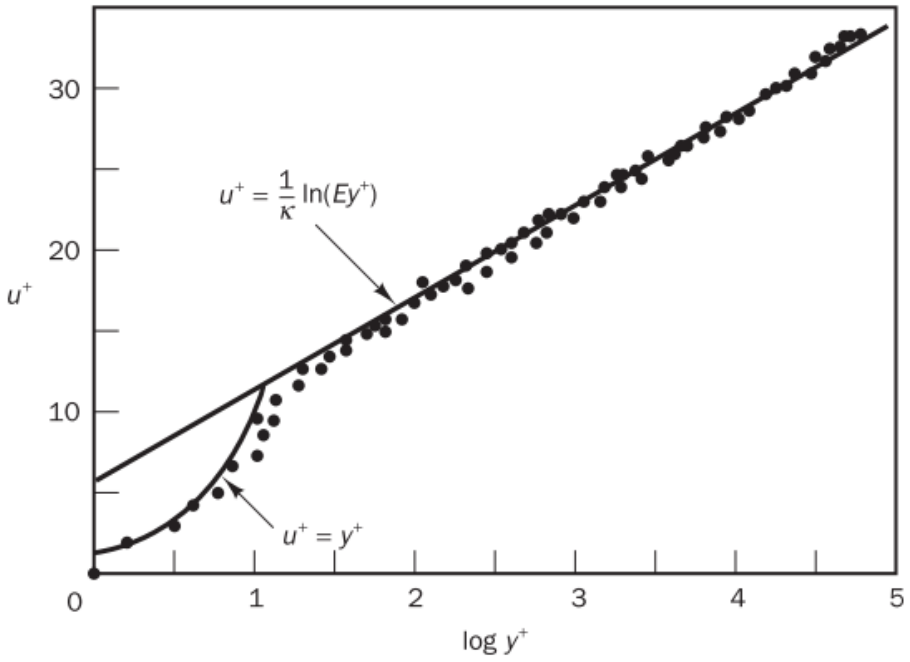


Figura 2.5: Distribución de velocidades cerca de pared sólida (fuente: Versteeg y Malalasekera, 2016)

2.2. Mecánica de fluidos computacional - Práctica

2.2.1. Parámetros de control del método numérico

Como se expone en la sección anterior, ambos programas presentan diversos esquemas de resolución para resolver cada término de las ecuaciones gobernantes que determina el problema. En esta sección se describen las variables de control estipuladas por cada programa para que, mediante el solver escogido, se lleve a cabo la simulación numérica del caso. Primero se hace revisión de OpenFOAM, ya que presenta en un mismo documento las variables importantes a mostrar en esta sección.

En OpenFOAM las variables de control están localizadas en el archivo *fvSolution*. Aquí se establece el mecanismo de discretización de cada variable (U, k, p, etc) así como los parámetros de control del esquema macro que las utiliza, SIMPLE por ejemplo. A continuación se ve un extracto del documento empleado en la resolución del caso de estudio en donde se aprecia el solver y los parámetros establecidos para la resolución de la variable de presión p_rgh (para más detalle ver Anexo A)

Cada diccionario perteneciente a una variable a resolver tendrá que tener como mínimo un *solver* y un método de tolerancia para la solución. Este último dependerá de la implementación del usuario ya que el proceso iterativo de resolución puede terminar mediante más de un gatillante.

Los residuos de las soluciones iterativas son una medida de error en la solución. Mientras más pequeño sea, más precisa es la solución. Estos valores se consiguen al evaluar las soluciones numéricas de una ecuación y medir el error entre el lado izquierdo y derecho de la igualdad, siendo este valor igual a cero cuando se ha alcanzado la solución y satisfecho la ecuación inicial (OpenCFD, 2019; Moukalled et al., 2016).

Código 2.1: Extracto fvSolution

```
1  p_rgh
2  {
3     solver          GAMG;
4      tolerance       1e-8;
5      relTol          0.01;
6     maxIter         1000;
7     smoother         DICGaussSeidel;
8  }
9  p_rghFinal
10 {
11    \$p_rgh;
12    relTol          0;
13 }
```

2.2.2. Mallado

El proceso de mallado es uno de los pasos más importantes luego de definir la geometría del caso de estudio. La mecánica de fluidos computacional, requiere de una subdivisión del dominio principal en numerosos subdominios más pequeños que no se superpongan entre si para poder resolver el problema físico del dominio geométrico inicial. Este proceso de subdivisión resulta en la generación de una malla cuyos elementos representan pequeños volúmenes de control. La solución del problema en términos de propiedades físicas, se define sobre los nodos formados a partir de la subdivisión del dominio, siendo la precisión de los resultados generalmente una función del número de celdas (Versteeg y Malalasekera, 2016).

El procedimiento de aumentar los elementos de la malla se conoce como ”refinamiento de malla” y trae consigo un considerable aumento de costo computacional. Sin embargo, se debe procurar que el nivel de mallado sea tal que las soluciones obtenidas se acerquen a la independencia de este (Hong et al., 2017). Esta independencia se traduce como la no variación sustantiva de resultados ante un refinamiento de malla. Un refinamiento aceptable es el doble de elementos en la malla, de presentar variaciones significativas en los valores obtenidos, entonces la solución es función del número de elementos considerados y debe volver a realizase un refinamiento (Tu, 2019).

Se extrae de lo anterior que el mallado de la geometría debe procurar no entorpecer la obtención de los resultados, es decir, debe adecuarse tanto a la geometría del problema como a las solicitudes computacionales para resolver la física asociada. Dependiendo de la geometría del caso serán utilizadas diferentes técnicas de mallado. A continuación, se describen brevemente tres aspectos fundamentales para la generación y selección de un correcto mallado.

2.2.3. Tipos de mallado

La principal distinción a la hora del tipo de mallado corresponde al estructuramiento de las mallas, en donde se encuentran mallas estructuradas, semi-estructuradas y mallas no estructuradas. Es importante mencionar que el proceso de mallado comúnmente es realizado por un software específico para este fin, si bien los hay gratuitos, existe una amplia variedad de códigos comerciales con poderosas herramientas de mallado incorporadas (Tu, 2019). Dado que el alcance de este trabajo no transciende en los algoritmos de generación de malla, la clasificación siguiente engloba los resultados visibles del ordenamiento de los elementos (o celdas) de la malla según los lineamientos de (Versteeg y Malalasekera, 2016; OpenCFD, 2019; Tu, 2019).

Mallas estructuradas

- Mallas cartesianas: Involucran la discretización de las ecuaciones gobernantes de la forma más pura, ya que el mallado de la geometría es coincide con la discretización empleada por el modelo. Como ejemplo, se puede tener una geometría cúbica con elementos del mallado de la misma geometría. Presenta problemas de precisión al mallar zonas curvas.

- Mallas ajustadas al cuerpo (Body-fitted): Se basan en trasladar el mapeo del dominio físico a uno computacional. Dependiendo de la geometría, puede ser altamente beneficioso en contornos cilíndricos, sin embargo, no admite complejos mapeos 3D en geometrías con objetos o partes internas.

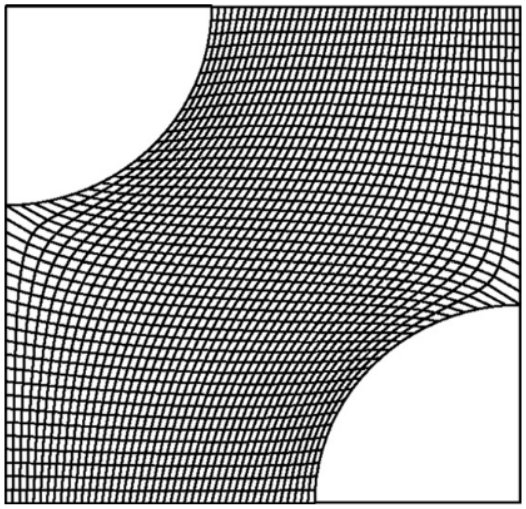


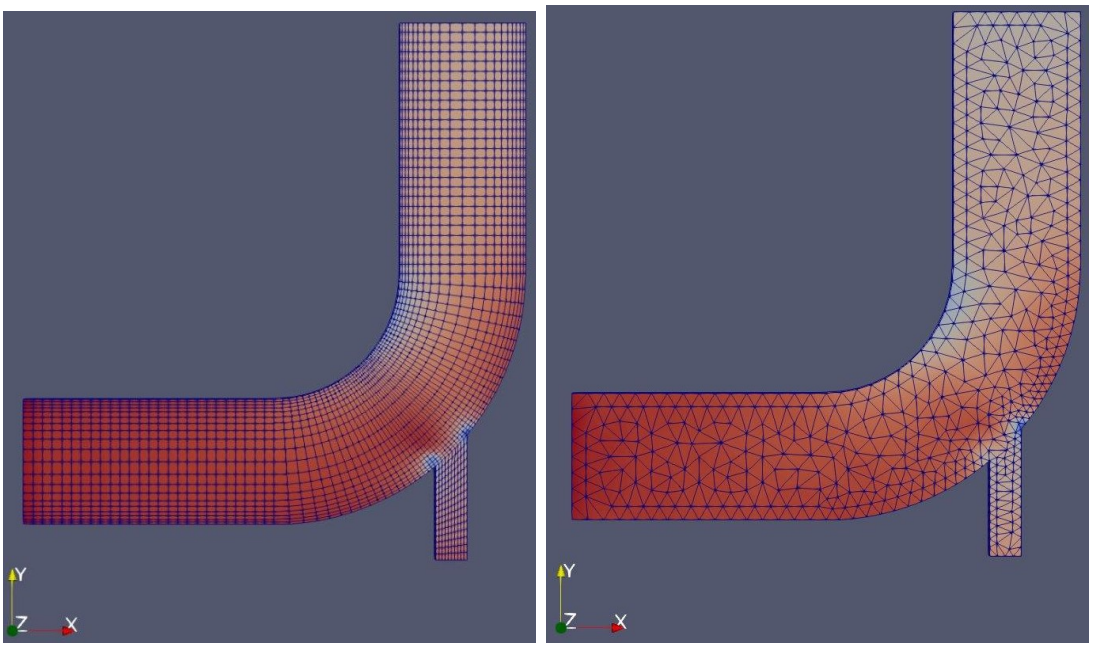
Figura 2.6: Generación de malla estructurada tipo H-Grid

- Mallas de bloque estructurado (Block-structured meshes): Se basa en la división de la geometría de manera de emplear mallados estructurados en diversas partes. Este tipo de mallado puede resultar beneficioso ante geometrías complejas ya que no solo se limita a estructuras de tipo cartesiano, presentando considerables ventajas de adaptación al medio. Este mallado da paso al mallado con superposición de bloques (también denominado quimera), donde se dispone de variados tipos de mallas estructuradas para adaptarse al medio como se puede apreciar en la figura 2.6

Mallas no estructuradas

En este tipo de mallado no existe una estructura rígida predeterminada, los volúmenes de control pueden tener formas arbitrarias dependiendo del tipo de elemento utilizado para la generación de la malla. En la figura 2.7 se ejemplifica el mallado no estructurado para un caso de dos dimensiones realizado en OpenFOAM.

Dentro de los cambios que implica un mallado de este tipo es la ubicación de la información. En un mallado cartesiano, los valores de presión y velocidad son almacenados en los nodos externos, mientras que en el mallado no estructurado estos valores se registran comúnmente en un nodo central del elemento (co-located arrangement). A pesar de esto, este tipo de mallas implica un ahorro considerable de tiempo en la generación del mallado junto con beneficios considerables a la hora de realizar refinamientos locales o globales, lo cual dentro de otras cosas, lo ha convertido en el método más común de mallado en software comerciales actualmente.



(a) Malla estructurada

(b) Malla no estructurada

Figura 2.7: Ejemplo de tipos de mallado realizados en OpenFOAM v7 visualizados mediante Paraview

Mallas semi-estructuradas

El mallado semi-estructurado permite resolver problemas de geometrías complejas de forma eficiente (Heidenreich, Gaspar, Lisbona, y Rodrigo, 2010). El método empleado consiste en sobre una malla no estructurada que capture correctamente la geometría del dominio, generar un mallado estructurado sobre los elementos geométricos complejos o que requieran mayor detalle. El resultado es un mallado globalmente no estructurado pero si localmente. Otras ventajas sobre este tipo de mallados se proyectan en la resolución de los modelos de elementos finitos (Heidenreich et al., 2010).

2.2.4. Tipos de elementos

Dentro de la topología de los elementos, se realiza una diferenciación sujeta a las dimensiones del espacio. Diferenciando entre elementos planos y en las tres dimensiones se tendrán tipos de elementos que impliquen una mayor estructura en el mallado que otros. Un elemento de la malla es básicamente un polígono en 2D o un poliedro en el espacio. Los elementos más utilizados para el mallado se muestran en la figura 2.8. El tipo de caras de los elementos tridimensionales también representan estructuras bidimensionales (Moukalled et al., 2016).

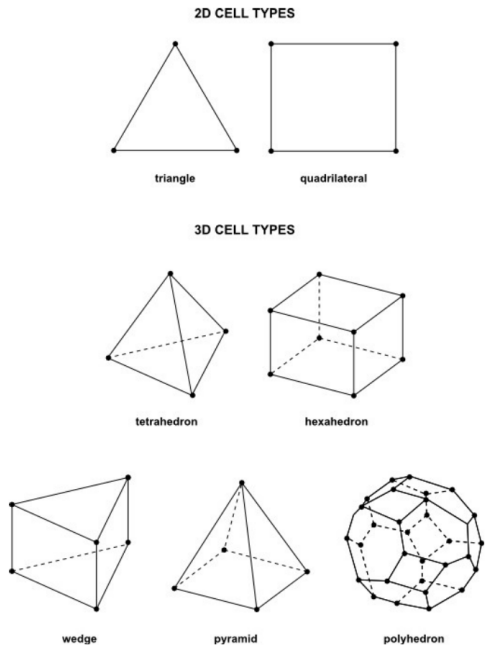


Figura 2.8: Tipos de elementos

ANSYS Fluent utiliza estructuras de datos internas para asignar orden a las celdas, caras y nodos del mallado, de forma que no requiere una indexación espacial de la forma i,j,k para localizar las celdas contiguas (Fluent, 2019). El gran beneficio de esto es la flexibilidad en el uso de diversas topologías para abordar el proceso de simulación.

Por otra parte, OpenFOAM fuerza una evaluación sobre el mallado y los elementos que lo componen (geometrías, volúmenes, áreas, centroides y factores de peso en las caras de los elementos), de forma de asegurar que se cumple con un set de restricciones de validez impuestas por el usuario.

Este programa establece que si bien las correcciones pueden resultar frustrantes, un correcto mallado impedirá soluciones defectuosas desde el inicio (OpenCFD, 2019; Moukalled et al., 2016).

2.2.5. Validación de mallado

La calidad de la malla juega un importante rol en la precisión y en la estabilidad de los cálculos numéricos. Independiente del tipo de malla usada en el dominio, comprobar la calidad del mallado es esencial (Fluent, 2019). En esta sección, se revisan los parámetros más utilizados en la comprobación de calidad de malla.

- Ortogonalidad: La ortogonalidad dice relación con la orientación del mallado en comparación con las coordenadas cartesianas, siendo evaluada según la orientación de cada celda con su vecina. La importancia de mantener una buena ortogonalidad se relaciona con la con la precisión y la estabilidad de las simulaciones, debido a que las discretizaciones de las ecuaciones gobernantes implican balances y cálculos a través de las caras de los elementos. Con una menor

ortogonalidad los cálculos se vuelven más imprecisos, por lo que en la generación del mallado generalmente se aplican factores de corrección ortogonal (OpenCFD, 2019).

- Skewness: Dirá relación de la diferencia de forma entre una celda cualquiera y una celda del mismo volumen pero de geometría equilátera. Celdas altamente deformadas disminuyen la precisión de la solución e inducen a la desestabilización de esta misma (Fluent, 2019).

Las formas de manejar a ortogonalidad y la falta de simetría difiere entre si para cada software comercial. OpenFOAM maneja parámetros sin normalizar y es necesario conocer con precisión si el cálculo de los parámetros es basado en la celda o la cara de los elementos. Por otra parte, ANSYS Fluent normalizará los valores entregando el peor caso, por lo que el correcto entendimiento de los datos mostrados por los programas es fundamental.

En la práctica se encuentran diferencias de convergencia ante diversas mallas para estos programas como las mostradas por (Welahettige y Vaagsaether, 2018), donde OpenFOAM muestra un inferior manejo en mallas de alto factor de skewness, es decir, es menos robusto ante un peor mallado en comparación a ANSYS Fluent.

En cuanto a parámetros, para la ortogonalidad y el Skewness definidos por ANSYS Fluent en el rango de 0 a 1, serán deseados valores cercanos a la unidad en el caso del primer factor y valores cercanos a cero en el caso del Skewness. OpenFOAM en tanto limita la ortogonalidad cercana al 70 % y diferencia parámetros de Skewness en cara y cuerpo, sugiriéndose valores menores a 4 y a 20 respectivamente (OpenCFD, 2019).

Capítulo 3

Metodología

Si bien el objetivo principal de este trabajo de memoria de título es transversal a la elección de un caso de estudio, la selección de éste condicionará el desarrollo de la metodología. Es por esto que SAME S.A. proporciona al alumno información de dos casos de estudio reales de ventilación industrial, de manera de abarcar las principales condiciones de ventilación presentes en proyectos de esta índole.

Si bien en un inicio se desean abordar ambos caso de estudio, en el desarrollo del trabajo se opta por la reducción a una sola geometría de análisis, con la salvedad de que estén presentes todos los fenómenos relevantes de ambos casos. En este capítulo se resume el principal caso de estudio con parámetros originales de operación, la simplificación realizada a este caso para ser objeto de análisis y el desarrollo llevado a cabo sobre este caso de estudio que permite el análisis comparativo en los programas CFD.

3.1. Caso de estudio

El caso de estudio presentado a continuación corresponde a un Laboratorio metalúrgico. Aquí, se analizan muestras de material que requieren diversos procesos de conminución los cuales conllevan la generación de material particulado en suspensión. La propuesta de solución del proyecto busca mejorar el control de polvo en estas dependencias, para lo cual se decide aumentar la extracción de aire que luego será filtrado mediante un filtro de mangas en el exterior.

Tabla 3.1: Dimensiones Laboratorio de Conminución

Sala	Largo [m]	Ancho [m]	Alto [m]	Volumen [m3]
Conminución	7.90	5.17	2.85	116.40
Alimentación	4.55	5.17	2.85	67.04
Concentrado	3.10	5.17	2.85	45.68
Baño/camarín	3.05	5.17	2.85	44.94

En la imagen 3.1 se aprecia la propuesta de solución. Aquí, existe una linea de insuflado de aire, mecanismo por el cual se inyecta aire de renovación a las dependencias. La propuesta de extracción de aire implica tres mecanismos, extracción mediante campanas superiores, extracción directa sobre equipos que emiten material particulado y extracción mediante una pared extractora. La vista isométrica del dibujo también muestra la posición relativa de cada mecanismo de extracción propuesto, siendo los colectores superiores e inferiores los encargados de canalizar la extracción de material hacia el filtro de mangas final.

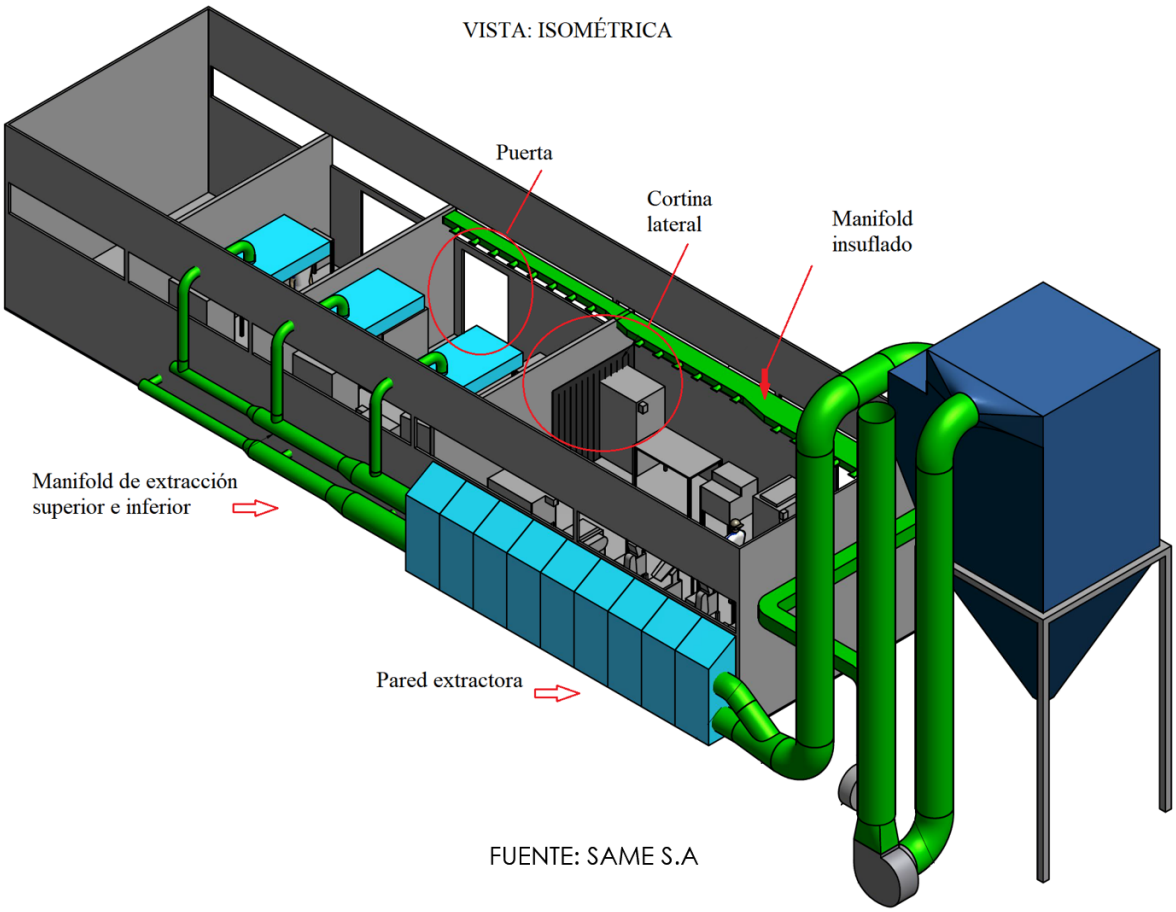


Figura 3.1: Laboratorio de conminución

La extracción de aire es impulsada por un ventilador centrífugo de 14.000 CFM[1]. En las tablas 3.2 y 3.1 se detalla la subdivisión de los caudales que extrae cada equipo y las dimensiones de las salas del laboratorio de conminución.

[1] CFM: Unidad de medida de caudal equivalente a pies cúbicos por minuto. 1 CFM equivale a 28.3 [l/min]

Tabla 3.2: Resumen Caudales de extracción por sala

Sala	Equipo	Cantidad	Caudal [CFM]	Total [CFM]
Conminución	Pared extractora	1	5000	5000
Alimentación	Campana	2	1500	3000
	Estación de trabajo	2	1000	2000
Concentrado	Campana	1	2000	2000
	Estación de trabajo	1	1000	1000
Baño/camarín	-	-	-	-

3.2. Simplificación caso de estudio final

Expuesto el caso anterior, se busca realizar una representación de la geometría que permita tanto el análisis de diversas condiciones de borde como del correcto fenómeno de ventilación. Es por esto que se decide simplificar la geometría optando por modelar un solo espacio que represente los elementos presentados en las dependencias del Laboratorio de Conminución.

Empleando el programa Inventor 2020, se modela la geometría que da origen a la figura 3.2. Esta representación alberga las dimensiones existentes de la *Sala de Alimentación*, donde se adiciona una pared extractora perteneciente a la *Sala de Conminución* eliminando la extracción directa a equipos específicos. El equipamiento interno existente se ve simplificado a dos geometrías capaces albergar condiciones de borde futuras y a un elemento adicional de mueblería.

Las campanas de extracción (5 y 6) tienen la función de extraer aire de forma uniforme desde la parte superior de la sala hacia un tubo de extracción ligado al manifold principal. Se dispone la geometría de forma de mantener ambas campanas dispuestas en el diseño original de la Sala de Alimentación. Ambas campanas tienen las siguientes dimensiones: 1200[mm] x 1800[mm] x 100[mm], además, el inferior presenta una placa con orificios de 35[mm] de diámetro dispuestos en un entramado de 7 x 10 como se aprecia en la figura 3.3.

El principal ingreso de aire a la sala se origina a través de las boquillas de insuflado (o insufladores - N°10). Si bien el ingreso proveniente desde otros lugares es factible, se considera que la principal renovación de aire de la sala es a través de esta geometría. Bajo esta consideración, el ingreso secundario de aire se limita a la cortina lateral de la sala (N°9).

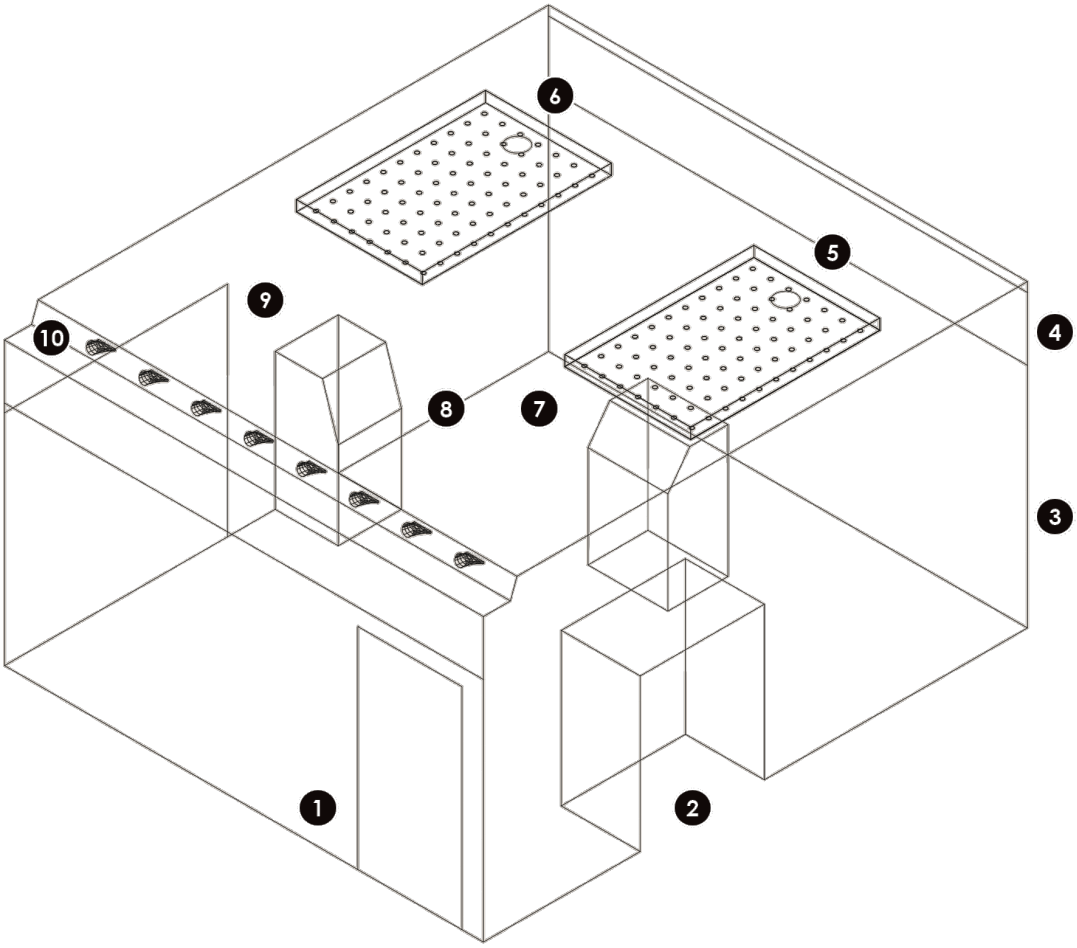


Figura 3.2: Sala de Alimentación: 1: Puerta; 2: Mueble; 3: Pared extracción; 4: Celosía; 5: Campana 1; 6: Campana 2; 7: Fuente 1; 8: Fuente 2; 9: Cortina lateral; 10: Insufladores

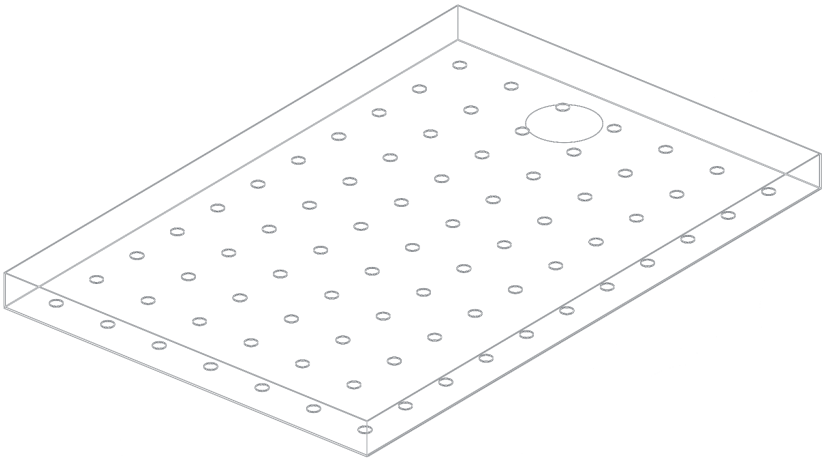


Figura 3.3: Campana de extracción superior

Los ocho insufladores presentes en la sala están dispuestos en un ángulo de 20° hacia el interior de esta, de manera de forzar una recirculación de aire y ayudar a la renovación de este. En las siguientes imágenes se puede apreciar con más detalle la geometría modelada.

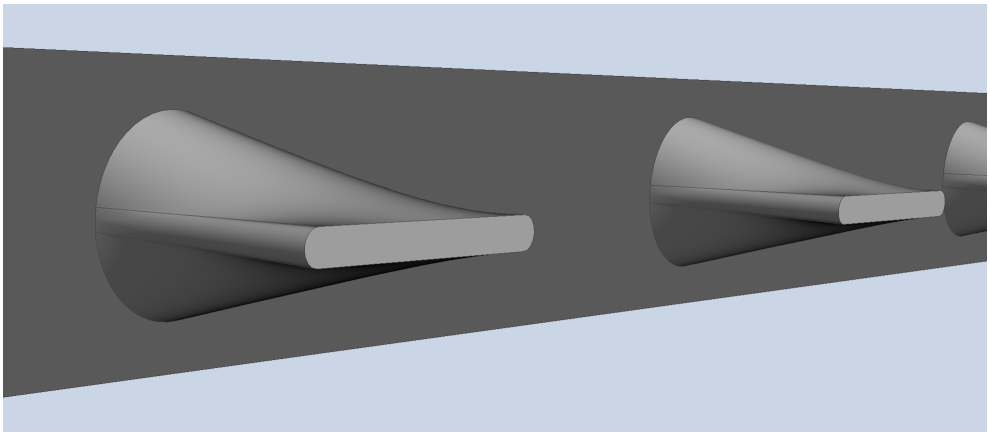


Figura 3.4: Boquillas de insuflado

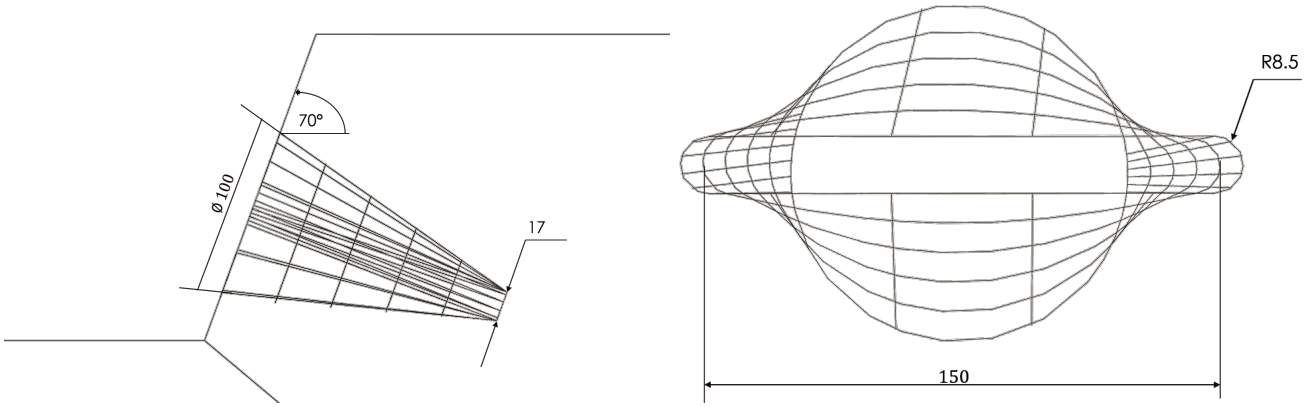


Figura 3.5: Dimensiones boquilla de insuflado

Con el fin de tener un mayor control en el proceso de mallado se realizan subdivisiones a la geometría presentada anteriormente (ver figura 3.7). Para fines de la realización del mallado se tendrán tres volúmenes: 1: Entorno insufladores, 2: Entorno campanas y 3: Volumen remanente.

Durante la formulación del caso de estudio en el programa OpenFOAM, se encuentran una serie de aspectos que impulsan profundos cambios en la metodología de mallado. Las principales consideraciones recaen en el tipo y cantidad de elementos al momento de realizar la malla. La topología de la malla está directamente relacionada con la estabilidad de las simulaciones, mientras que la cantidad de elementos aumenta considerablemente el tiempo de calculo en este software.

Es por estas consideraciones, que luego de realizar simulaciones exitosas en OpenFOAM con un mallado de la geometría mostrada en la figura 3.2, se adoptan los siguientes cambios:

- Se opta por generar mallas con topología polinomial, reduciendo significativamente la cantidad de elementos en los mallados.
- Se realiza una simplificación adicional a la geometría del caso de estudio eliminando la placa perforada de ambas campanas de extracción.

Ante esta modificación en la geometría, se adiciona un análisis sobre la extracción superior de aire. Para argumentar este cambio, la geometría mostrada en la figura 3.6 es sometida a simulaciones que den cuenta de los cambios en ventilación que implica la realización aquí descrita. Esta geometría al igual que la principal, es sometida a un test de independencia de malla para su óptima validación.

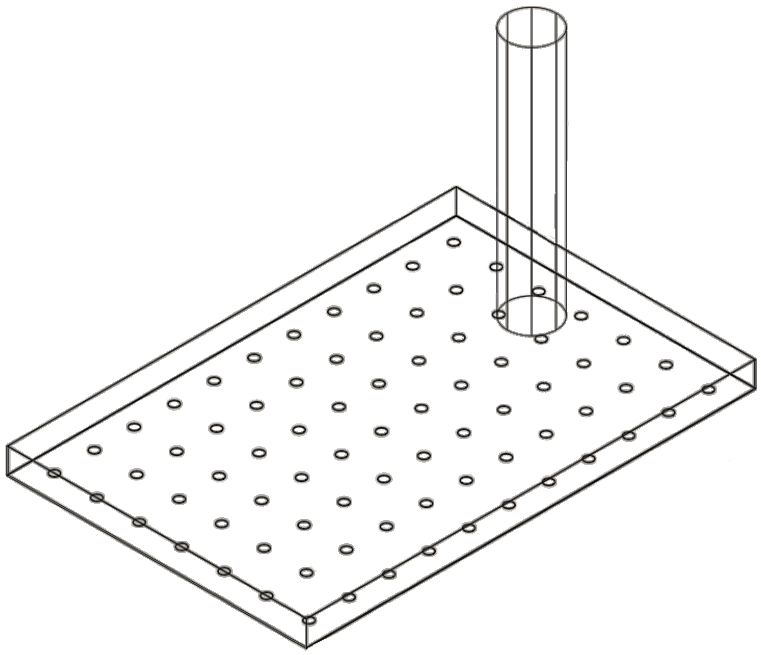


Figura 3.6: Geometría campana

En la figura 3.7 se pueden apreciar los tres volúmenes descritos anteriormente. Esta geometría final es la empleada en el proceso de mallado y posteriores simulaciones en ambos programas.

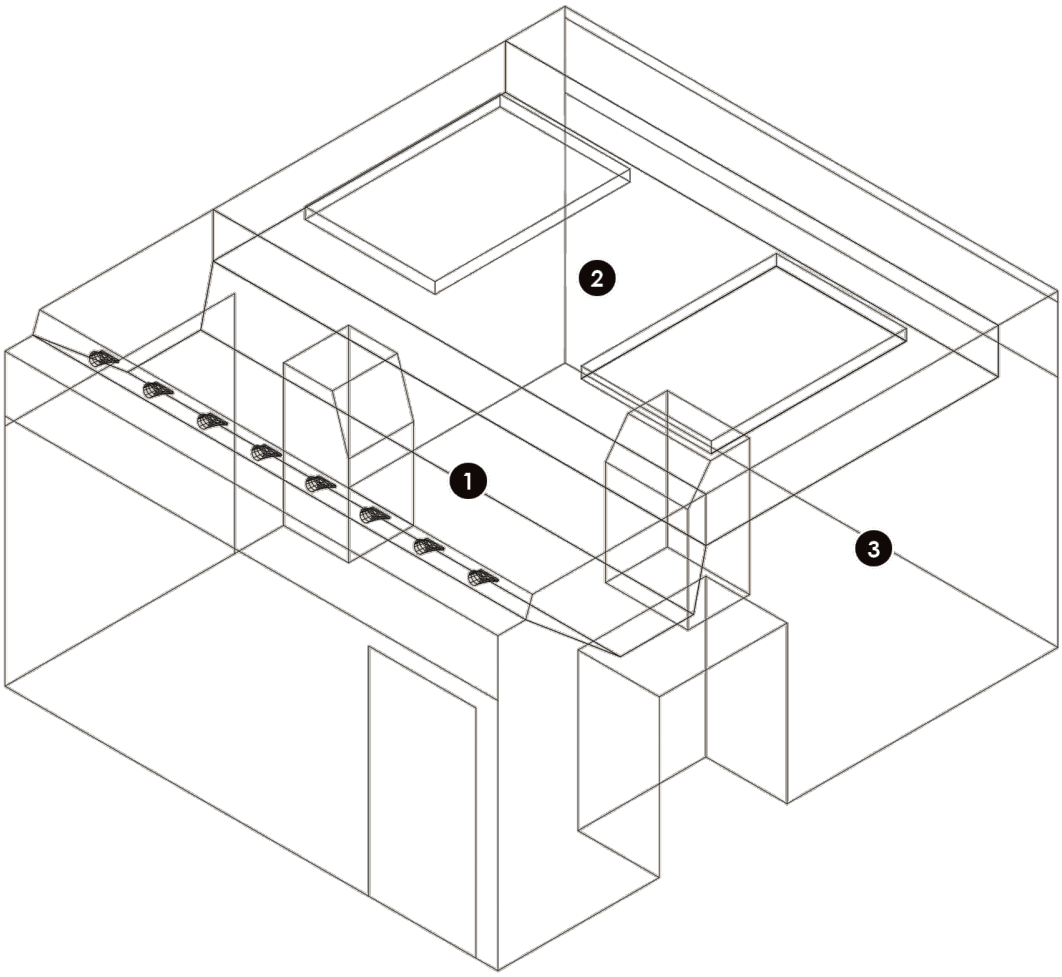


Figura 3.7: Sala de alimentación con simplificaciones en campanas y subdivisiones de volumen

3.3. Condiciones de borde y funciones de pared

Si bien tanto las condiciones de borde como las funciones de pared son parámetros que deben ser aplicados en ambos programas de simulación, las configuraciones distan bastante entre uno y otro generando cambios mayores de estructura en el caso de OpenFOAM. Para este programa, los cambios en el caso de estudio como la resolución de la ecuación de energía o la adición de una nueva fase del fluido conllevan una serie de modificaciones que no se encuentran en el programa Fluent. Es por esto que en esta y en la siguiente sección se tiene especial cuidado en la presentación de los parámetros involucrados en las simulaciones.

La estructura de OpenFOAM requiere solvers específicos para la resolución de la ecuación de energía. Siguiendo las opciones presentadas en la sección 2.1.5 se emplean los algoritmos *Buoyant*

SIMPLE Foam y *Buoyant PIMPLE Foam* para la resolución de los casos con transferencias de calor. Para los casos sin este fenómeno, se emplean los algoritmos SIMPLEC en estado estacionario y PIMPLE para la formulación transiente.

De acuerdo a la geometría del caso de estudio (figura 3.2), se tienen dos ingresos (cortina lateral e insufladores) y tres salidas de flujo (ambas campanas y pared extractora). Teniendo en cuenta además los caudales expuestos en la tabla 3.2 y las consideraciones geométricas sobre el caso de estudio, se establecen las siguientes condiciones de borde para los flujos expuestos en la tabla 3.3.

Todas las condiciones de velocidad son impuestas sobre la condición de perpendicularidad en la dirección de la superficie hacia el interior del volumen de control. Para Fluent, en la condición de *velocity inlet* el signo negativo significará un cambio de sentido en la condición, equivaliendo a una imposición de salida del flujo en dicha superficie, convención usada en la tabla a continuación. OpenFOAM en tanto, opera bajo coordenadas globales y en las condiciones de velocidad empleadas en este trabajo, es preciso especificar la dirección del fluido de forma vectorial.

Tabla 3.3: Condiciones de borde de flujo Sala de Alimentación (SA)

Superficie	Ingreso/extracción	Tipo de condición	valor
Cortina lateral	ingreso	pressure inlet	calculado
Insufladores	ingreso	velocity inlet	35.15 [m/s]
Campanas	extracción	velocity inlet	-0.328 [m/s]
Pared extractora	extracción	velocity inlet	-0.05 [m/s]

Como se ha mencionado anteriormente, las implementaciones de las condiciones de borde variarán dependiendo el caso de estudio y el software en cuestión. En las tablas 3.6 - 3.9 se resumen las implementaciones de los parámetros aquí descritos para cada caso, en conjunto con las condiciones de borde explícitas para cada parámetro que requiere OpenFOAM.

Para la geometría auxiliar de la campana, se tiene en consideración el objetivo de su estudio para imponer las condiciones de borde. Se considera que ingresa aire desde tres de las cuatro caras de la geometría, descontando la superficie que equivale a la pared de extracción. La condición de extracción del flujo es impuesta al extremo superior del tubo de extracción.

Tabla 3.4: Condiciones de borde de flujo Campana

Superficie	Ingreso/extracción	Tipo de condición	valor
Extremo de tubo extracción	extracción	velociity inlet	-23.45 [m/s]
Contorno	ingreso	pressure inlet	-
Pared extractora	-	pared	-

De acuerdo a lo expuesto en la sección de antecedentes, OpenFOAM requiere que el usuario determine explícitamente las funciones de pared y las condiciones de borde de cada variable presente en la formulación de un caso. A continuación en las tablas 3.6 - 3.9 se aprecia esta información para cada superficie relevante del volumen de control de la Sala de Alimentación. Se hace la distinción entre dos formulaciones de OpenFOAM distintas de forma de incluir todas las condiciones posibles.

3.4. Independencia de mallado

El análisis de las mallas empleadas se realiza sobre la formulación del caso de estudio sin transferencia de calor, es decir, considerando los valores mostrados en las tablas 3.3 y 3.4. Para cada volumen de control (Sala y campana) se realiza un mallado con parámetros óptimos para ser empleado en ambos software. A partir de dicho mallado base se realiza al menos una malla más fina y otra más gruesa tomando valores de velocidad en puntos representativos de la geometría. Las coordenadas de dichos puntos se muestran a continuación. Tanto el punto medio de SA como de Campana son exactamente el punto central de ambos volúmenes de control. El punto de nombre *tubo extracción* en tanto corresponde a la disposición media del tubo de extracción de la campana mostrada en 3.6.

Tabla 3.5: Coordenadas para test de independencia

Geometría	Nombre	Coordenada		
		x	y	z
SA	Jet de insuflado	2.05	0.8927	2.5238
Campana	interior campana	3.545	3.8735	2.8
	tubo extracción	3.545	3.8735	3.35

3.5. Parámetros de simulación

En esta sección se busca dar cuenta de la implementación de los parámetros de control utilizados en las simulaciones definitivas en ambos programas. Para dar con estos parámetros y configuraciones, se ha procurado la mayor similitud numérica para tanto condiciones de borde como iniciales siempre y cuando en la práctica se logre la estabilidad numérica de la simulación. Dadas las características intrínsecas de cada programa, se resumen a continuación todos los valores y funciones de borde empleadas según corresponde a cada parámetro en el caso de las simulaciones en estado transiente aplicando y no la aproximación de Boussinesq en el caso de OpenFOAM.

Tabla 3.6: Implementación condiciones de borde 1/4

	Superficie	Entrada	alphat	epsilon
SIMPLE/PIMPLE	Insufladores	type	-	fixedValue
		value	-	uniform 14.855
	Cortina	type	-	fixedValue
		value	-	uniform 14.855
	Campanas	type	-	zeroGradient
		value	-	-
	Pared extracción	type	-	zeroGradient
		value	-	-
	Fuente 1 y 2	type	-	epsilonWallFunction
		value	-	uniform 14.855
	Paredes	type	-	epsilonWallFunction
		value	-	uniform 14.855
Buoyant SIMPLE/PIMPLE	Insufladores	type	calculated	fixedValue
		value	uniform 0	uniform 14.855
	Cortina	type	calculated	fixedValue
		value	uniform 0	uniform 14.855
	Campanas	type	calculated	zeroGradient
		value	uniform 0	-
	Pared extracción	type	calculated	zeroGradient
		value	uniform 0	-
	Fuente 1 y 2	type	compressible:: alphatWallFunction	epsilonWallFunction
		value	uniform 0	uniform 14.855
	Paredes	type	compressible:: alphatWallFunction	epsilonWallFunction
		value	uniform 0	uniform 14.855
ANSYS Fluent	Insufladores	valor	-	14.855
	Cortina	valor	-	14.855
	Campanas	valor	-	14.855
	Pared extracción	valor	-	14.855
	Fuente 1 y 2	valor	-	14.855
	Paredes	valor	-	14.855

Tabla 3.7: Implementación condiciones de borde 2/4

	Superficie	Entrada	k	nut
SIMPLE/PIMPLE	Insufladores	type value	fixedValue uniform 0.375	calculated uniform 0
	Cortina	type value	fixedValue uniform 0.375	calculated uniform 0
	Campanas	type value	type value	calculated uniform 0
	Pared extracción	type value	type value	calculated uniform 0
	Fuente 1 y 2	type value	kqRWallFunction uniform 0.375	nutkWallFunction uniform 0
	Paredes	type value	kqRWallFunction uniform 0.375	nutkWallFunction uniform 0
Buoyant SIMPLE/PIMPLE	Insufladores	type value	fixedValue uniform 0.375	calculated uniform 0
	Cortina	type value	type fixedValue uniform 0.375	calculated uniform 0
	Campanas	type value	type value	calculated uniform 0
	Pared extracción	type value	type value	calculated uniform 0
	Fuente 1 y 2	type value	kqRWallFunction uniform 0.375	nutkWallFunction uniform 0
	Paredes	type value	kqRWallFunction uniform 0.375	nutkWallFunction uniform 0
ANSYS Fluent	Insufladores	valor	0.375	-
	Cortina	valor	0.375	-
	Campanas	valor	0.375	-
	Pared extracción	valor	0.375	-
	Fuente 1 y 2	valor	0.375	-
	Paredes	valor	0.375	-

Tabla 3.8: Implementación condiciones de borde 3/4

	Superficie	Entrada	p	p_rgh
SIMPLE/PIMPLE	Insufladores	type	zeroGradient	-
		value	-	-
	Cortina	type	zeroGradient	-
		value	-	-
	Campanas	type	calculated	-
		value	uniform 0	-
	Pared extracción	type	calculated	-
		value	uniform 0	-
	Fuente 1 y 2	type	zeroGradient	-
		value	-	-
	Paredes	type	zeroGradient	-
		value	-	-
Buoyant SIMPLE/PIMPLE	Insufladores	type	calculated	zeroGradient
		value	$internalField	-
	Cortina	type	calculated	fixedValue
		value	$internalField	$internalField
	Campanas	type	calculated	zeroGradient
		value	$internalField	-
	Pared extracción	type	calculated	zeroGradient
		value	$internalField	-
	Fuente 1 y 2	type	calculated	fixedFluxPressure
		value	$internalField	uniform 1e5
	Paredes	type	calculated	fixedFluxPressure
		value	$internalField	uniform 1e5
ANSYS Fluent	Insufladores	valor	-	-
	Cortina	valor	-	-
	Campanas	valor	-	-
	Pared extracción	valor	-	-
	Fuente 1 y 2	valor	-	-
	Paredes	valor	-	-

Tabla 3.9: Implementación condiciones de borde 4/4

	Superficie	Entrada	T	U
SIMPLE/PIMPLE	Insufladores	type value	fixedValue 293	fixedValue uniform (0 33.03 -12.02)
	Cortina	type value	fixedValue 293	zeroGradient -
	Campanas	type value	zeroGradient -	fixedValue uniform (0 0 0.328)
	Pared extracción	type value	zeroGradient -	fixedValue uniform (0 0.05 0)
	Fuente 1 y 2	type value	fixedValue 373	noSlip -
	Paredes	type value	zeroGradient -	noSlip -
Buoyant SIMPLE/PIMPLE	Insufladores	type value	fixedValue 293	fixedValue uniform (0 33.03 -12.02)
	Cortina	type value	fixedValue 293	zeroGradient -
	Campanas	type value	zeroGradient -	fixedValue uniform (0 0 0.328)
	Pared extracción	type value	zeroGradient -	fixedValue uniform (0 0.05 0)
	Fuente 1 y 2	type value	fixedValue 373	noSlip -
	Paredes	type value	zeroGradient -	noSlip -
ANSYS Fluent	Insufladores	valor	293	35.15
	Cortina	valor	293	-
	Campanas	valor	293	-0.328
	Pared extracción	valor	293	-0.05
	Fuente 1 y 2	valor	373	-
	Paredes	HF[1]	0	-

[1] HF representa la condición de flujo de calor en ANSYS. Aquí el valor por defecto define una condición adiabática de transferencia igual a 0 [W/m2]

Capítulo 4

Resultados y discusión

En esta sección se busca dar cuenta de los resultados obtenidos tanto para validar el proceso de simulación como para la obtención de información cuantificable sobre el desempeño de ambos programas. Para esto, se presentan en una primera instancia las pruebas de independencia de mallado realizadas para el caso de estudio, mostrando posteriormente los parámetros de calidad finales sobre las mallas definitivas.

Más adelante, se expone la información que da cuenta de las implicancias sobre la simplificación de la campana de extracción en la geometría principal del caso de estudio. Se muestra también el seguimiento de parámetros de velocidad y temperatura en puntos al interior de la sala en el transcurso de una simulación en estado transiente de 10 segundos luego de una incialización en estado estacionario. Finalmente, se visualiza la dinámica del fluido simulada al interior del volumen de control y se cuantifican principales diferencias entre medición de flujos y parámetros relevantes.

4.1. Mallado

En esta sección se busca presentar los mallados empleados, así como sus parámetros de calidad junto con un resumen de los principales parámetros de cada geometría. En las figuras 4.2 y 4.1 se observan los mallados finales de las geometrías de *Sala de Alimentación* y de *Campana*, ambas obtenidas empleando el software ANSYS Meshing y Fluent.

Como se menciona brevemente en la sección 3.4, el proceso de mallado del caso de estudio se realiza de forma iterativa en pos de lograr simulaciones satisfactorias al emplear tanto ANSYS Fluent como OpenFOAM. Las figuras 4.2 y 4.1 corresponden a los mallados logrados luego de un proceso de conversión de topologías de elementos tetraedros a poliédricos, mejora necesaria realizada en pos de mejorar la estabilidad de la simulación y aminorar el uso computacional.

A continuación se presentan de forma resumida los valores de sus parámetros de calidad en las tablas 4.1 y 4.2. La totalidad de los parámetros de calidad de malla se encuentran en B.1.

Tabla 4.1: Calidad de malla seleccionada Campana

Parámetro	máximo/mínimo	$\bar{x}$	σ
Skewness	0.609	0.259	0.101
Razón de aspecto	4.419	1.890	0.374
Calidad Ortogonal	0.391	0.740	0.100
Elementos	2642895		
Nodos	484514		

Tabla 4.2: Calidad de malla seleccionada Sala de alimentación

Parámetro	máximo/mínimo	$\bar{x}$	σ
Skewness	0.600	0.245	0.100
Razón de aspecto	4.360	1.855	0.370
Calidad Ortogonal	0.405	0.755	0.099
Elementos	2155881		
Nodos	380256		

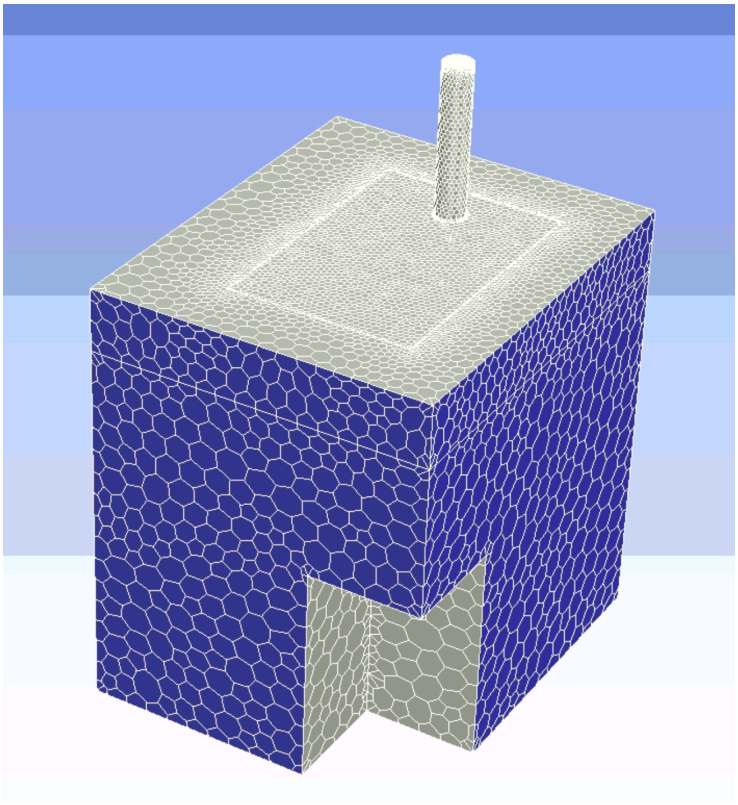


Figura 4.1: Mallado óptimo de Campana

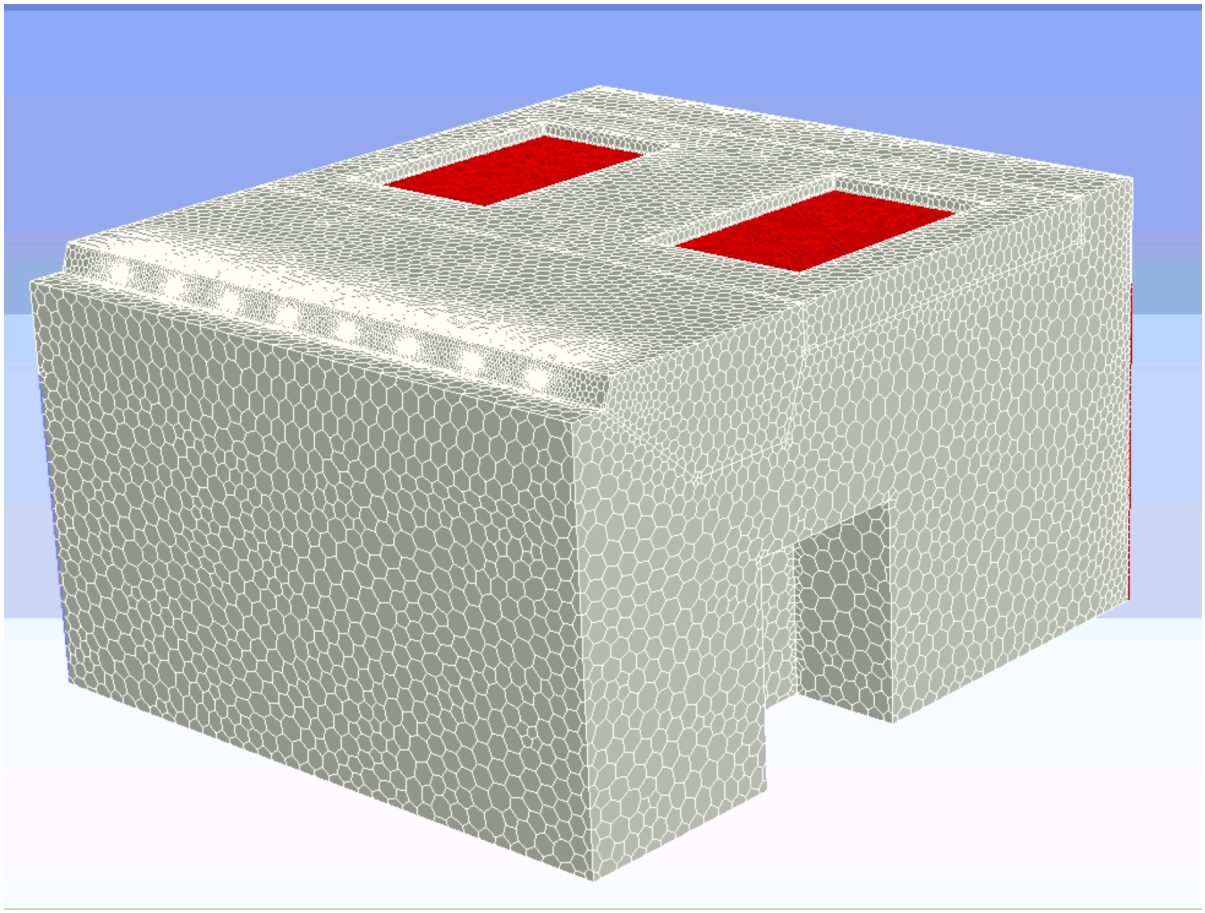


Figura 4.2: Mallado óptimo de Sala de Alimentación

4.2. Independencia de malla

Las mallas anteriormente exhibidas son tomadas como punto de inicio para la realización de los test de independencia de mallado descritos en la sección 3.4. Para cada geometría se realizan otros mallados procurando la menor desviación posible en los parámetros de calidad de malla. A continuación se muestran dichos parámetros para cada caso en su topología tetraédrica.

Tabla 4.3: Mallas campana

Malla	Elementos	Nodos	Skewness		Calidad Ortogonal	
			$\bar{x}$	max	$\bar{x}$	min
Base	2642895	484514	0.259	0.609	0.740	0.391
Fina	3579924	652869	0.258	0.707	0.741	0.293
Gruesa	996193	188073	0.260	0.853	0.739	0.147

Los resultados de realizar la prueba de independencia se aprecian en las figuras 4.4, 4.3 y 4.5. Aquí, se observan dos curvas para cada gráfico, ya que se realiza el análisis tanto para la malla con

elementos tetraédricos como para su símil con elementos poliédricos. El análisis sobre los valores de velocidad en cada caso han sido generados para cada malla descrita anteriormente, presentando no menores diferencias en los valores finales. Es evidente la gran diferencia en la cantidad de elementos de ambos mallados para cada geometría luego de aplicar la conversión de topología.

Tabla 4.4: Mallas Sala de Alimentación

Malla	Elementos	Nodos	Skewness		Calidad Ortogonal	
			$\bar{x}$	max	$\bar{x}$	min
Base	2155881	380256	0.245	0.600	0.755	0.405
Fina	5118808	890403	0.241	0.602	0.758	0.398
Gruesa	1675528	295499	0.248	0.621	0.751	0.378

Los valores de velocidad descritos a continuación son obtenidos luego de aplicar el siguiente procedimiento sobre ambos volúmenes de control:

- Campana: Inicialización de todas las variables en 0. Luego, 800 iteraciones en estado estacionario o hasta alcanzar criterio de convergencia. Finalmente, $0 - 1000$ iteraciones en estado transiente hasta lograr estabilidad en el parámetro de medición.
- Sala de Alimentación: Inicialización de todas las variables en 0. Luego, 500 iteraciones en estado estacionario y $0 - 1500$ iteraciones en transiente hasta lograr estabilidad en parámetro de medición.

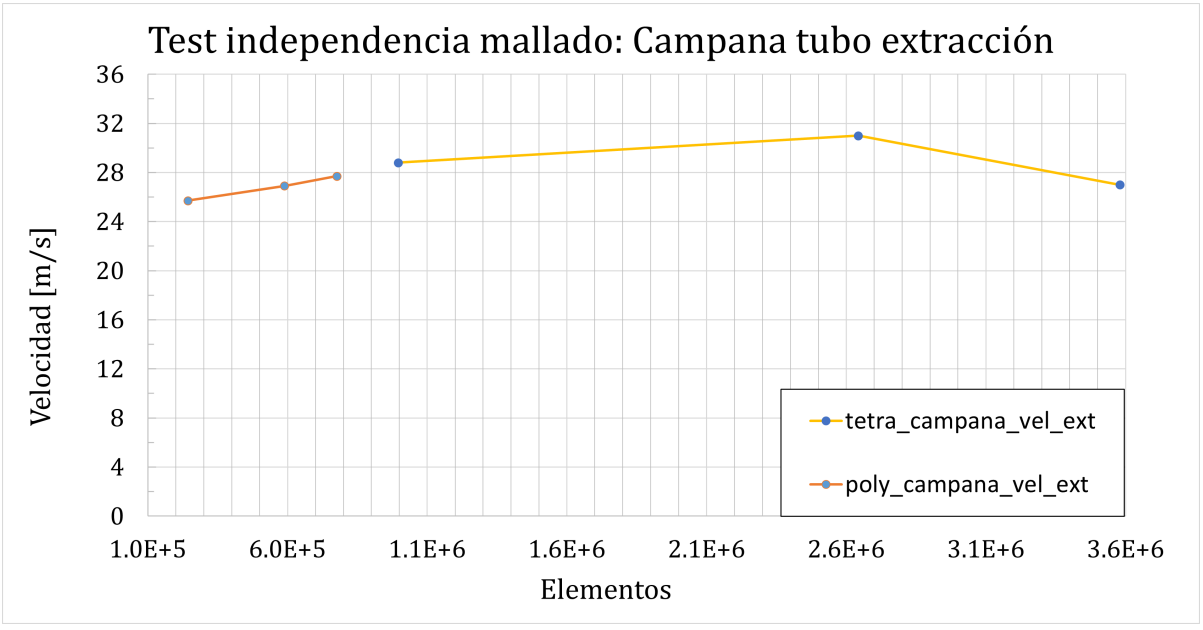


Figura 4.3: Velocidad para punto medio en tubo de extracción para malla poly y tetra

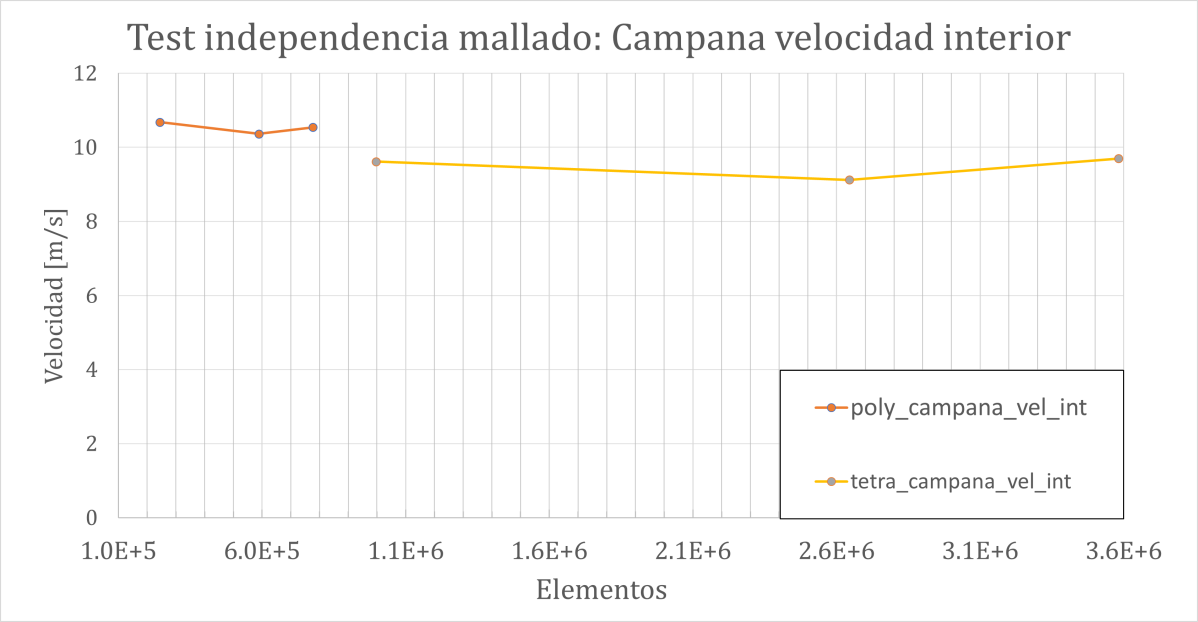


Figura 4.4: Velocidad para punto medio en campana para malla poly y tetra

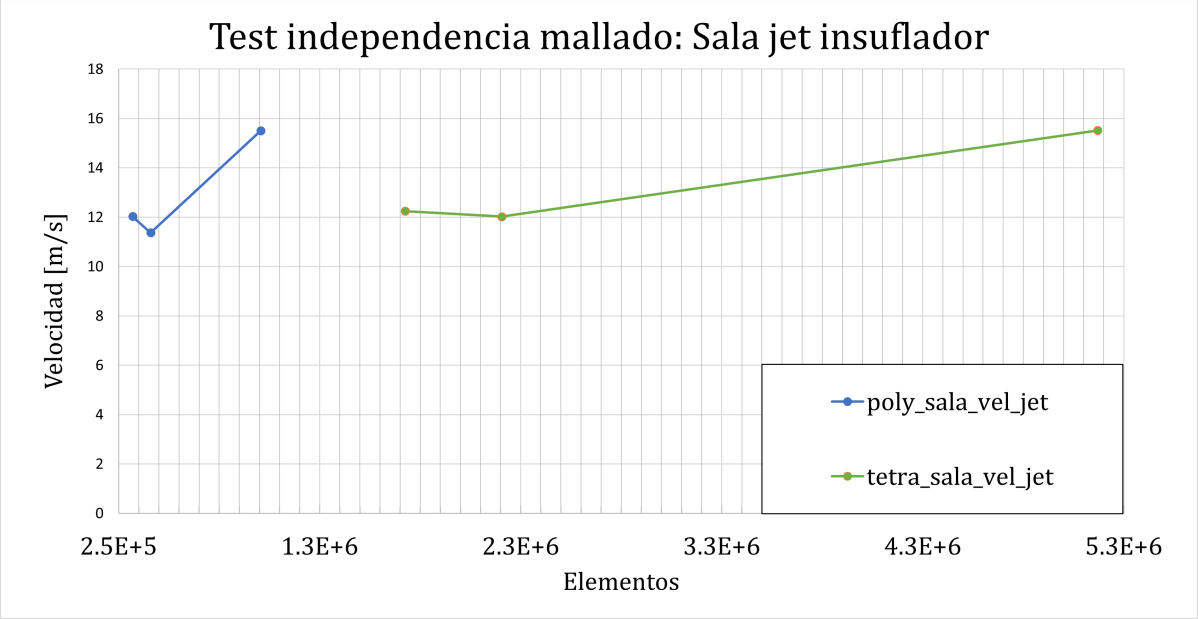


Figura 4.5: Velocidad para punto medio en SA para malla poly y tetra

En esta sección se presentan los mallados finales empleados en las simulaciones del caso de estudio. La estabilidad de la simulación, así como los métodos numéricos empleados en OpenFOAM fuerzan la obtención de mallas con parámetros excepcionalmente buenos según lo descrito en la sección 2.2.5. Luego de realizar un proceso sucesivo de conversión de mallado hacia dicha herramienta se decide por seleccionar aquellas mallas “Base” mostradas anteriormente, cuyos parámetros de calidad son posibles de encontrar en las tablas 4.4 y 4.3.

Para ambas geometrías se aplica una conversión de topología a partir de los mallados descritos y de forma independiente son sometidos a pruebas de independencia respecto a medición de velocidad en puntos característicos e idénticos para cada geometría. Para la geometría de *campana* se aprecian las mayores diferencias de valores entre las mallas de menor número de elementos al realizar la conversión de topología. Sin perjuicio de esto, es posible notar que para ambos puntos de medición presentes en esta geometría (Tubo extracción y velocidad interior) los valores medidos muestran pequeñas variaciones en las versiones de mallas con elementos poliédricos.

Los valores de velocidad calculados en la geometría *Sala de Alimentación* presentan una diferencia significativamente menor para cada punto entre las mallas con distintas topologías. Si bien los parámetros de malla dan cuenta de la óptima calidad de los elementos presentes en la selección escogida, el desempeño de las pruebas de independencia indican que el número de elementos del mallado condiciona los valores obtenidos si se aplica un mayor refinamiento. En conjunto, ambos resultados indican que la malla seleccionada si bien presenta buenos parámetros de calidad, son mejorables empleando un mayor número de elementos, por lo que el mallado para esta geometría presenta una incidencia en los valores de las futuras simulaciones.

4.3. Validación geometría extracción superior

En esta sección, se da cuenta de los resultados que cuantifican los cambios en ventilación realizados por la simplificación de las campanas de extracción. Se busca además, caracterizar las diferencias de extracción de aire en la campana al retirar la placa perforada como es mencionado en la sección 3.2. Para la realización de este análisis se emplea por comodidad solo ANSYS Fluent, ya que es un paso intermedio en el desarrollo del caso de estudio y su validación no incide en el análisis comparativo final.

Ante la simplificación de la geometría, es impuesta una condición de extracción uniforme de flujo sobre la superficie total que representa la placa, fenómeno similar al buscado mediante el diseño mismo de la campana de extracción. Para el análisis de extracción del fluido sobre la campana, se imponen las condiciones del caso de estudio (tabla 3.3) sobre el tubo de extracción mostrado en la figura 4.1. La imposición de estas condiciones otorga una velocidad de salida constante de 23.45 $[m/s]$.

En las figuras 4.6 y 4.7 se da cuenta de forma gráfica acerca de los valores de caudal másico por orificio como resultado del proceso de simulación (imagen referencial B.1). Se puede notar que existe una diferencia máxima del valor del flujo del orden de cinco veces para el orificio $(3, 4)$ en comparación con los orificios más alejados del punto de extracción. En la tabla B.4 se muestra el detalle numérico de los valores de flujo por cilindro.

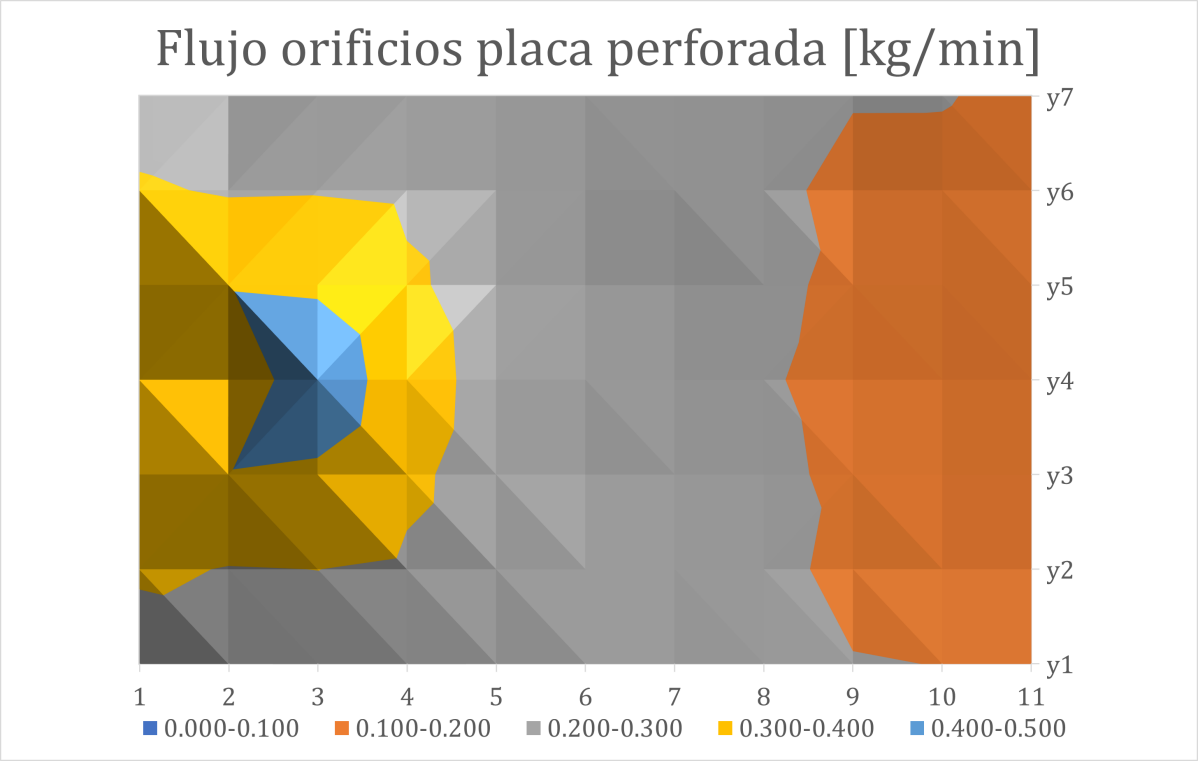


Figura 4.6: Flujo másico a través de placa perforada. Vista en planta

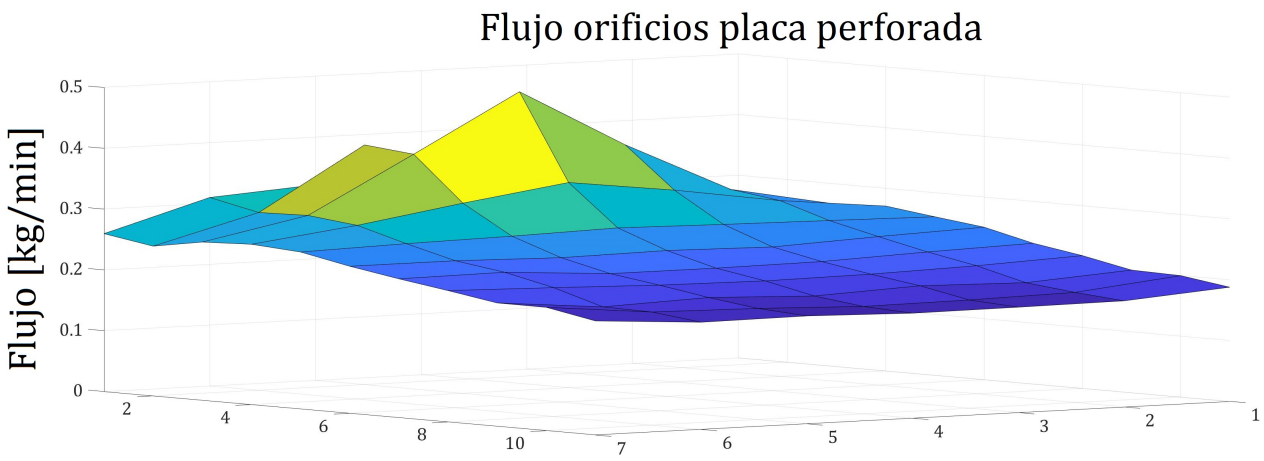


Figura 4.7: Flujo másico a través de placa perforada. Superficie

Si se realiza una subdivisión transversal a la placa perforada y se da cuenta de los valores de extracción promedio de las siete hileras de perforaciones, se encuentran los valores mostrados en la tabla 4.5. Aquí se aprecia que las diferencias de extracción promedio por hileras distan a lo más en un 12 %. Esta comparación expresa que si bien la extracción por orificio no es del todo homogénea, de forma global la campana si presenta un grado de extracción medianamente uniforme.

Tabla 4.5: Valores promedio de extracción según eje y

Eje	y1	y2	y3	y4	y5	y6	y7
Promedio [kg/min]	0.236	0.245	0.264	0.265	0.263	0.243	0.233
Desviación std	0.030	0.051	0.081	0.092	0.079	0.049	0.027

Dado que la simplificación del caso de estudio obedece a condiciones de tiempo de cálculo y los principales objetivos de este trabajo radican en el análisis de la resolución de la dinámica de flujo por parte de ambos programas, el grado de uniformidad en la extracción del flujo por parte de la eliminación de la placa perforada resulta ser aceptable.

Producto de este análisis se evidencia que de todas formas el diseño de extracción es perfectible en búsqueda de una extracción de flujo más uniforme. Se proponen dos posibles mejoras de diseño. En primera instancia se estima que un aumento de la altura de la campana, de forma que la sección de tranquilidad del fluido sea mayor ayudaría a una extracción más uniforme. Por otra parte disminuciones en los diámetros de los orificios cercanos al punto de extracción o el aumento del tamaño del tubo superior tendrían efectos similares.

4.4. Resultados Sala de Alimentación

En esta sección se detallan los resultados de las simulaciones realizadas sobre el caso de estudio de la Sala de Alimentación bajo la simplificación de extracción de flujo superior presentada en la sección anterior. La información planteada en esta sección es generada a partir de tres conjuntos de datos generados producto de las simulaciones Se incluye además un análisis de estabilidad de la simulación según el número de Courant máximo estipulado por OpenFOAM.

4.4.1. Estabilidad de simulación según Courant

De forma de comprobar la estabilidad de la simulación en virtud del número de Courant para aplicaciones en OpenFOAM, se realiza el análisis de sensibilidad basado en los estipulado en la sección 2.1.2. En la figura 4.8 se ha limitado el número de iteraciones para valores de Co máximo de 5 a un valor de 1500. El propósito de esta decisión es mantener la escala del gráfico dado que las pruebas realizadas muestran estabilidad de la simulación sobre las cinco mil iteraciones.

Este análisis es realizado en una etapa previa a las simulaciones finales, donde se busca dar cuenta de las capacidades de simulación de OpenFOAM para el caso de la Sala de Alimentación sin transferencia de calor y con placas perforadas en ambas campanas de extracción. Al ser una simulación en estado transiente se emplea el solver *pimpleFoam*.

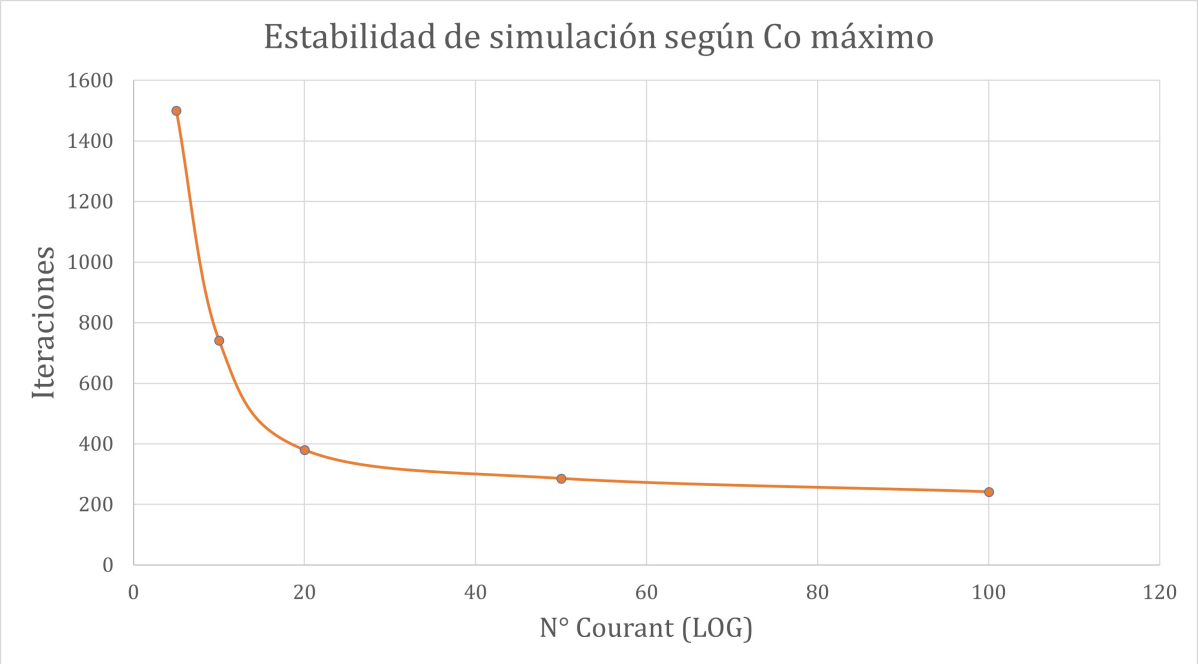


Figura 4.8: Estabilidad de simulación según Co máximo: pimpleFoam

Producto del análisis anterior es que se determina como buena práctica seguir las indicaciones del programa y no exceder el máximo número de Courant sugerido. Bajo esta hipótesis se obtiene el primer conjunto de datos finales a analizar en la siguiente sección. Sin embargo, el desarrollo del trabajo y las nuevas inestabilidades encontradas en simulaciones empleadas en los nuevos mallados en conjunto con la adición del fenómeno de transferencia de calor, fuerzan la implementación de esquemas de discretización más robustos y menores tolerancias sobre los algoritmos de resolución.

Lograda la estabilidad de las simulaciones en OpenFOAM bajo las condiciones de las simulaciones finales, surge la interrogante relacionada con el tiempo de cómputo de la solución. Mientras que OpenFOAM requiere de más de siete días de tiempo de cómputo para el desarrollo de 5 segundos de simulación, ANSYS Fluent realiza una tarea similar en 15 horas. Si bien ambas herramientas son empleadas en computadores distintos, la capacidad entre ellos no sostiene la diferencia en tiempo de cómputo, la explicación radica en la diferencia de los pasos de tiempo empleados en las simulaciones.

La tabla 4.6 recopila las diferencias entre ambas implementaciones (aquella que da origen a la figura 4.8 y la implementación final de OpenFOAM que a su vez es mostrada en detalle en el Anexo A). Las principales diferencias se dan en la restricción absoluta de tolerancias relativas, en la disminución de las tolerancias en la resolución de todos los parámetros y en la modificación de los esquemas numéricos asociados a los gradientes. La variación de parámetros es amplia, sin embargo se puede hacer principal alusión a los cambios de estabilidad a las tolerancias relativas y a los esquemas de discretización.

Tabla 4.6: Diferencias de implementación OpenFOAM para análisis de sensibilidad y simulaciones finales

Documento	Parámetro	Valor previo	Valor final
fvSolution	tolerancia p	1.00E-07	1.00E-08
	relTol p	0.01	0
	maxIter	500	1000
	Smoother	GaussSeidel	DICGaussSeidel
	tolerancia U, K, e	1E-05	1E-08
	relTol U, K, e	0.01	0
	maxIter U, K, e	500	1000
	Smoother	GaussSeidel	DICGaussSeidel
fvSchemes	gradSchemes	cellLimited Gauss linear 0.5	cellMDLimited leastSquares 1
	snGradSchemes	limited 1	limited corrected 1

Si bien no se modifica la tolerancia final relativa a cada parámetro (Ej: *pFinal*), la restricción de tolerancia en las iteraciones anteriores a la final considera una baja significativa en la oscilación de la solución, siendo el costo de esta implementación el aumento del tiempo de computo del solver.

Por otra parte, en la sección C.2.1.4 del manual adjunto a este informe se encuentra la tabla C.3 que diferencia brevemente ambos esquemas. Son dos las principales diferencias en ambas formulaciones. La primera consiste en que *leastSquares* empleará el método de mínimos cuadrados para ponderar valores. La adición del término *MD* hará referencia a que el proceso de cálculo incluye todos los valores de las celdas circundantes. Dicho esto, el método de discretización empleado en los casos de estudio posteriores es claramente más robusto ante perturbaciones mayores en el desarrollo cálculo.

Por otra parte, la configuración del paso de tiempo en el software ANSYS Fluent para solver en base al cálculo de presión carece de limitante del número de Courant. Si bien se aconsejan rangos de valores mediante el manual (discutidos en la sección 2.1.2) no existe un valor máximo en el paso de tiempo cuando la opción de *paso de tiempo variable* es seleccionada en el programa. En la práctica luego de analizar la oscilación del paso de tiempo variable se opta por fijar este parámetro a un valor mínimo que garantice la rápida convergencia en cada iteración. Se establece un paso de tiempo igual a 0.001.

Con este paso de tiempo se realizan las últimas dos simulaciones finales para el caso de estudio. Tanto OpenFOAM como ANSYS Fluent demoran un total de 22 y 13 horas en completar 10 segundos de simulación respectivamente.

4.4.2. Resultados Sala de Alimentación - Transferencia de calor

Se presentan a continuación resultados concernientes a mediciones de flujo, velocidad promedio y perfiles de temperatura obtenidos a partir de simulaciones comparables bajo un mismo mallado en ambos programas. En la tabla 4.7 se resumen las tres simulaciones que dan paso a estos resultados. En el caso de la simulación I se obtienen cinco segundos de simulación con un paso de tiempo promedio de $3.69e-05$ a causa de un número de Courant máximo de 5. Los equipos utilizados para realizar todas las simulaciones se detallan en la tabla anexada B.1.

Tabla 4.7: Resumen tiempos de simulación SA

Simulación	Programa	Iteraciones estacionario	Tiempo transiente (s)	Co max	Paso de tiempo (s)
I	OpenFOAM	3000	5	5	3.6E-05
II	OpenFOAM	3000	10	135.5	1.0E-03
III	ANSYS Fluent	3000	10	135.5	1.0E-3

Dadas las condiciones de borde del caso de estudio, la naturaleza del problema es transiente y por ende no se logra la convergencia bajo una implementación estacionaria. Sin embargo, se hace uso de esta implementación para lograr un desarrollo del flujo de aire dentro de la sala previo a la simulación en estado transiente. En OpenFOAM, la simulación del caso de estudio en estado estacionario sobre las 3500 iteraciones desarrolla anormalidades en el flujo e inestabilidades en la simulación. Es por esto que luego de un análisis retrospectivo, se determinan 3000 iteraciones como un buen punto inicial para la simulación en estado transiente. Con fines comparativos, se mantiene este número de iteraciones para todas las simulaciones en ambos software.

Para el análisis de resultados se hace un registro de los parámetros de magnitud de velocidad y temperatura en todo el transcurso de las simulaciones. Los puntos geométricos dentro de la sala son detallados en la tabla 4.8 y son mostrados en la figura 4.9 a continuación. Se toman puntos cercanos a las condiciones de borde de *Cortina (co)* y *Placa 1 (P1)*. Adicionalmente se adiciona el punto medio de la sala *midpoint* y uno cercano a uno de los muebles de la sala lejano al ingreso de aire lateral y a las fuentes de calor (*P2*).

Tabla 4.8: Localización de puntos de muestreo

Punto/coordenada	X [m]	Y [m]	Z [m]
midpoint	2.275	2.585	1.425
p1	0.7	2.875	1.300
p2	3.5	2.325	1.250
co	1	1.060	1.500

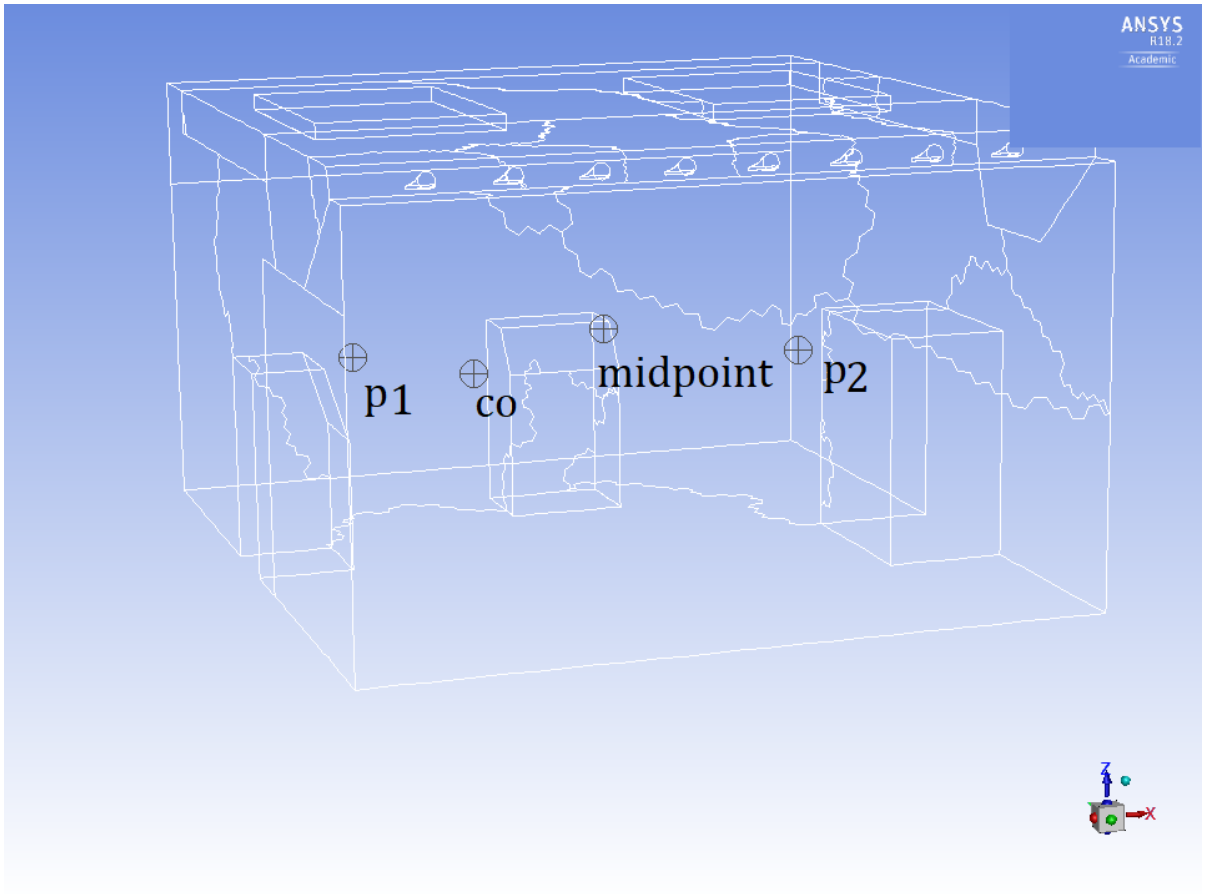


Figura 4.9: Ubicación de puntos de muestreo Sala de Alimentación

4.4.3. Estado estacionario

Si bien las simulaciones realizadas en estado estacionario son un punto de inicio para el desarrollo posterior del proceso de simulación, otorgan información relevante acerca del desempeño de ambos programas. En este sentido, los valores generados en esta instancia reflejan el empleo del algoritmo SIMPLEC y su manejo a través del proceso de simulación.

En las tablas 4.9 y 4.10 se resumen los valores de temperatura y velocidad promedio en estado estacionario para los puntos de medición al interior de la sala. La curva de datos a lo largo de las iteraciones se incluye anexada en las figuras B.4 - B.3. En cuanto a los valores de temperatura, en la tabla 4.9 se aprecian leves diferencias en cuanto a los valores promedio, se encuentran todos dentro de un acotado rango de valores. Se puede notar que OpenFOAM presenta temperaturas promedio menores en la mayoría de los puntos de muestreo.

Ambos programas muestran valores mínimos similares, Fluent es quien presenta valores levemente por debajo de las condiciones de borde e iniciales del caso de estudio. En cuanto a los máximos se puede ver que dependerá el punto a analizar quien tendrá el máximo valor. En este

sentido, OpenFOAM presenta el máximo de temperatura global para el punto p1, llegando a 300.38 K. Sin embargo y como se muestra en la figura B.2, valores de esta magnitud se presentan una única vez al inicio de la simulación y no presenta un rango de valores alcanzados en el desarrollo posterior de la simulación.

Ambos programas presentan una mayor desviación estándar de datos en el punto *p1*, en concordancia con su posición cercana a una fuente de calor. Respecto a este mismo punto, de forma global los resultados obtenidos mediante Fluent presentan una mayor oscilación de valores, en contrastarse con la simulación realizada en OpenFOAM.

Tabla 4.9: Valores de temperatura estado estacionario SA

Simulación	Punto	Promedio [K]	Desviación Std	Min [K]	Max [K]
I y II	midpoint	293.34	0.09	293.00	293.56
	p1	293.79	0.43	293.08	300.38
	p2	293.32	0.09	293.00	293.56
	co	293.15	0.10	293.00	293.59
III	midpoint	293.52	0.13	292.99	294.07
	p1	293.87	0.52	292.93	295.46
	p2	293.58	0.08	293.02	294.02
	co	293.09	0.10	292.97	293.59

Tanto en OpenFOAM como en ANSYS Fluent, los valores de velocidad presentan gran oscilación sin un patrón aparente (figuras B.4 y B.5). En la tabla 4.10 se encuentra un resumen de valores de la magnitud de velocidad para cada punto de muestreo. Aquí se puede apreciar una mayor diferencia entre los valores calculados por ambos software. La mayor diferencia de velocidad promedio se encuentra en el punto *co* donde OpenFOAM alcanza el doble del valor simulado por ANSYS Fluent.

En el punto *p1* cercano a una fuente de calor, se aprecia que OpenFOAM presenta una mayor magnitud promedio, siendo que según lo revisado anteriormente, este punto estaría con más temperatura que su par calculado empleando ANSYS Fluent.

Por otra parte, tanto la dispersión de datos como los valores máximos alcanzados en ambas simulaciones vienen de la mano del software de código abierto. Las velocidades máximas alcanzan el doble de su símil en tres de los cuatro puntos de análisis.

Como se ha mencionado anteriormente, la simulación I presenta una diferencia sustancial en el paso de tiempo respecto a las simulaciones II y III. En la figura 4.10 se muestran los valores de temperatura de los puntos *p1* y *co* para las simulaciones realizadas en OpenFOAM. Como

Tabla 4.10: Valores de velocidad estado estacionario SA

Simulación	Punto	Promedio [m/s]	Desviación Std	Min [m/s]	Max [m/s]
I y II	midpoint	0.82	0.67	0.10	5.39
	p1	1.30	0.62	0.11	2.73
	p2	1.53	0.99	0.02	5.04
	co	1.44	0.98	0.13	6.86
III	midpoint	0.69	0.30	0.13	2.47
	p1	0.74	0.46	0.08	2.90
	p2	1.36	0.33	0.13	2.16
	co	0.72	0.18	0.18	1.36

aquí se puede apreciar, los valores obtenidos para todos los puntos distan sustancialmente entre la simulación empleando un máximo número de Courant igual a 5 y aquella que no.

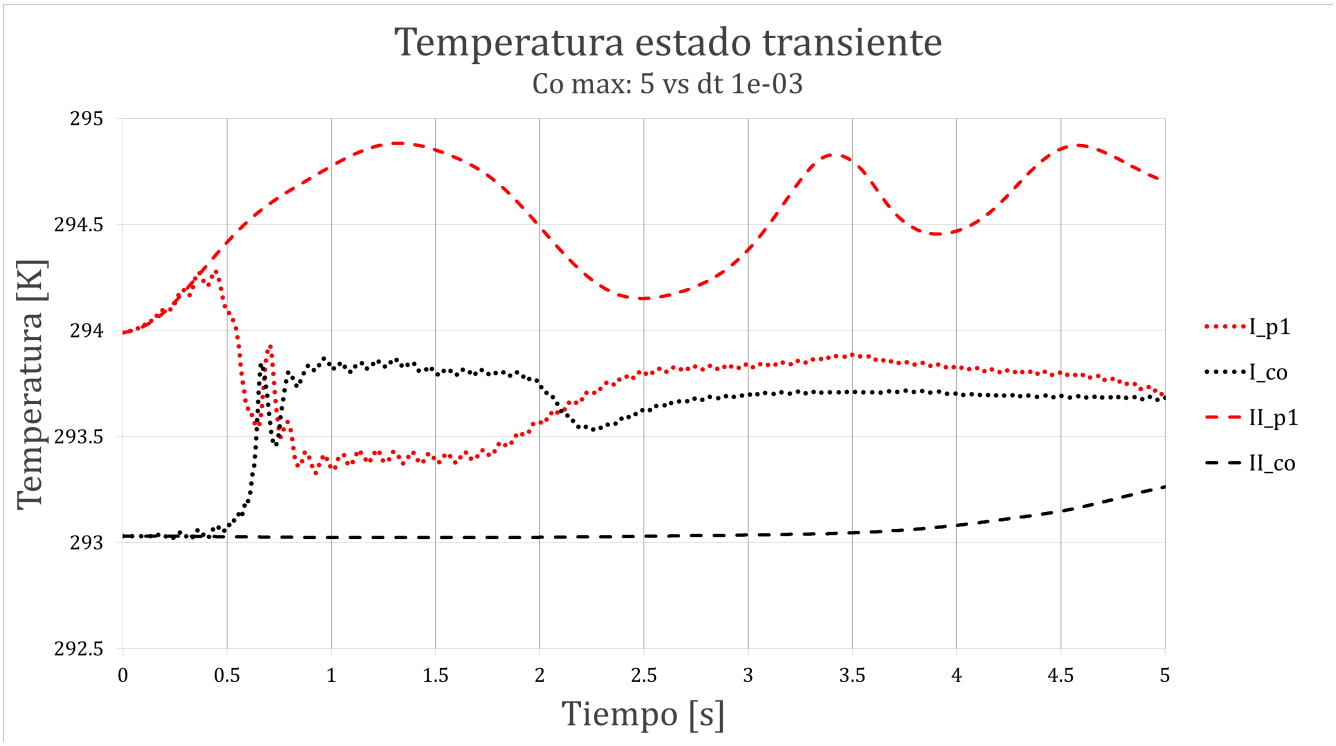


Figura 4.10: Temperatura simulaciones I y II: Puntos co y p1

En las tablas 4.11 y 4.12 se da cuenta de cuan dispares son los valores de la simulación I comparados con II y III. En el caso de la temperatura como se visualiza en la figura 4.10, los valores están acotados de forma envolvente por las temperaturas simuladas bajo paso de tiempo mayor. Los valores de temperatura promedio son bastante similares a aquellos calculados a partir de 10 segundos de tiempo de simulación.

Tabla 4.11: Valores de temperatura estado transiente SA

Simulación	Punto	Promedio [K]	Desviación Std	Min [K]	Max [K]
I	midpoint	293.68	0.13	293.34	294.03
	p1	293.74	0.22	293.33	294.28
	p2	293.66	0.10	293.41	294.08
	co	293.63	0.23	293.02	293.87
II	midpoint	293.39	0.02	293.34	293.43
	p1	294.33	0.43	293.59	294.88
	p2	293.42	0.02	293.36	293.44
	co	293.18	0.14	293.02	293.43
III	midpoint	293.62	0.05	293.51	293.68
	p1	294.01	0.51	293.29	294.84
	p2	293.67	0.01	293.64	293.69
	co	293.18	0.09	293.02	293.36

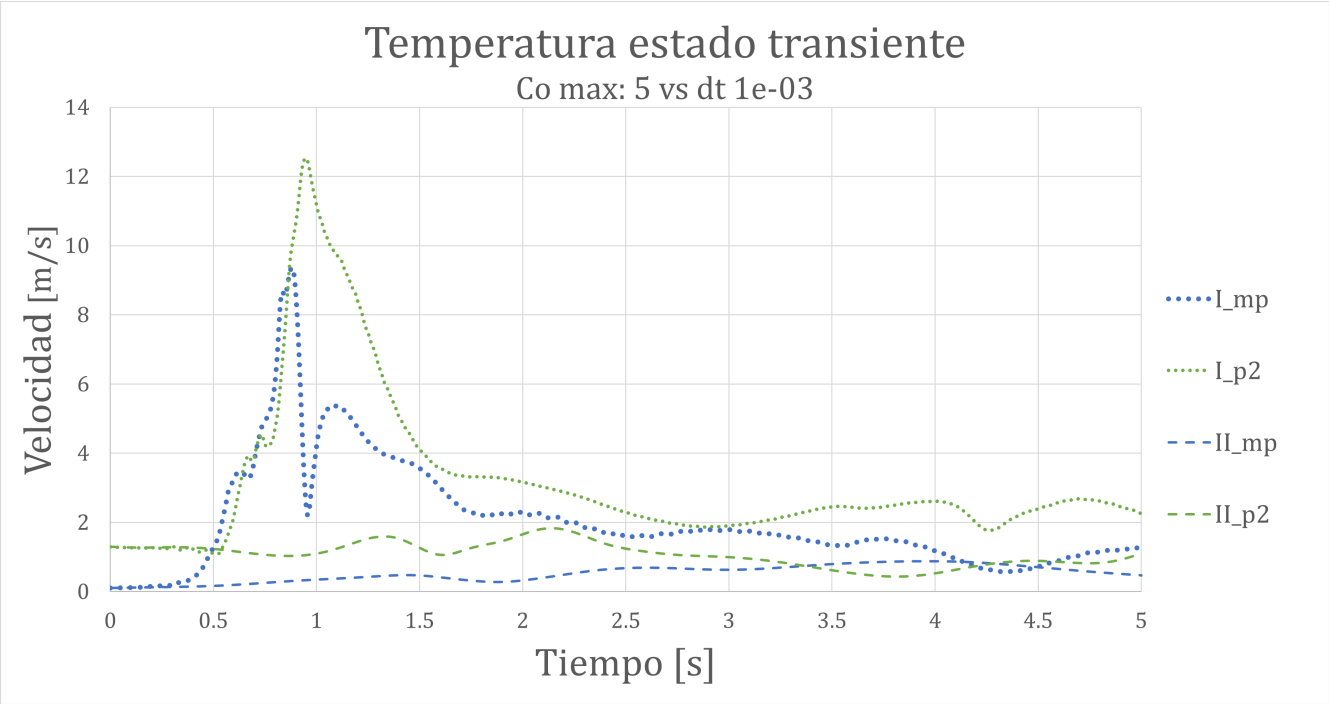


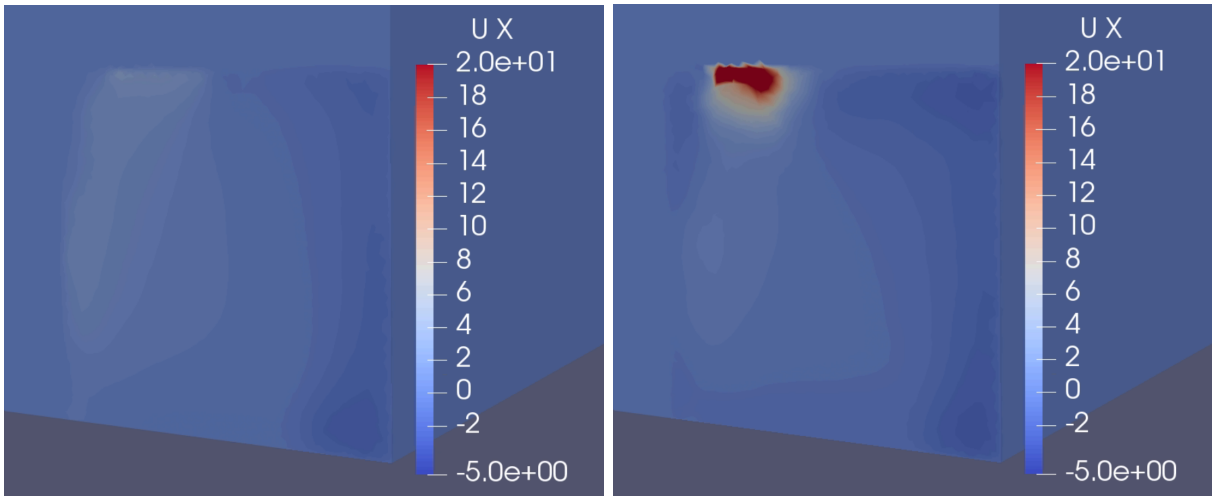
Figura 4.11: Velocidad simulaciones I y II: Puntos midpoint y p2

Las velocidades presentan diferencias aún más considerables entre simulaciones. En la figura 4.11 se aprecia como para el caso I, en los tres primeros segundos se encuentran máximos de velocidad desproporcionados y que superan en más de 6 veces los valores de la velocidad para los puntos $p2$ y mp en comparación con su símil en la simulación II. El punto $p1$ es quien tiene una mayor diferencia, alcanzando un máximo de 43 $[m/s]$, se nuestra junto con las evoluciones temporales de los otros puntos en la figura B.6.

La simulación I luego de tener un inicio inesperado en los valores de velocidad, se puede ver que los valores se estabilizan. Por el tiempo de simulación generado no es posible evidenciar si esta estabilidad es duradera y tampoco si los valores se asemejan en el largo plazo a aquellos de las simulaciones II y III.

Numéricamente, las diferencias en los resultados de velocidad dan cuenta del nulo propósito de comparación entre las simulaciones I con II y III. En I el rango de valores excede incluso la magnitud de la condición de borde impuesta sobre los insufladores de aire, puntos que debiesen obtener la mayor velocidad dentro del caso de estudio. Los máximos de velocidad se ven reflejados en cada punto de monitoreo e influye notoriamente en el promedio de valores. Estas velocidades están lejos de los valores generados en las simulaciones realizadas bajo un paso de tiempo constante, tanto en OpenFOAM como en ANSYS Fluent. A continuación se muestra una serie de imágenes que busca dar cuenta de lo sucedido en la simulación I.

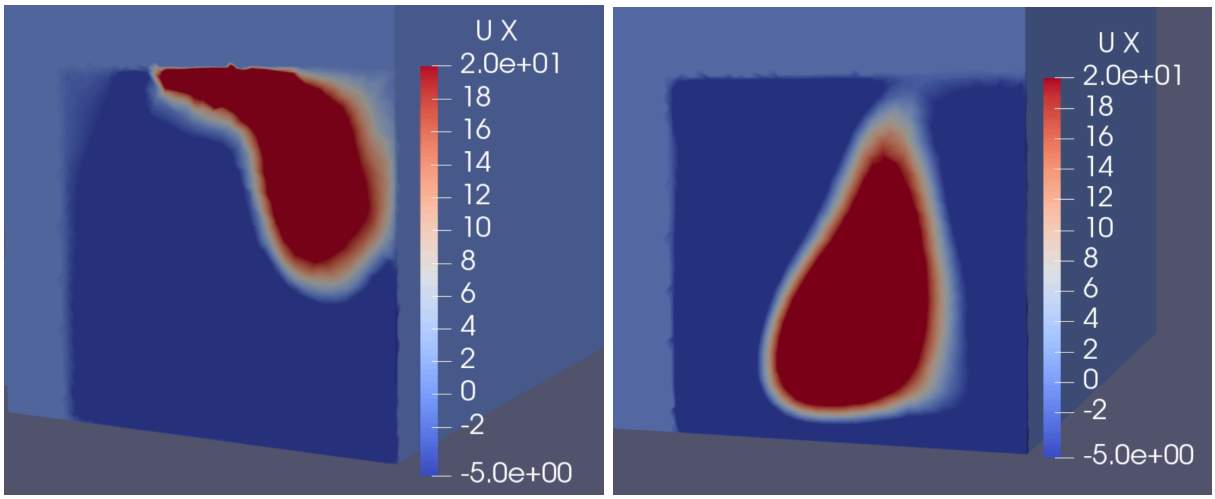
Como se menciona al inicio de esta sección, las simulaciones I y II comparten un mismo punto de partida para el desarrollo transiente. Como se puede apreciar en la figura 4.12 (a), las condiciones de entrada de flujo a través de la cortina lateral son perfectamente normales. Este estado cambia bruscamente al llegar a los 0.4 segundos, donde se aprecia un considerable aumento de velocidad originado en el borde superior de la cortina lateral (figura 4.12 (b))



(a) Tiempo: 3000 it. (b) Tiempo: 0.4 s

Figura 4.12: Anormalidad simulación I 1/3

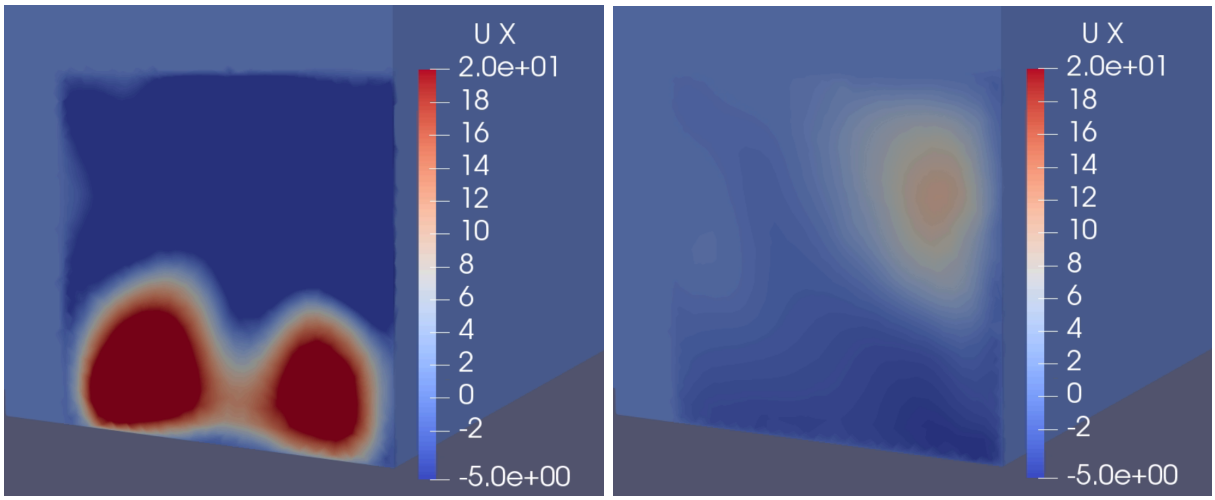
La inestabilidad mostrada en la figura anterior se traslada desde el borde superior de la cortina hacia la zona media de esta como se aprecia en la figura 4.13 (a). Es en este punto donde las altas velocidades de ingreso de fluido generan los valores inesperados en los puntos de muestreo colocados en la sala.



(a) Tiempo: 0.6 s (b) Tiempo: 0.7 s

Figura 4.13: Anormalidad simulación I 2/3

A medida que se sigue desarrollando la simulación, la magnitud de los valores de entrada tiende a disminuir conforme descienden las máximas velocidades hacia el inferior de la sala. En la figura 4.14 (a) se aprecia como el fenómeno generado en el segundo 0.4 se termina por dividir hacia las esquinas inferiores de la cortina. Tal como se aprecia en el gráfico de velocidades 4.11 en el segundo 3 el efecto anómalo del inicio se ha estabilizado. La figura 4.14 (b) presenta valores de velocidad de entrada para la sala dentro de un rango admisible y sin señales de nuevas inestabilidades.



(a) Tiempo: 0.9 s (b) Tiempo: 3 s

Figura 4.14: Anormalidad simulación I 3/3

Tanto la simulación I como II son realizadas empleando los mismos esquemas numéricos y parámetros de estabilidad y control de la simulación. La razón de anormalidad se debe al manejo

de la simulación en estado transiente empleando un paso de tiempo de $3.69E-05$. Esta escala de tiempo es significativamente inferior a los $1e-03$ segundos del paso de tiempo de las simulaciones II y III. El origen de la anormalidad puede deberse a la combinación de un paso de tiempo pequeño junto con una combinación de esquemas incapaces de controlar sucesivos máximos locales.

La simulación I es prueba de que el conjunto de parámetros y esquemas seleccionado es capaz de manejar la inestabilidad generada al inicio de la simulación, mas no impedir su aparición. Se puede ver en el transcurso de I que el sistema se estabiliza en cuanto a valores esperados, sin embargo se requiere de un mayor tiempo se simulación para realizar aseveraciones de estabilidad en el tiempo.

Ya que el número de Courant es un parámetro de estabilidad de la simulación y siendo 5 el valor recomendado por OpenFOAM y por Fluent para el inicio de las simulaciones de formulación transiente, queda propuesto un análisis de sensibilidad respecto al valor de Courant máximo escogido y el consecuente paso de tiempo de la simulación I. Es innegable que en este caso la elección de este parámetro ha desarrollado anomalías que, una vez violada la sugerencia, no se han presentado en el caso de la simulación II. Un análisis de sensibilidad otorgaría la información necesaria para determinar si en este caso se encuentra el punto justo que genera anomalías o estas se generarán de todas formas en el empleo de un número de Courant por ejemplo igual a 1 o 10.

A partir de este punto se realiza un análisis de las simulaciones II y III, tanto para temperaturas como velocidades simuladas al interior de la geometría de estudio.

4.4.4. Estado transiente

A partir de la visualización de las temperaturas para los puntos de muestreo se obtienen las figuras 4.15 y 4.16. En la primera se observa como la evolución temporal de temperaturas de los puntos *midpoint (mp)*, *p2* y *co* se asemejan bastante. En los dos primeros existe un claro desfase de 0.3 grados para los tiempos simulados. mientras que las curvas del punto *co* presentan evoluciones temporales similares en cuanto a magnitud y tiempo, lo que corrobora de forma visual el valor promedio de temperaturas para ambas curvas de 293.18 K mostrado en la tabla 4.11.

Para *midpoint* y *p2* se obtienen leves diferencias en cuanto a magnitud promedio, sin embargo se puede ver que las temperaturas simuladas en ANSYS Fluent tienden a decaer en el transcurso del tiempo de simulación, mientras que para para sus símil en OpenFOAM, los valores se mantienen oscilando levemente en torno a los 293.4 K.

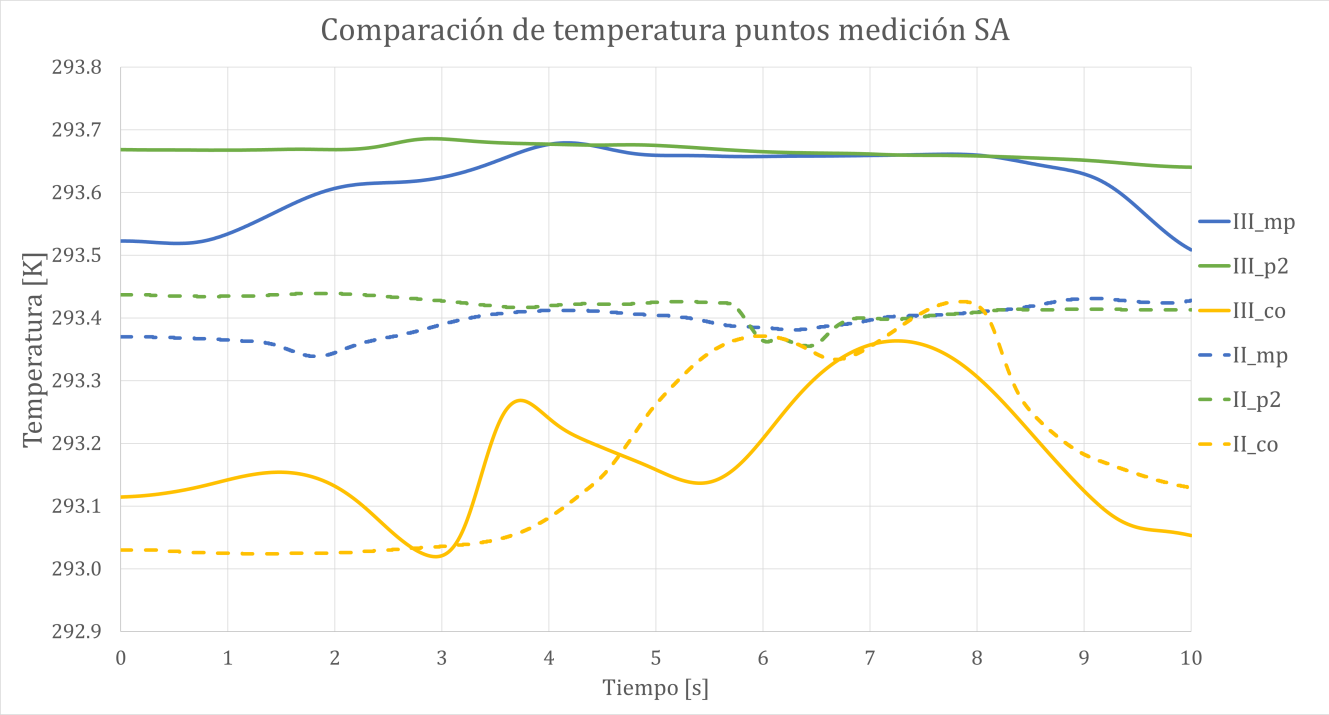


Figura 4.15: Temperaturas a lo largo del tiempo para AF y OF.

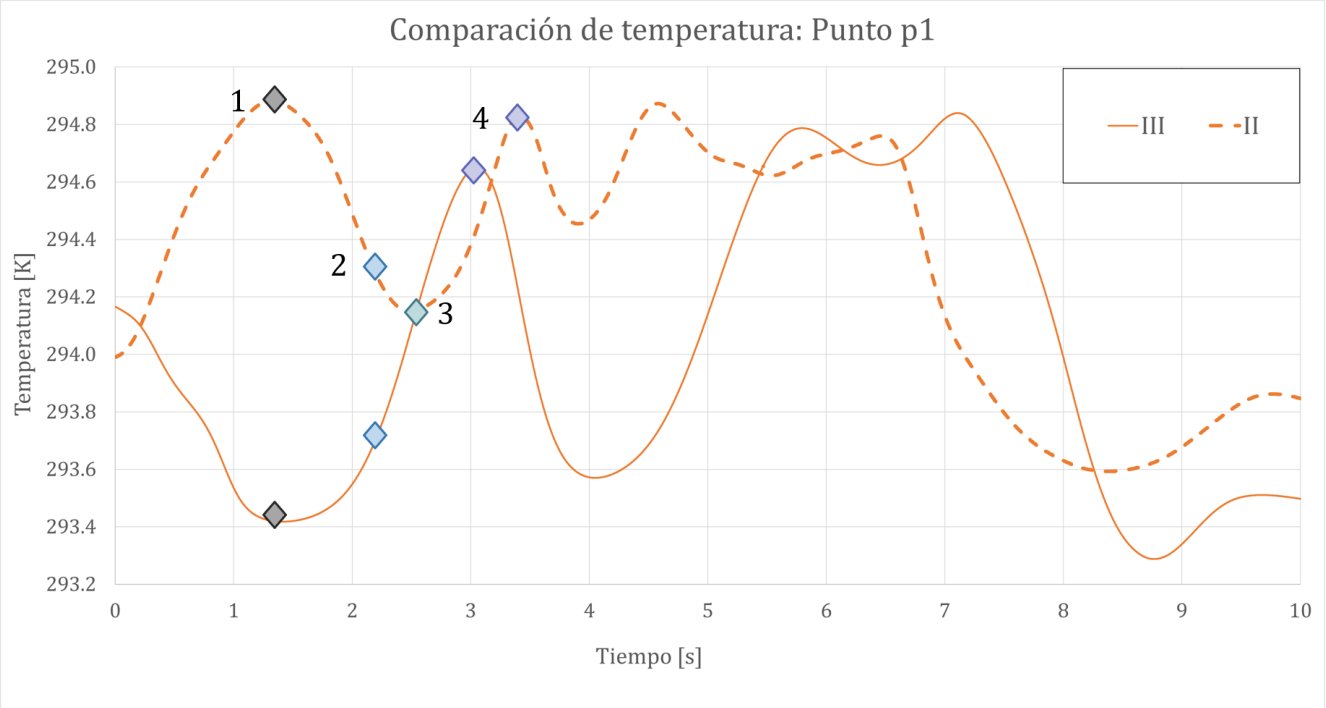
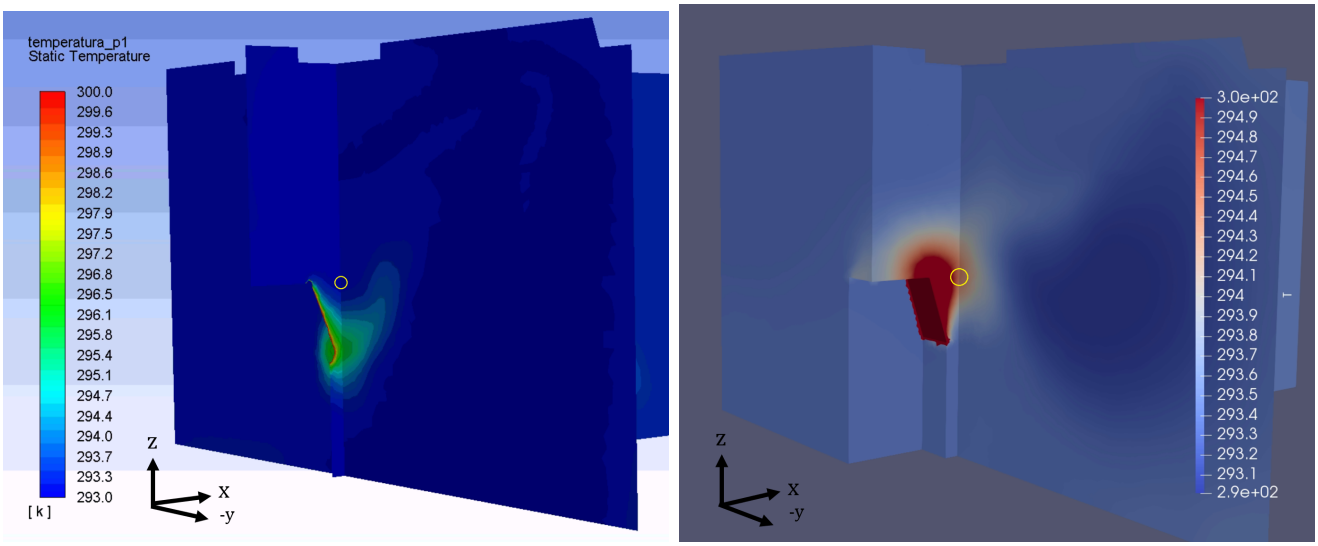


Figura 4.16: Temperaturas punto p1 a lo largo del tiempo para AF y OF.

En la figura 4.16 se muestra el desarrollo de la temperatura para el punto de muestreo *p1*. Aquí se puede apreciar que en un comienzo de la formulación transiente existe una diferencia notoria en el comportamiento de los valores, pero a medida que transcurre el tiempo de simulación, ambas curvas se asemejan considerablemente. A modo de entender la dinámica del flujo detrás de esta diferencia, se realiza a continuación una muestra gráfica de la evolución temporal de temperatura en ambas simulaciones. Para esto se han tomado los puntos temporales $1 - 4$ mostrados sobre las curvas de ambos software en la figura ya mencionada.

El punto *p1* (mostrado en las imágenes por un círculo claro) se encuentra localizado justo en el vértice de las secciones mostradas a lo largo de la serie de imágenes y a la altura superior de la placa emisora de calor. Como se muestra en las figuras 4.17 (a) y 4.18 (a), la masa de aire caliente cercano a la placa realiza un movimiento ascendente, cuya evolución temporal de temperatura luego se ve igualada a los mostrado por OpenFOAM en las figuras 4.19 (b) y 4.20 (b).

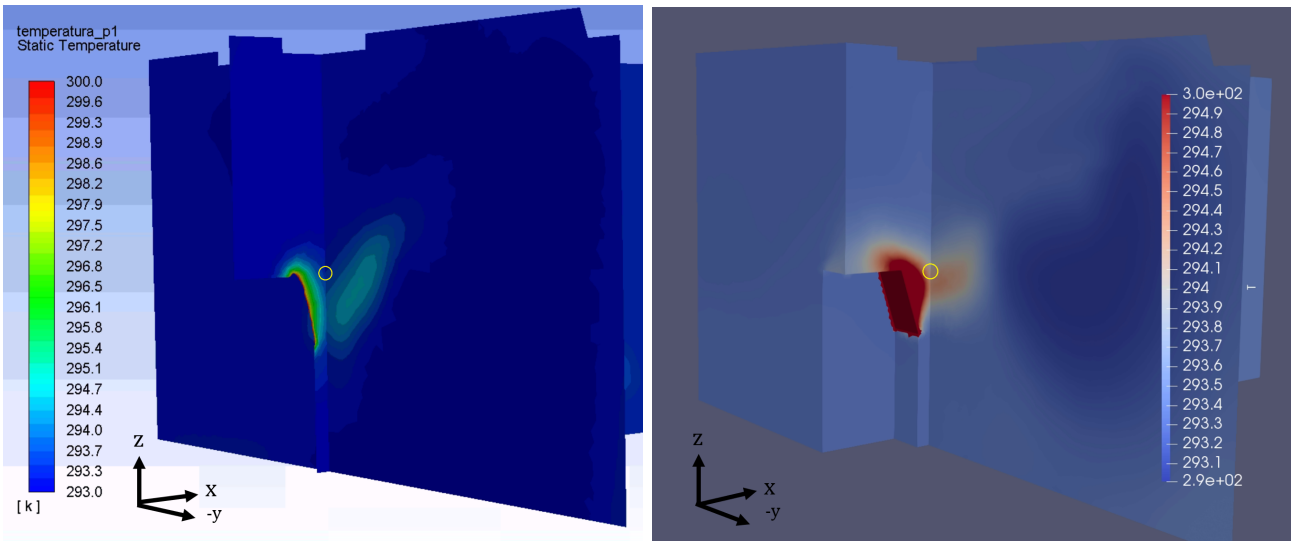


(a) ANSYS Fluent: 1 (s)

(b) OpenFOAM: 1 (s)

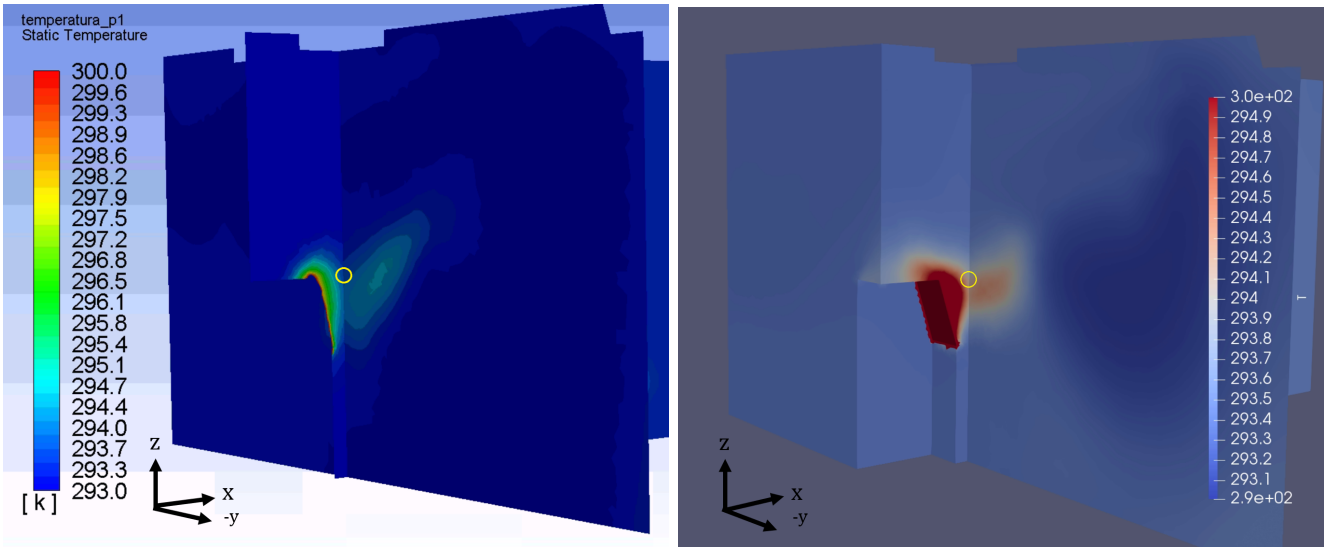
Figura 4.17: SA: Evolución temporal de temperatura 1/4

Otro aspecto en común en el desarrollo de la transferencia de calor en el medio, es la evolución temporal de temperatura en el plano normal al eje x. El color azul intenso de los planos de corte en ANSYS Fluent permite notar la evolución temporal de temperatura en la dirección $y-$, dando cuenta de la elevación de temperatura del área circundante a la placa. Este efecto de propagación también es visible en la información entregada por OpenFOAM. Se ha demarcado en color naranjo el contorno que de colores en ambas imágenes de la figura 4.20 a modo de ilustración.



(a) ANSYS Fluent: 2.3 (s) (b) OpenFOAM: 2.3 (s)

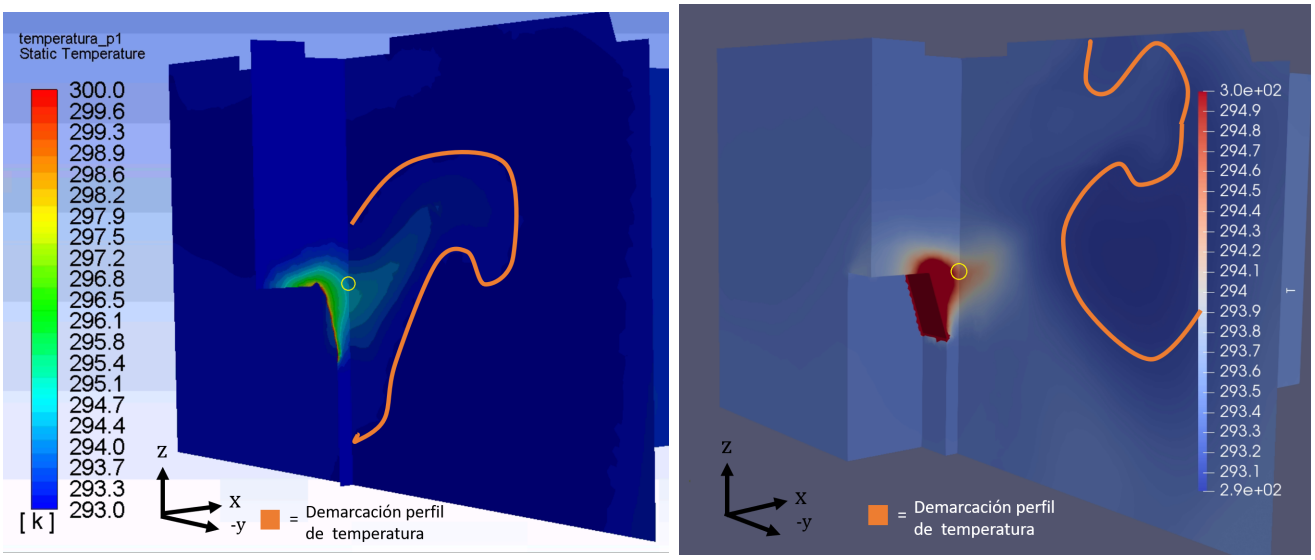
Figura 4.18: SA: Evolución temporal de temperatura 2/4



(a) ANSYS Fluent: 2.5 (s) (b) OpenFOAM: 2.5 (s)

Figura 4.19: SA: Evolución temporal de temperatura 3/4

La información provista por la figura 4.16 junto con lo visualizado en la sucesión de imágenes, es posible determinar que la diferencia inicial mostrada en los primeros 2.5 segundos de simulación (curvas cóncava en AF y convexa en OF) son producto de la dinámica del flujo presentada en el instante cero. Se aprecia claramente como la pendiente positiva en el instante cero induce una curva convexa en OpenFOAM, mientras que sucede lo contrario para el caso de ANSYS Fluent. A medida que se desarrollan ambas simulaciones, los comportamientos de la transferencia de calor en la zona se equiparan para luego seguir de forma muy similar.



(a) ANSYS Fluent: 3 (s)

(b) OpenFOAM: 3.4 (s)

Figura 4.20: SA: Evolución temporal de temperatura 4/4

El comportamiento mostrado por la curva de temperaturas del punto *p1* no es extrapolable por completo al comportamiento que presentan los otros puntos de muestreo, sin embargo, de acuerdo al desarrollo de tanto los perfiles de temperatura como los valores numéricos de esta muestran gran similitud en su forma.

Para realizar aseveraciones más precisas de acuerdo de la semejanza por completo de las curvas o de los valores promedio de las simulaciones, es necesario un tiempo de simulación más prolongado al aquí mostrado. Sin perjuicio de esto, el análisis muestra una clara consistencia en los valores obtenidos, presentándose leves diferencias que llevadas a una escala mayor en el rango de temperaturas, podrían carecer de relevancia.

4.4.5. Velocidad transiente

Los valores de los puntos de muestreo para la magnitud de velocidad se encuentran en la tabla 4.12. Aquí se puede apreciar nuevamente las diferencias de valores de magnitud de velocidad de forma resumida. Se puede ver nuevamente que OpenFOAM mediante la simulación II presenta valores promedio levemente mayores a los obtenidos por su símil en la simulación III. Además de esto, el programa de código abierto presenta una mayor dispersión de datos respecto al promedio para tres de los cuatro puntos muestreados. En tanto al rango de valores, OpenFOAM presenta un intervalo mayor para los valores de magnitud de velocidad, hecho que queda en evidencia en los gráficos de velocidad para los puntos muestreados (figuras 4.21, 4.22 y B.7).

Tabla 4.12: Valores de velocidad estado transiente SA

Simulación	Punto	Promedio [m/s]	Desviación Std	Min [m/s]	Max [m/s]
I	midpoint	2.04	1.59	0.10	9.36
	p1	2.55	1.42	0.98	7.95
	p2	3.20	2.29	1.10	12.53
	co	3.03	4.92	0.19	43.09
II	midpoint	0.54	0.22	0.10	1.03
	p1	0.87	0.54	0.18	1.73
	p2	1.37	0.95	0.19	4.72
	co	0.92	0.39	0.27	1.91
III	midpoint	0.68	0.33	0.17	1.34
	p1	0.71	0.10	0.53	0.89
	p2	1.08	0.54	0.36	2.23
	co	1.12	0.22	0.86	1.77

Las curvas que describen los valores de velocidad en el transcurso de las simulaciones II y III distan bastante de lo visto en el análisis de temperatura. Aquí, se puede ver como para cada punto se presentan comportamientos similares pero sin una correlación clara. La principal diferencia en las similitudes de las curvas puede deberse a la naturaleza vectorial de la velocidad. En las figuras 4.21 y 4.22 se aprecian la magnitud de los valores, mas no el sentido que pueda tener el flujo ni las contribuciones de las distintas componentes de la velocidad.

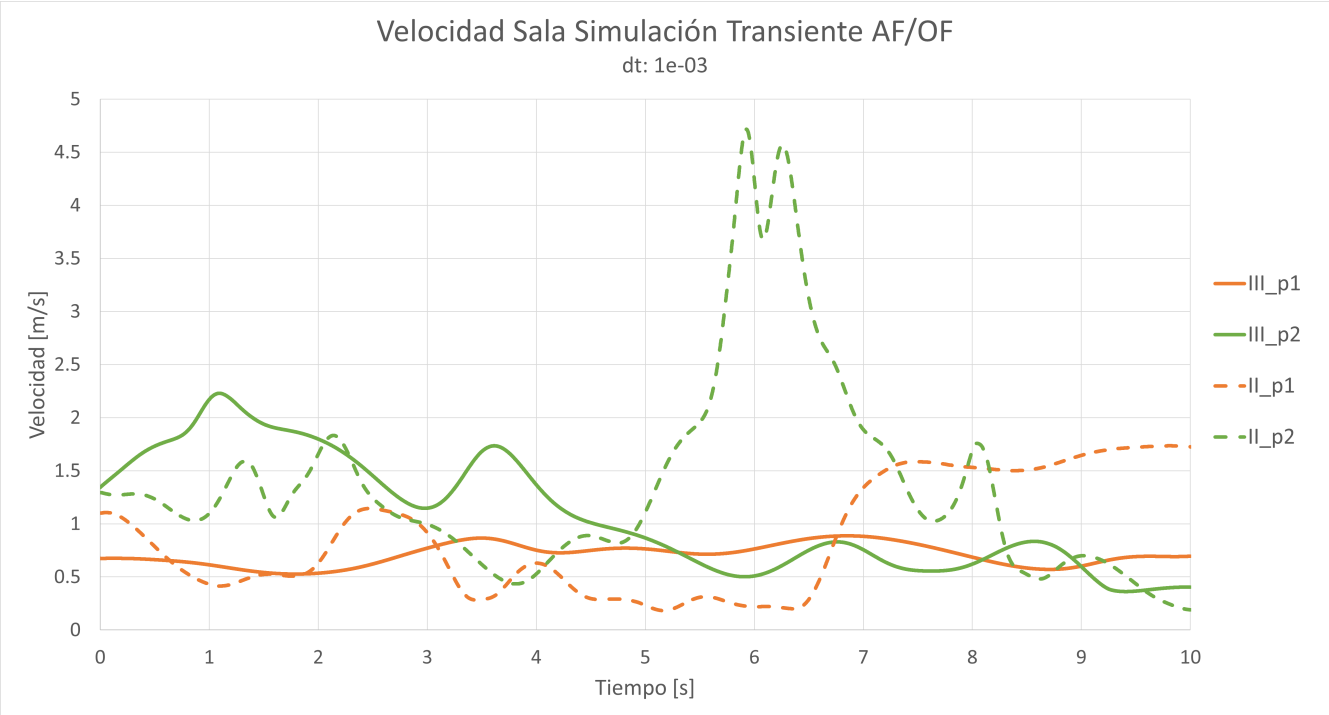


Figura 4.21: SA: Velocidad transiente puntos p1 y p2

Centrando el análisis a valores promedio, se encuentra que para tres de los cuatro puntos de muestreo los valores generados por las simulaciones II y III presentan rangos de operación e intervalos de oscilación en las curvas que no distan mucho unos de otros. La notoria diferencia en este análisis comparativo recae en el máximo de velocidad presentado en la figura 4.21. Aquí se muestra un máximo en la curva de velocidad de OpenFOAM dos veces mayor al encontrado en el mismo punto bajo la simulación de ANSYS Fluent.

Dado que con los valores de magnitud no representan el comportamiento del flujo al interior de la sala, se ha tomado un plano de corte normal al eje y en ambas simulaciones hacia el término de estas. En las figuras 4.23 y 4.24 se aprecian los contornos de velocidad en el eje x, definido positivo de izquierda a derecha. En ambos cuadros se aprecia la existencia de un flujo en esta dirección tanto desde una posición aproximada a la placa como en otras zonas. Se aprecia en mayor medida en OpenFOAM que cerca de ambas campanas, en la parte superior se generan los principales movimientos de flujo en la dirección positiva de x. El flujo se devuelve en ambos casos desde un punto medio de altura hacia la placa, generando una especie de bucle hacia el piso de la sala.

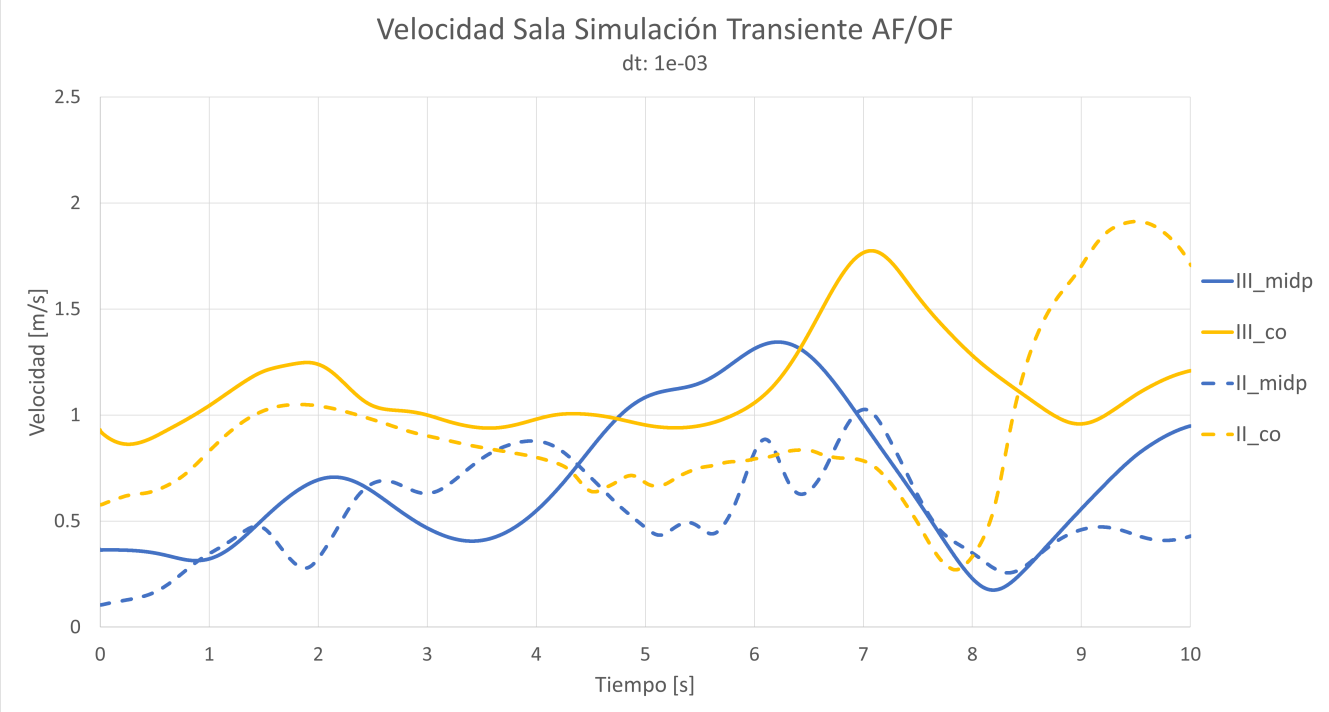


Figura 4.22: SA: Velocidad transiente puntos midpoint y co

Tanto para las velocidades transientes descritas en las figuras anteriores, como para el análisis de las velocidades de la simulación I se nota un fundamental aspecto entre los resultados provistos por OpenFOAM y ANSYS Fluent. En la mayoría de los casos la amplitud del rango de valores obtenidos es mayor para OpenFOAM. Junto con esto existe una mayor desviación en los valores para las formulaciones en estado transiente. Si bien en el promedio, la magnitud de las velocidades

difiere en un máximo de 27 %, no es posible evidenciar un claro patrón de velocidades para ambas simulaciones como si es posible de determinar con el análisis de temperaturas.

Otro aspecto relevante en cuanto a la magnitud de los valores de velocidad son los máximos inesperados y poco recurrentes a lo largo de la simulación. Claramente el primer caso de anormalidad mostrado en la simulación I, se da por consideraciones con en desarrollo de la simulación en un inicio, sin embargo el máximo de velocidades mostrado en la figura 4.21 para el punto *p2* no representa las mismas condiciones anómalas descritas con anterioridad.

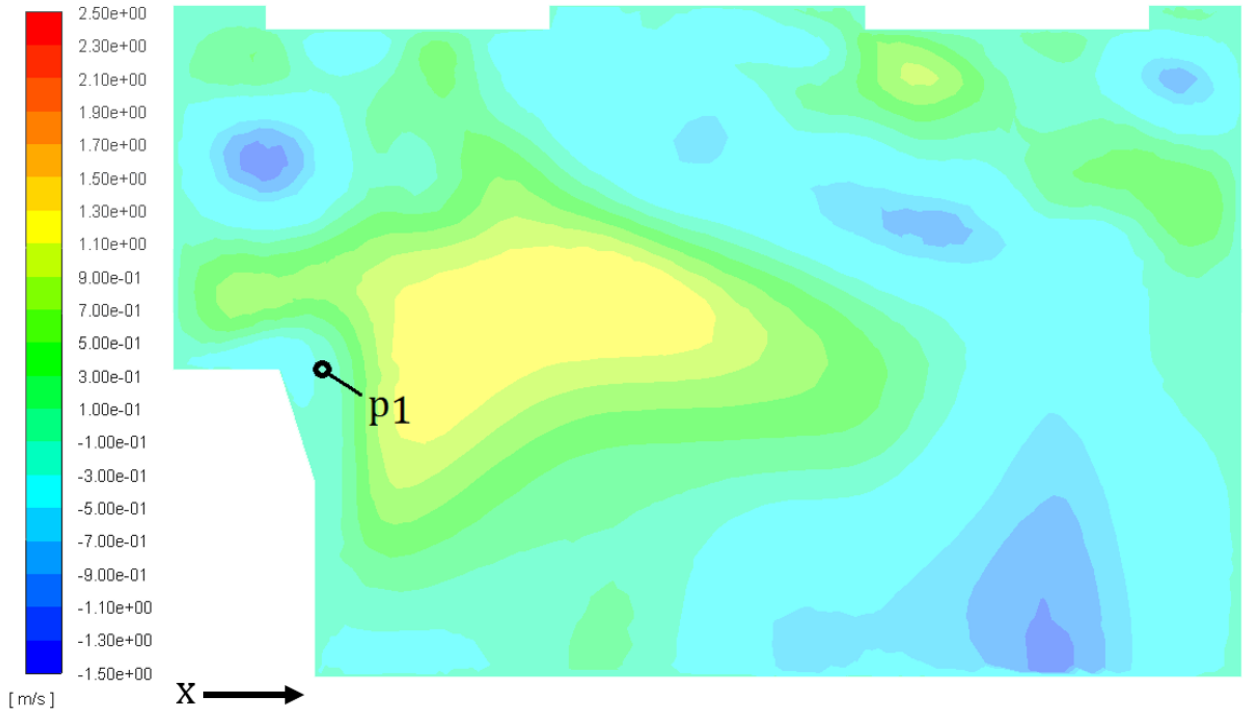


Figura 4.23: Perfil de velocidad Ux A. Fluent: t=10 s

Para comprender si el valor no esperado en la magnitud de velocidad para *p1* corresponde o no a valores correctamente simulados se debe considerar al menos un periodo de simulación más extenso, con el fin de determinar si esta anomalía es esporádica o bien se repite en el tiempo y dadas las condiciones iniciales del problema ha quedado en evidencia en la figura 4.21.

A pesar de las imprecisiones de los valores promedio de velocidad y anomalías varias que puedan diferenciar los comportamientos de las simulaciones II y III, se aprecia que la dinámica de flujo simulada en la sala da cuenta de mayores similitudes que diferencias. Tanto en la figura 4.23 como 4.24 se puede apreciar la existencia de un flujo en la dirección x al interior de la sala en ambas simulaciones. Este, a su vez origina una recirculación de aire localizado en el extremo inferior derecho. También se puede notar en ambas figuras el efecto de la extracción superior de aire. Si

bien existe un parecido en la magnitud y dirección de la velocidad cercana a la campana superior derecha, OpenFOAM presenta una mayor magnitud que su par.

En la campana izquierda por otra parte, el comportamiento es más bien distinto. La extracción de aire mostrada por la figura 4.23 da cuenta de un movimiento de aire hacia el centro de la campana, fenómeno que dista del descrito en la figura 4.24 donde este patrón no se distingue. Por último, en ambas figuras se aprecia el desplazamiento de aire en torno a la placa calefactora y al punto $p1$, aquí se podría dar un fenómeno muy similar al descrito en la sección 4.4.4, donde se aprecia un movimiento de aire muy similar en ambas simulaciones pero desfasado temporalmente.

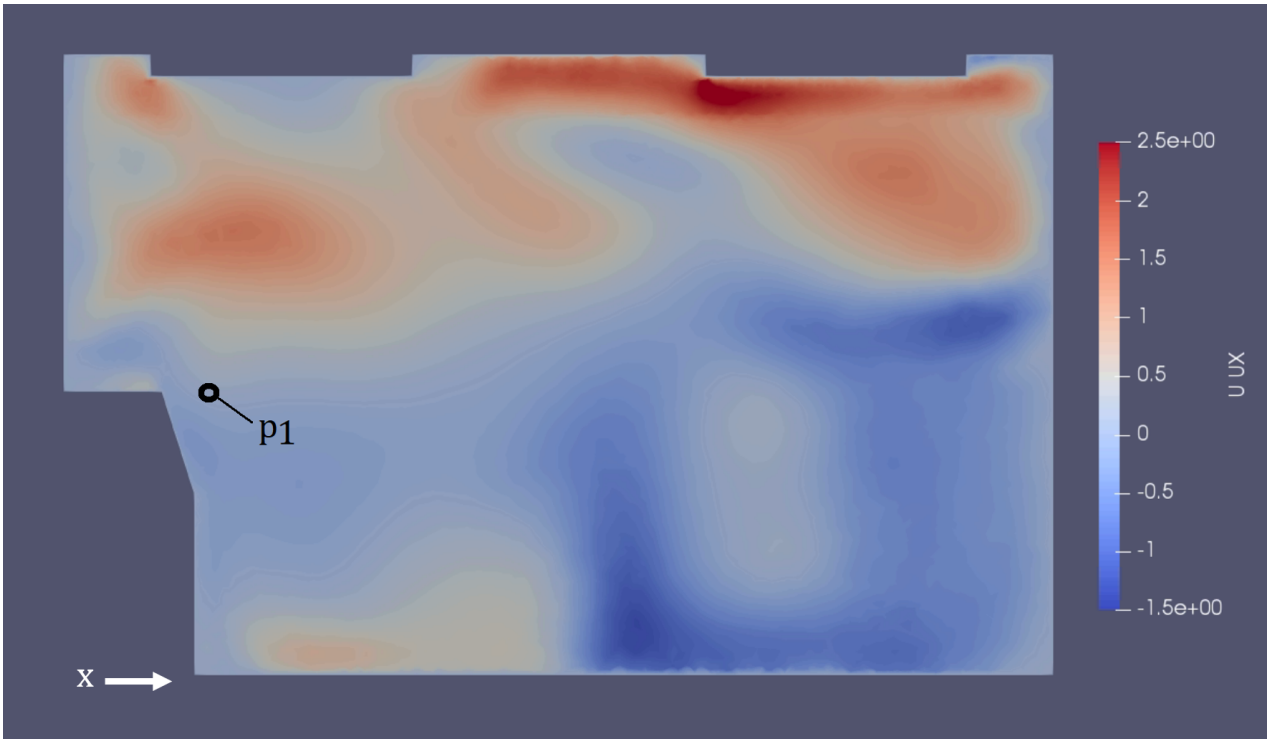


Figura 4.24: Perfil de velocidad Ux OpenFOAM: t=10 s

4.4.6. Comparación de continuidad y flujos

Para completar el análisis sobre las simulaciones I, II y III se realiza una medición a lo largo del tiempo de simulación sobre los flujos másicos en los ingresos y salidas de flujo. Si se tiene en consideración que la ecuación de continuidad es una máxima para resolver las ecuaciones gobernantes, tanto en la formulación estacionaria como transiente, los valores esperados para este apartado debiesen cumplir con esta vital condición.

La figura 4.25 muestra el flujo másico promedio por segundo para cada entrada y salida dentro de la sala. Recordando que las normas de signo difieren en ambos programas, se ha establecido como ingreso los valores negativos y los egresos como valores positivos. En este sentido, para las

simulaciones I y II se aprecian valores idénticos en todas las condiciones de borde. Las diferencias entre flujos se evidencian al realizar la comparación entre OpenFOAM y ANSYS Fluent. Este último software posee valores de flujo ligeramente mayores que su par de código abierto para cada condición de borde. En la tabla 4.13 se muestran los valores promedio de flujo a lo largo de la extensión de cada simulación.

Los valores de flujo calculados por Fluent maximizan en un 3 % los flujos de *outlet.campana*, *outlet.Pared* e *inlet.insufladores*, mientras que se maximiza en un 4 % respecto a OpenFOAM el valor del flujo en *inlet.cortina*. Este último es aquella superficie cuya condición de borde presenta un ingreso de aire como gradiente de presión.

Tabla 4.13: SA: Valores promedio de flujo másico

Simulación	Punto	Promedio [kg/s]	Desviación Std
I	inlet.cortina	-1.33	7.2E-01
	inlet.insufladores	-0.93	2.7E-04
	outlet.campanas	1.68	8.1E-04
	outlet.pared	0.58	2.5E-04
II	inlet.cortina	-1.34	2.1E-03
	inlet.insufladores	-0.93	1.5E-05
	outlet.campanas	1.68	8.4E-05
	outlet.pared	0.58	7.7E-06
III	inlet.cortina	1.38	3.1E-05
	inlet.insufladores	0.96	1.7E-13
	outlet.campanas	-1.74	1.2E-13
	outlet.pared	-0.60	5.1E-14

En la formulación da cada caso de estudio en ambos programas se procura establecer los mismos parámetros, constantes y unidades de los valores a simular. En este caso se pone especial énfasis en las propiedades del aire presentando en su mayoría los mismos valores para ambos programas. Existen pequeñas diferencias entre la declaración de valores para parámetros que podrían inducir la diferencia planteada en la figura 4.25. OpenFOAM emplea como parámetro la viscosidad dinámica del fluido, mientras que ANSYS Fluent utiliza los parámetros de viscosidad cinemática y densidad. La diferencia numérica entre ambas formulaciones recae en un 0.1 %, descartando esta como la principal causa de las diferencias mostradas.

Otra pequeña diferencia se encuentra en la formulación de la condición de borde de velocidad de entrada a través de los insufladores. Aquí en OpenFOAM se ha hecho una sub-división por componentes de la velocidad impuesta en Fluent, sin embargo las diferencias de magnitud son menores al 0.003 %. Descartados los valores ingresados por el usuario, la diferencia se explica en la

forma que poseen ambos programas de calcular el flujo a través de una sección. Hipótesis apoyada fuertemente por la constante diferencia porcentual en los cuatro valores de flujo medidos.

Finalmente, no está de más mencionar la naturaleza de la comparación. Ambos software poseen estructuras de cálculo diferentes acompañadas con esquemas de discretización numéricas distintas, por lo que para realmente poder determinar el origen de las diferencias, se debe conocer más en detalle la estructura del cálculo de flujo. Se propone entonces, ahondar en el cálculo de flujo másico, entendiendo que existen una serie de parámetros involucrados en el proceso con distintas tolerancias y precisiones.

Finalmente, independiente de las variaciones discutidas a partir de los casos simulados en este trabajo, se debe considerar una tercera etapa evaluadora y experimental que corrobore las condiciones aquí descritas. La validez de las simulaciones para fines prácticos, solo quedará respaldada y será posible ahondar más en detalle al tener una contraparte experimental. La realización u obtención de datos experimentales serían de gran ayuda para tanto validar las simulaciones presentadas en este desarrollo como para esclarecer algunas de las diferencias encontradas en el proceso.

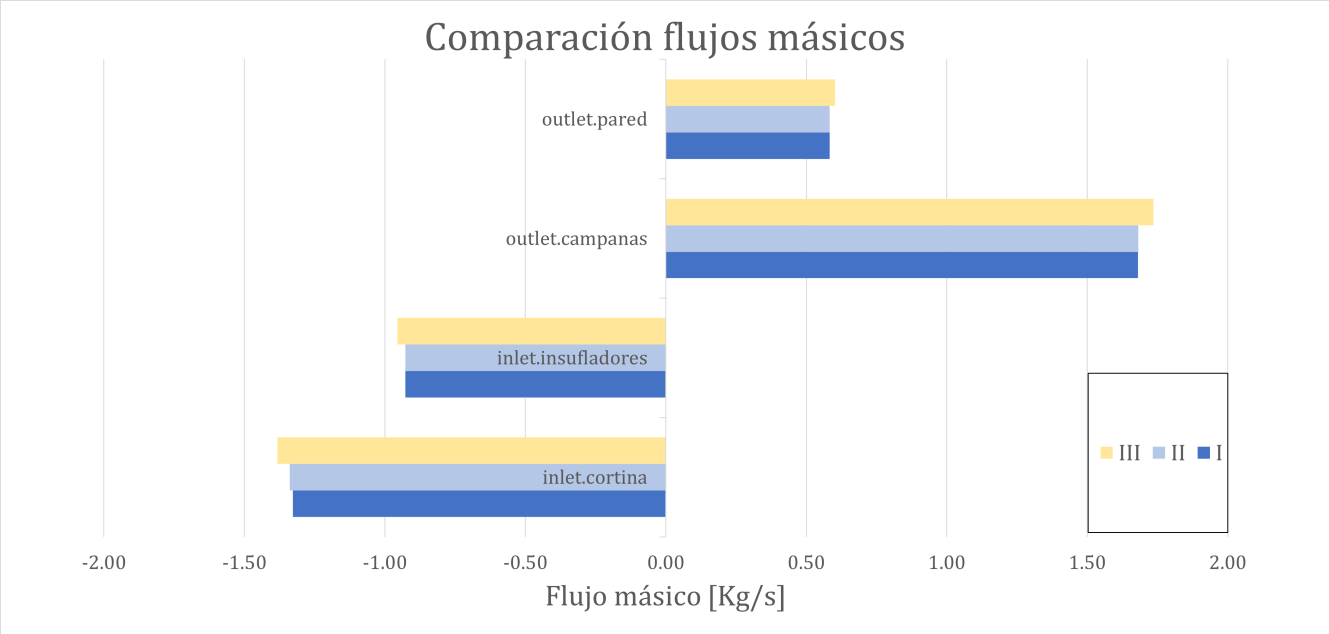


Figura 4.25: Flujo másico para simulaciones II y III

Capítulo 5

Metodología de implementación de OpenFOAM

Como parte de la realización de un análisis comparativo de las herramientas de simulación CFD ANSYS Fluent y OpenFOAM en un caso de ventilación industrial aplicado, existen una serie de aprendizajes tanto en el uso de estas herramientas como en las capacidades y desempeños de ambos programas. Es de interés práctico, realizar una metodología de uso que permita a un usuario con conocimientos básicos de mecánica de fluidos computacional, utilizar el software OpenFOAM para realizar simulaciones computacionales logradas en otros programas del rubro.

Este documento es una recopilación de información y experiencias asociadas principalmente a la realización de un caso de estudio de ventilación modelada empleando el modelo de turbulencia K-epsilon, tanto para estado estacionario como para transiente, con y sin la resolución de ecuación de energía. Ha sido desarrollado utilizando la séptima versión de este software en dos sistemas operativos en base a la distribución linux, Ubuntu 18.04 y CAE Linux.

OpenFOAM carece de una interfaz de usuario, quizás la mayor diferencia con los programas comerciales más conocidos. Sin embargo, el usuario podrá ver que la lista de instrucciones para llevar a cabo una simulación es acotada y breve. Por otra parte, es posible automatizar instrucciones y modificar la totalidad de los archivos empleados en el caso, otorgando al usuario un sinnúmero de modificaciones posibles.

El lector encontrará el documento en su totalidad en el Anexo C. El manual ha sido subdividido en una primera sección de conocimientos básicos y funcionamiento, luego un apartado referente al mallado y resumen de condiciones de borde mínimas. Finalmente, se describen los parámetros de control necesarios para la puesta en marcha de las simulaciones así como la visualización de los resultados finales.

Capítulo 6

Conclusiones

Mediante la información otorgada por SAME S.A. a través de información gráfica de un proyecto real, se construye de forma exitosa una geometría simplificada que da paso a la construcción de un caso de estudio de ventilación. En este proceso, se realizan pequeñas simplificaciones a la geometría que permiten dar cuenta de un volumen de control más sencillo, sin perder las dimensiones generales y condiciones de borde que hacen del caso de estudio final un ejemplo real de un caso de ventilación industrial aplicado.

El desarrollo del caso de estudio conlleva simplificaciones adicionales a la geometría principal determinada en un inicio. Estas se realizan en pos de un mejor desempeño computacional, originando un análisis sobre una geometría secundaria al proyecto. El desarrollo de este, continúa mediante un proceso de mallado y validación para ambas geometrías, donde resalta la importancia de la calidad de malla y la selección de topologías en la estabilidad numérica de las simulaciones posteriores. Al realizar una conversión del tipo de elemento de malla (tetraédricos a poliédricos), se logra una disminución del 81 % en la cantidad de elementos del mallado de la geometría principal manteniendo óptimos parámetros de calidad.

Se realizan simulaciones en estado transiente a partir de una inicialización en estado estacionario de 3000 iteraciones para una misma geometría en ambos programas utilizando como base el algoritmo SIMPLEC. Empleando transferencia de calor, son estudiadas las variables de temperatura, velocidad y flujo másico a través de las entradas y salidas del volumen de control. El análisis de estos parámetros evidencian similitudes en los comportamientos de las variables entre las simulaciones realizadas en OpenFOAM y ANSYS Fluent, considerando ambas herramientas apropiadas para su comparación. Sin embargo, se establecen importantes diferencias en la implementación entorno a un número de Courant igual a 5 en comparación a un paso de tiempo fijo de 0.001 segundos. Al emplear la primera condición, se encuentran máximos desproporcionados en el cálculo de velocidad de la simulación. Se da entonces origen a un análisis cuyo resultado muestra la propagación de valores inadmisibles para el problema desde de una de las condiciones de borde

hacia el interior de la geometría simulada, anulando su validez para futuras comparaciones.

A través del análisis de temperatura en el tiempo de simulación se evidencia como el estado estacionario influye en el desarrollo de formulación transiente, encontrándose sustanciales diferencias en los perfiles para los primeros dos segundos de simulación. Establecida esta diferencia, se da cuenta como durante la evolución temporal de las simulaciones, los valores de temperatura se asemejan mostrando un comportamiento similar el el desarrollo numérico por parte de ambos software.

Por otra parte, mediante análisis sobre la variable de velocidad se evidencian las mayores diferencias, mostrando un máximo de 27 % de diferencia en magnitud promedio para un punto de comparación entre ambas formulaciones transientes con un paso de tiempo constante de 0.001 segundos. La comparación de magnitudes de velocidad por parte de ambos programas muestra que OpenFOAM desarrolla un intervalo mayor de valores para esta variable de forma constante. Junto con esto, los valores de magnitud promedio son levemente mayores y presentan una mayor dispersión de datos en comparación a las simulaciones equivalentes con ANSYS Fluent.

La mayor similitud de este análisis recae en el cálculo de flujos másicos, donde se obtiene una diferencia numérica máxima de 4 % de OpenFOAM respecto al flujo obtenido mediante ANSYS Fluent.

El conocimiento adquirido sobre el funcionamiento de ANSYS Fluent y OpenFOAM mediante la literatura en un primer lugar y luego la experiencia de la formulación de este proyecto, recae en la capacidad de lograr resultados apropiados y comparables para el caso de estudio. Este conocimiento es registrado en un manual de 49 páginas que permite realizar simulaciones de las mismas características de ventilación industrial a futuros usuarios de OpenFOAM.

Para el desarrollo futuro de este estudio, se sugiere realizar un análisis de sensibilidad sobre el número de Courant para determinar las reales implicancias en la estabilidad numérica y exactitud de los valores simulados. Se sugiere también, aumentar el tiempo de simulación en aras de extender el análisis realizado en este trabajo. Finalmente, se hace hincapié en que el proceso de simulación debe ser contrastado y validado con datos experimentales, los cuales darán cuenta de cuan certeras y útiles pueden ser las herramientas CFD comparadas en esta memoria de título.

Bibliografía

Ansys. (2013). Ansys Fluent Theory Guide. *ANSYS Inc., USA*, *15317*(November), 724–746.

ANSYS, I. (2018). ANSYS Fluent Tutorial Guide R18. *ANSYS Fluent Tutorial Guide 18*(April), 724–746.

Blocken, B. (2018). *LES over RANS in building simulation for outdoor and indoor applications: A foregone conclusion?* (Vol. 11) (n.º 5). doi: 10.1007/s12273-018-0459-3

Earls Brennen, C. (1995). Law of the wall. *An Internet Book on Fluid Dynamics Law*, *7*(12), 3078. doi: 10.1063/1.868685

Fluent, A. (2019). *ANSYS User Guide v191.*

Fridolin, K. (2012). CFD for air induction systems with OpenFOAM. *Thesis*, 1–59.

Greenshields, C. (2014). The OpenFOAM 7 User Guide. (February), 212.

Heidenreich, E., Gaspar, F. J., Lisbona, F. J., y Rodrigo, C. (2010). MÉTODOS MULTIMALLA SOBRE MALLAS SEMI-ESTRUCTURADAS.

Hong, S. W., Exadaktylos, V., Lee, I. B., Amon, T., Youssef, A., Norton, T., y Berckmans, D. (2017). Validation of an open source CFD code to simulate natural ventilation for agricultural buildings. *Computers and Electronics in Agriculture*, *138*, 80–91. doi: 10.1016/j.compag .2017.03.022

ISSA, R. I. (1986). Solution of the Implicity Discretised Fluid Flow Equations by Operator-Splitting. *Journal of Computational Physics*, *62*(1), 40–65. doi: 10.1080/10407782.2016 .1173467

Jiang, Y., Allocca, C., y Chen, Q. (2004). Validation of CFD Simulations for Natural Ventilation. *International Journal of Ventilation*, *2*(4), 359–369. doi: 10.1080/14733315.2004.11683678

Joel H. Ferziger, M. P. (2001). *Computational Methods For Fluid Dynamics*. Springer.

Liu, F. (2016). A Thorough Description Of How Wall Functions Are Implemented In OpenFOAM. *CFD With OpenSource Software, edited by Nilsson H.*.

Moukalled, F., Mangani, L., y Darwish, M. (2016). Erratum to: The Finite Volume Method in

Computational Fluid Dynamics. En (Vol. M, pp. E1–E1). doi: 10.1007/978-3-319-16874-6_21

Munson, B. R., Young, D. F., y Okiishi, T. H. (2002). *Fundamentals of Fluid Mechanics, fourth edition*. doi: 10.1016/B978-0-12-381383-1.00002-3

OpenCFD. (2019). Open FOAM v1906: User Guide. (June), 78.

OpenCFD Ltd 2019. (2019). *OpenFOAM: User Guide: OpenFOAM®: Open source CFD : Documentation.*

Patankar, S. V. (1980). *Numerical Heat Transfer and Fluid Flow*. McGraw-Hill.

Posner, J. D., Buchanan, C. R., y Dunn-Rankin, D. (2003). Measurement and prediction of indoor air flow in a model room. *Energy and Buildings*, *35*(5), 515–526. doi: 10.1016/S0378-7788(02)00163-9

Ramponi, R., y Blocken, B. (2012). CFD simulation of cross-ventilation for a generic isolated building: Impact of computational parameters. *Building and Environment*, *53*, 34–48. doi: 10.1016/j.buildenv.2012.01.004

The OpenFOAM Foundation. (2021). *OpenFOAM v7 User Guide.*

Tian, Z. F., Tu, J. Y., Yeoh, G. H., y Yuen, R. K. (2006). On the numerical study of contaminant particle concentration in indoor airflow. *Building and Environment*, *41*(11), 1504–1514. doi: 10.1016/j.buildenv.2005.06.006

Tu, J. G.-H. Y. C. L. (2019). *Computational Fluid Dynamics*. doi: 10.1016/B978-0-444-64046-8.00123-3

Versteeg, H. K., y Malalasekera, W. (2016). *Introduction to Computational Fluid Dynamics* (Vol. M). doi: 10.1002/9781119369189

Welahettige, P., y Vaagsaether, K. (2018). Comparison of OpenFOAM and ANSYS Fluent. *Proceedings of The 9th EUROSIM Congress on Modelling and Simulation, EUROSIM 2016, The 57th SIMS Conference on Simulation and Modelling SIMS 2016*, *142*, 1005–1012. doi: 10.3384/ecp171421005

Wimshurst, A. (2018). *The Boussinesq Approximation for Bouyancy Driven (Natural Convection) Flow.*

Zhang, J., Liang, Y., Ren, T., Wang, Z., y Wang, G. (2016). Transient CFD modelling of low-temperature spontaneous heating behaviour in multiple coal stockpiles with wind forced convection. *Fuel Processing Technology*, *149*, 55–74. doi: 10.1016/j.fuproc.2016.04.011

Anexo A

Archivos de simulación OpenFOAM

A continuación se muestran los archivos de OpenFOAM empleados en la simulación definitiva del caso de estudio. Se hace la distinción entre archivos de condiciones de borde, control, propiedades del fluido y diccionarios externos de registro.

A.1. Condiciones iniciales y de borde

Código A.1: Archivo OpenFOAM: alphat

```
FoamFile{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      alphat;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [1 -1 -1 0 0 0 0];
internalField   uniform 0;
boundaryField{
   "(wall-fluid.vol|wall-fluid.entorno.insufladores|wall-fluid.entorno.campanas|wall.particulado2|wall.particulado1|wall.celosia.back)"{
        type            compressible::alphatWallFunction;
        nut             nut;
        Prt             0.85;
        value           uniform 0;    }
    outlet.campanas{
       type calculated;
      value uniform 0;    }
    outlet.pared{
        type calculated;
      value uniform 0;    }
    inlet.insufladores{
        type calculated;
      value uniform 0;    }
    inlet.cortina{
        type calculated;
      value uniform 0;    }}
```

Código A.2: Archivo OpenFOAM: epsilon

```
  /*--------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 2 -3 0 0 0 0];
internalField   uniform 14.855;
boundaryField{
   "(wall-fluid.vol|wall-fluid.entorno.insufladores|wall-fluid.entorno.campanas|wall.particulado2|wall.particulado1|wall.celosia.back)"{
        type            epsilonWallFunction;
        value           uniform 14.855;
    }
    outlet.campanas{
      type            zeroGradient;
      }
    outlet.pared{
        type            zeroGradient;
    }
    inlet.insufladores    {
        type            fixedValue;
        value           uniform 14.855;
    }
    inlet.cortina    {
        type            zeroGradient;
    }
}
```

Código A.3: Archivo OpenFOAM: k

```
  /*--------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 2 -2 0 0 0 0];
internalField   uniform 0.375;
boundaryField{
   "(wall-fluid.vol|wall-fluid.entorno.insufladores|wall-fluid.entorno.campanas|wall.particulado2|wall.particulado1|wall.celosia.back)"{
        type            kqRWallFunction;
        value           uniform 0;
    }
    outlet.campanas    {
   type            zeroGradient;
    }
    outlet.pared    {
        type            zeroGradient;
    }
    inlet.insufladores    {
        type            fixedValue;
        value           uniform 0.375;
    }
    inlet.cortina    {
        type            zeroGradient;
    }
}
```

Código A.4: Archivo OpenFOAM: nut

```
  *---------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 2 -1 0 0 0 0];
internalField   uniform 0;
boundaryField{
    "(wall-fluid.vol|wall-fluid.entorno.insufladores|wall-fluid.entorno.campanas|wall.particulado2|wall.particulado1|wall.celosia.back)"{
        type            nutkWallFunction;
        value           uniform 0;
    }
    outlet.campanas{
      type            calculated;
        value           uniform 0;
    }
    outlet.pared{
        type            calculated;
        value           uniform 0;
    }
    inlet.insufladores{
        type            calculated;
        value           uniform 0;
    }
    inlet.cortina{
        type            calculated;
        value           uniform 0;
    }
}
```

Código A.5: Archivo OpenFOAM: p

```
  /*--------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [1 -1 -2 0 0 0 0];
internalField   uniform 1e5;
boundaryField{
   "(wall-fluid.vol|wall-fluid.entorno.insufladores|wall-fluid.entorno.campanas|wall.particulado2|wall.particulado1|wall.celosia.back)"{
        type            calculated;
        value           \$internalField;
    }
    outlet.campanas{
      type            calculated;
        value           \$internalField;
    }
    outlet.pared{
        type            calculated;
        value           \$internalField;
    }
    inlet.insufladores{
        type            calculated;
        value           \$internalField;
    }
    inlet.cortina{
        type            calculated;
        value           \$internalField;
    }
}
```

Código A.6: Archivo OpenFOAM: p_rgh

```
  /*--------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p_rgh;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [1 -1 -2 0 0 0 0];
internalField   uniform 1e5;
boundaryField{
   "(wall-fluid.vol|wall-fluid.entorno.insufladores|wall-fluid.entorno.campanas|wall.particulado2|wall.particulado1|wall.celosia.back)"{
        type            fixedFluxPressure;
        value           uniform 1e5;
    }
    outlet.campanas{
      type            zeroGradient;
    }
    outlet.pared{
      type            zeroGradient;
    }
    inlet.insufladores {
        type            zeroGradient;
    }
    inlet.cortina{
        type            fixedValue;
        value           \$internalField;
    }
}
```

Código A.7: Archivo OpenFOAM: T

```
  /*--------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile {
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      T;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 0 0 1 0 0 0];
internalField   uniform 293; // 20°C
boundaryField{
   "(wall-fluid.vol|wall-fluid.entorno.insufladores|wall-fluid.entorno.campanas|wall.celosia.back)"{
        type            zeroGradient;
    }
    wall.particulado2{
        type            fixedValue;
        value           uniform 373; // 100 degC
    }
    wall.particulado1{
        type            fixedValue;
        value           uniform 373; // 100 degC
    }
    outlet.campanas{
        type            zeroGradient;
    }
    outlet.pared{
        type            zeroGradient;
    }
    inlet.insufladores{
        type            fixedValue;
        value           uniform 293; // 20 degC
    }
    inlet.cortina{
        type            fixedValue;
        value           uniform 293; // 20 degC
    }
}
```

Código A.8: Archivo OpenFOAM: U

```
  /*--------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);
boundaryField{
   "(wall-fluid.vol|wall-fluid.entorno.insufladores|wall-fluid.entorno.campanas|wall.particulado2|wall.particulado1|wall.celosia.back)"{
       type            noSlip;
    }
    outlet.pared{
       type            fixedValue;
      value       uniform (0 0.05 0);
    }
    outlet.campanas{
       type            fixedValue;
      value       uniform (0 0 0.328);
    }
    inlet.insufladores{
       type            fixedValue;
      value        uniform (0 33.03  -12.02);
    }
    inlet.cortina    {
       type            zeroGradient;
    }
}
```

A.2. Control

Código A.9: Archivo OpenFOAM: controlDict

```
 /*--------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
application     buoyantPimpleFoam;
//~ application     buoyantSimpleFoam;

//~ startFrom       startTime;
startFrom       latestTime;
startTime       0;
stopAt          endTime;
endTime         3005;
deltaT          0.000005;
//~ deltaT        1;
writeControl adjustableRunTime;
//~ writeControl    timeStep;
writeInterval   0.1;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;
adjustTimeStep  yes;
maxCo           5;
functions{
    Co1{
        type            CourantNo;
      libs          ("libfieldFunctionObjects.so");
        executeControl      timeStep;
        writeControl        writeTime;}

   #includeFunc  flowRatePatch(name=outlet.campanas)
    #includeFunc  flowRatePatch(name=outlet.pared)
    #includeFunc  flowRatePatch(name=inlet.insufladores)
    #includeFunc  flowRatePatch(name=inlet.cortina)
    #includeFunc  probes
    #includeFunc  residuals}
```

Código A.10: Archivo OpenFOAM: decomposeParDict

```
 /*--------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      decomposeParDict;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
numberOfSubdomains 8;
method          scotch;

distributed     no;
roots           ( );
```

Código A.11: Archivo OpenFOAM: fvSolution

```
 /*--------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
solvers{
    "rho.*"{
        solver          PCG;
        preconditioner  DIC;
        tolerance       0;
        relTol          0;}
    p_rgh{
        solver          GAMG;
        tolerance       1e-8;
        relTol          0.0;
      maxIter       1000;
      smoother    DICGaussSeidel;}
    p_rghFinal{
        \$p_rgh;
        relTol          0;}

    "(U|h|e|k|epsilon|R)"{
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-08;
```

```
        relTol          0.0;
        maxIter     1000;}
    "(U|h|e|k|epsilon|R)Final"{
        \$U;
        relTol          0;}

    Phi{
      solver          PCG;
      preconditioner  DIC;
      tolerance       1e-06;
      relTol          0;
      minIter         0;
      maxIter         1000;}
}

SIMPLE{
   consistent  yes;
    momentumPredictor no;
    nNonOrthogonalCorrectors 2;
    pRefCell        0;
    pRefValue       0;

    residualControl{
        p_rgh           1e-4;
        U               1e-4;
        h               1e-4;
        "(k|epsilon|omega)" 1e-3;}
}

potentialFlow{
   nNonOrthogonalCorrectors 20;
   PhiRefCell 0;
   PhiRefPoint 0;
   pRefCell    0;
   PhiRefValue    0;}

PIMPLE{
    momentumPredictor yes;
    nNonOrthogonalCorrectors 1;
    nCcorrectors         2;
    residualControl{
        p_rgh               1e-3;
        U               1e-3;
        "(k|epsilon|omega|f|v2)" 1e-3;}
}

relaxationFactors{
    equations{
        p               0.3;
        p_rgh       0.3; //cambio ultimo 24 marzo
        U               0.7;
        k               0.7;
        epsilon         0.7;
        ".*"            0.9; }
}
```

Código A.12: Archivo OpenFOAM: fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
=========                 |
\\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration     | Website:  https://openfoam.org
  \\  /    A nd           | Version:  7
   \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
ddtSchemes{
#ifeq \${FOAM_APPLICATION}    buoyantSimpleFoam
   default         steadyState;
#else
   default         Euler;
#endif }

gradSchemes{
    default         cellMDLimited leastSquares 1.0;    }

divSchemes{
#ifeq \${FOAM_APPLICATION}    buoyantSimpleFoam
   default           none;
   div(phi,U)         bounded Gauss linearUpwind grad(U);
   div(phi,K)         bounded Gauss linear;
   div(phi,h)         bounded Gauss upwind;
   div(phi,k)         bounded Gauss upwind;
   div(phi,epsilon)   bounded Gauss upwind;
   div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;

#else
   default           none;
   div(phi,U)         Gauss linearUpwind grad(U);
   div(phi,k)         Gauss upwind;
   div(phi,epsilon)   Gauss upwind;
   div(phi,R)         Gauss upwind;
   div(R)             Gauss linear;
   div((nuEff*dev2(T(grad(U))))) Gauss linear;
   div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
   div(phi,K)       Gauss linear;
   div(phi,Ekp)     Gauss linear;
   div(phi,h)       Gauss upwind;
#endif}

laplacianSchemes{
    default         Gauss linear limited 1.0; }
interpolationSchemes{
    default         linear; }
snGradSchemes{
    default         limited corrected 1.0; }
```

A.3. Constantes

Código A.13: Archivo OpenFOAM: g

```
 /*--------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    location    "constant";
    object      g;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 1 -2 0 0 0 0];
value           (0 0 -9.81);
```

Código A.14: Archivo OpenFOAM: thermophysicalProperties

```
 /*--------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile {
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermophysicalProperties;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
thermoType{
    type            heRhoThermo; //modelo según entalpía y energia interna
    mixture          pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleEnthalpy;}

pRef            100000;
mixture //propiedades para el aire{
    specie    {
        molWeight       28.96;}
    thermodynamics{
        Cp              1006.43;
        Hf              0;}
    transport{
        mu              1.831e-05;
        Pr              0.705;} }
```

Código A.15: Archivo OpenFOAM: transportProperties

```
 /*--------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;
nu              [0 2 -1 0 0 0 0] 1.46e-05;
```

Código A.16: Archivo OpenFOAM: turbulenceProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
simulationType RAS;

RAS{
    RASModel        kEpsilon;
    turbulence      on;
    printCoeffs     on;}
```

A.4. Diccionarios complementarios

Código A.17: Archivo OpenFOAM: fvOptions

```
  /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvOptions;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
temperatureLimit{
    active true;
    max 400;
    min 200;
    selectionMode all;
    type limitTemperature;
}
```

Código A.18: Archivo OpenFOAM: probes

```
  /*--------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |

\*---------------------------------------------------------------------------*/

#includeEtc "caseDicts/postProcessing/probes/probes.cfg"

fields (p p_rgh U k epsilon T);
probeLocations
(
     // Midpoint
    (2.275 2.585 1.425)
    // placa 1
    (0.700 2.875 1.300)
    // placa 2
    (3.500 2.325 1.250)
    // cortina
    (1.000 1.060 1.500)
);
```

Código A.19: Archivo OpenFOAM: residuals

```
 /*--------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |
-------------------------------------------------------------------------------
\*---------------------------------------------------------------------------*/
#includeEtc "caseDicts/postProcessing/numerical/residuals.cfg"

fields (p T U k epsilon);
```

Código A.20: Archivo OpenFOAM: flowRatePatch

```
 /*--------------------------------*- C++ -*----------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     | Website:  https://openfoam.org
   \\  /    A nd           | Version:  7
    \\/     M anipulation  |

\*---------------------------------------------------------------------------*/

name    <patchName>;

#includeEtc "caseDicts/postProcessing/flowRate/flowRatePatch.cfg"
```

Anexo B

Complemento resultados

Tabla B.1: Resumen equipos utilizados para desarrollo

Computador	Sistema Operativo	Procesador	Software
Personal	Ubuntu 18.04	Intel® Core™ i7-4770S 3.10GHz × 8	OpenFOAM v7
Universidad I	Windows 7	Intel® Xeon E3-1241 v3 3.50GHz × 8	ANSYS 18.2
Universidad II	Windows 7	Intel® Core™ i7-4770S 3.10GHz × 8	ANSYS 18.2
SAME S.A.	CAE Linux en VM[1]	Intel® Xeon E7-4870 2.40GHz × 30	OpenFOAM v7

[1] VM representa una máquina virtual. En este caso se emplea el software VMware para albergar CAE Linux sobre Windows 10.

B.1. Mallado e independencia de mallado

A continuación se presentan las tablas complementarias al test de independencia de mallas.

Tabla B.2: Mallas campana

Malla	Elementos	Nodos	Skewness			Razón de aspecto			Calidad Ortogonal		
			promedio	max	std dev	promedio	max	std dev	promedio	max	std dev
M23	2642895	484514	0.259	0.609	0.101	1.890	4.419	0.374	0.740	0.391	0.100
M24	3579924	652869	0.258	0.707	0.105	1.880	4.515	0.370	0.741	0.293	0.099
M21	996193	188073	0.260	0.853	0.102	1.891	11.149	0.410	0.739	0.147	0.100

Tabla B.3: Mallas Sala de Alimentación

Malla	Elementos	Nodos	Skewness			Razón de aspecto			Calidad Ortogonal		
			promedio	max	std dev	promedio	max	std dev	promedio	max	std dev
M12	2155881	380256	0.245	0.600	0.100	1.855	4.360	0.370	0.755	0.405	0.099
M13	5118808	890403	0.241	0.602	0.100	1.850	4.330	0.369	0.758	0.398	0.099
M11	1675528	295499	0.248	0.621	0.102	1.860	4.970	0.373	0.751	0.378	0.100

B.2. Resultados complementarios: Campana

Tabla B.4: Valores de flujo por orificio para extracción de aire

Posición	y1	y2	y3	y4	y5	y6	y7	Promedio [kg/min]
x1	0.246	0.315	0.334	0.311	0.321	0.310	0.260	0.299
x2	0.261	0.297	0.396	0.315	0.394	0.292	0.246	0.314
x3	0.267	0.300	0.383	0.480	0.386	0.295	0.260	0.339
x4	0.271	0.289	0.316	0.337	0.313	0.286	0.264	0.296
x5	0.261	0.264	0.266	0.270	0.266	0.263	0.258	0.264
x6	0.251	0.244	0.237	0.240	0.237	0.242	0.242	0.242
x7	0.231	0.226	0.220	0.218	0.218	0.230	0.228	0.225
x8	0.218	0.211	0.206	0.204	0.207	0.212	0.216	0.211
x9	0.202	0.190	0.194	0.187	0.193	0.187	0.203	0.194
x10	0.200	0.181	0.178	0.177	0.183	0.186	0.203	0.187
x11	0.188	0.174	0.173	0.173	0.177	0.176	0.188	0.178
Promedio kg/min	0.236	0.245	0.264	0.265	0.263	0.243	0.233	-

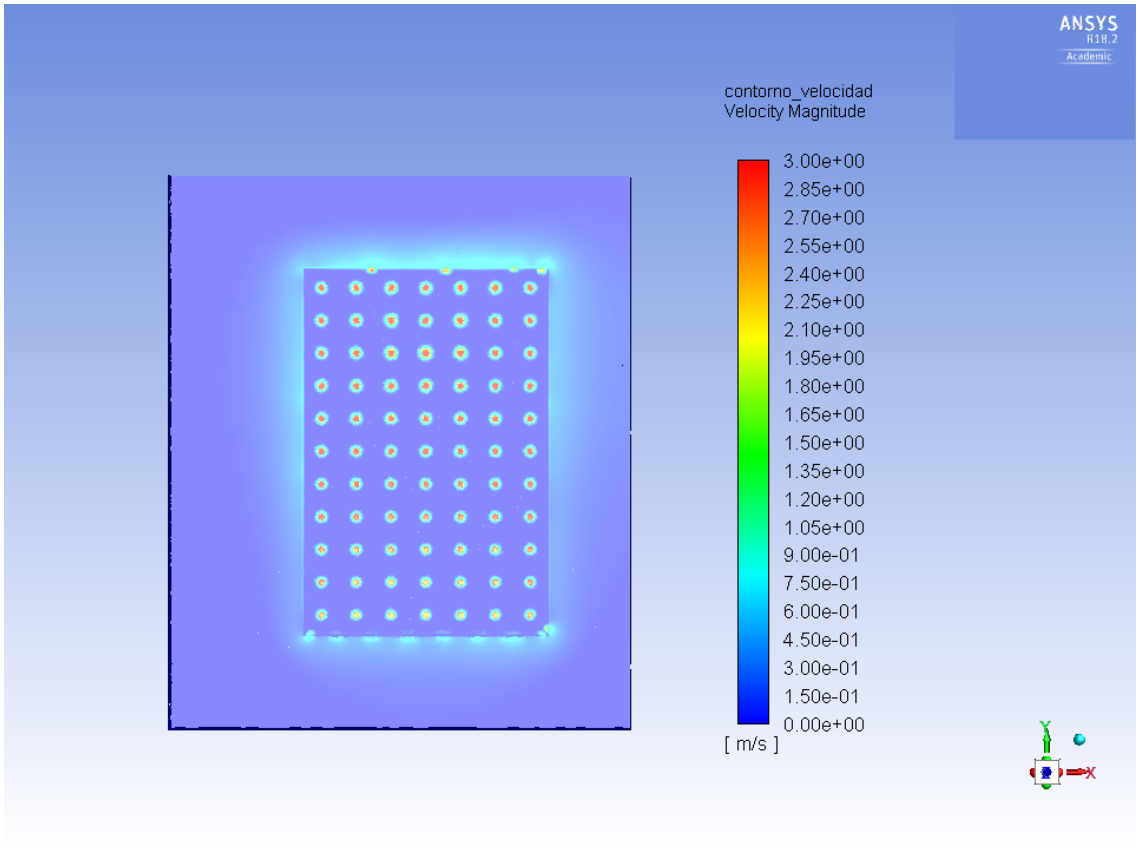


Figura B.1: Plano en corte de orificios Campana

B.3. Resultados complementarios: Sala de Alimentación

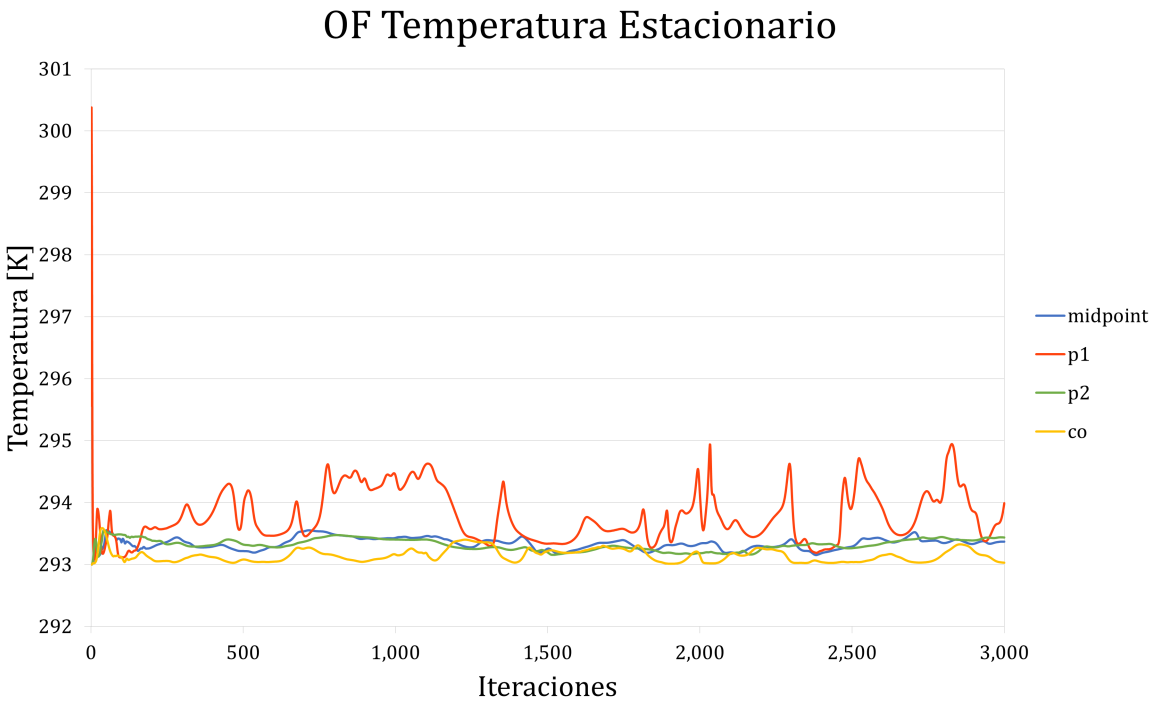


Figura B.2: Temperatura puntos de medición estado estacionario OpenFOAM

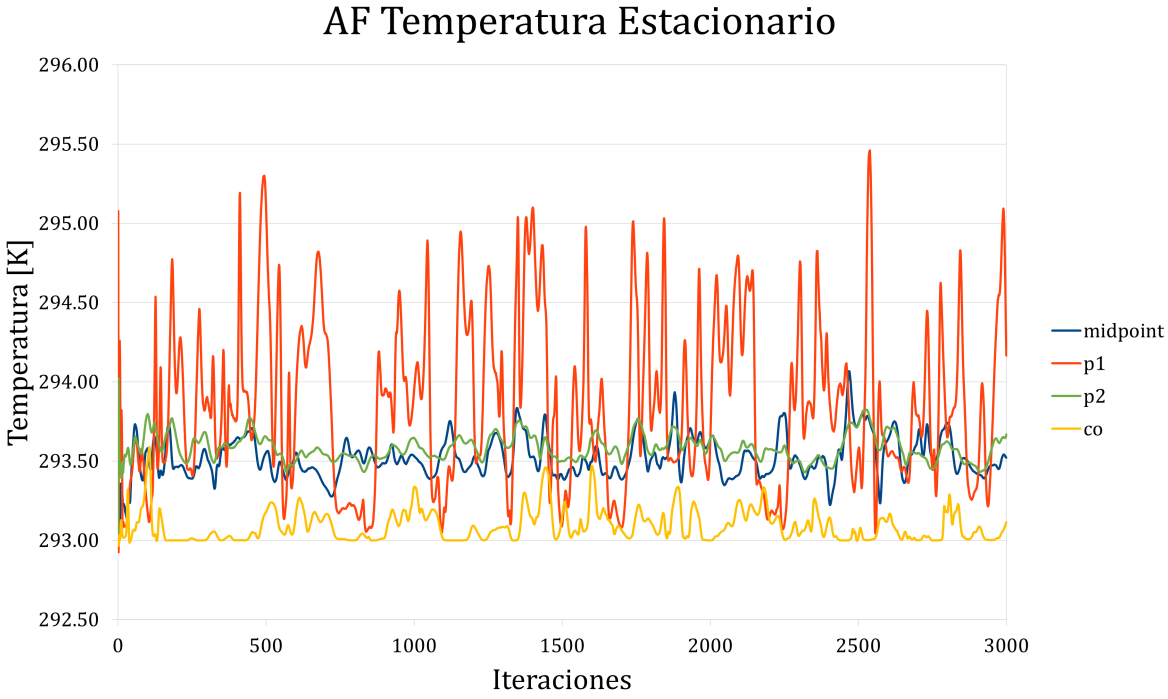


Figura B.3: Temperatura puntos de medición estado estacionario ANSYS Fluent

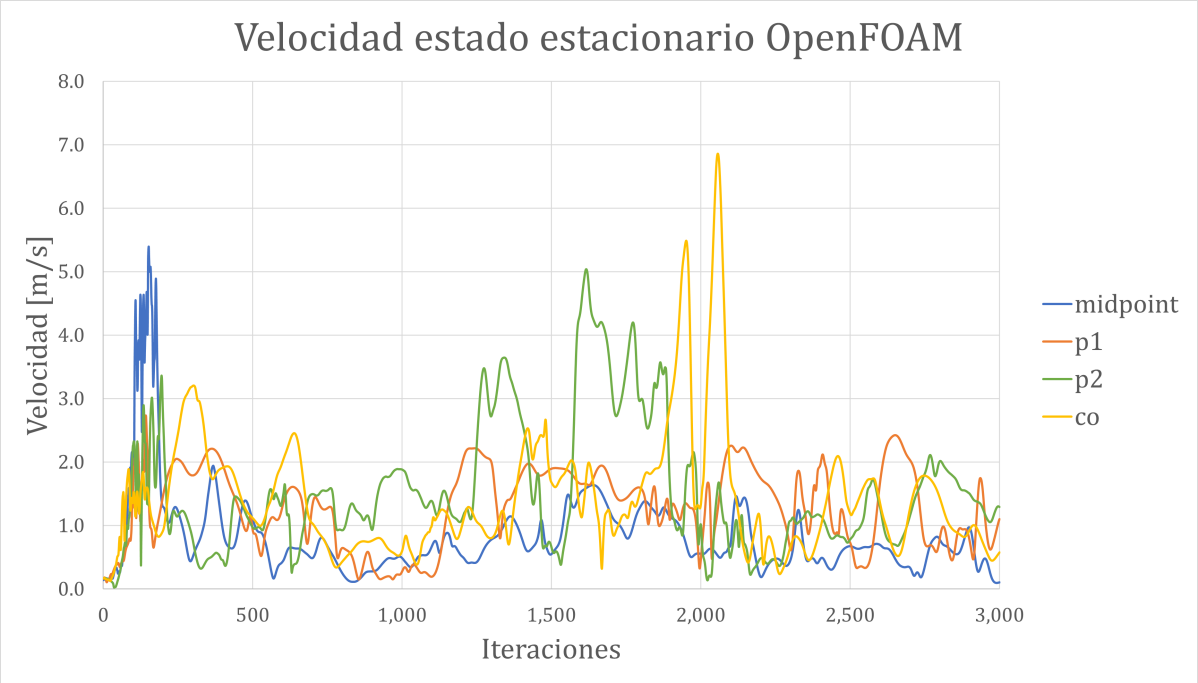


Figura B.4: Velocidad puntos de medición estado estacionario OpenFOAM

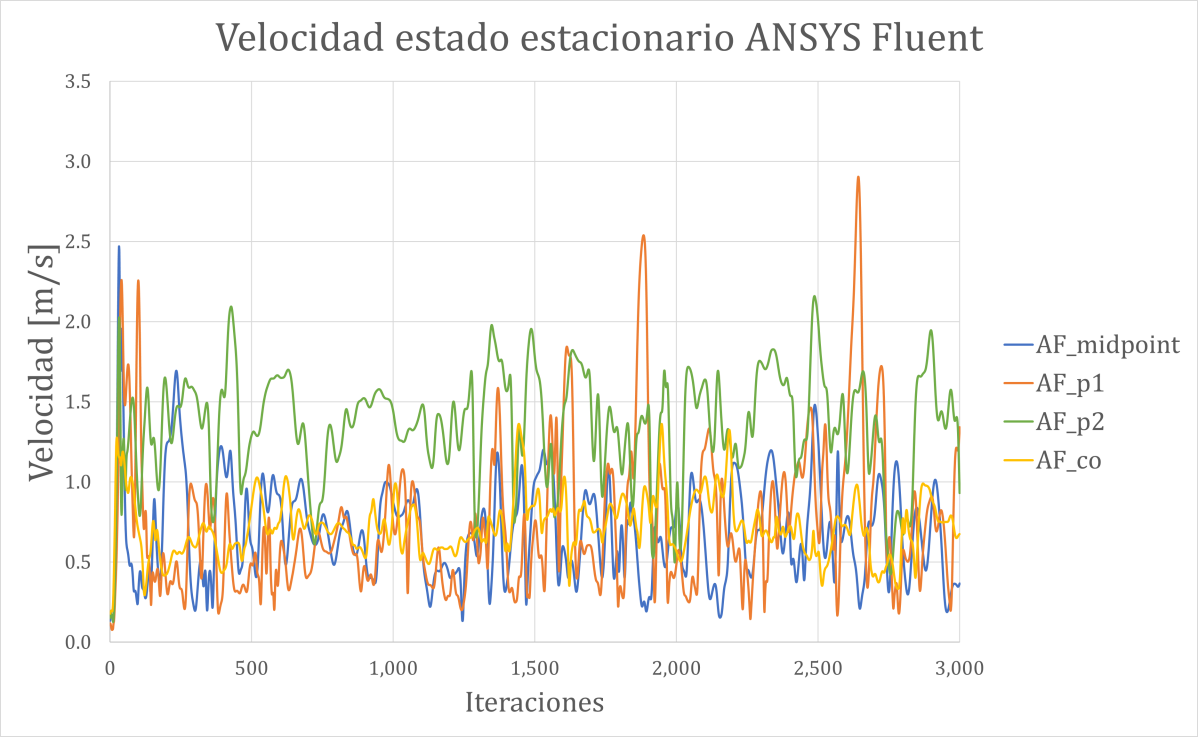


Figura B.5: Velocidad puntos de medición estado estacionario ANSYS Fluent

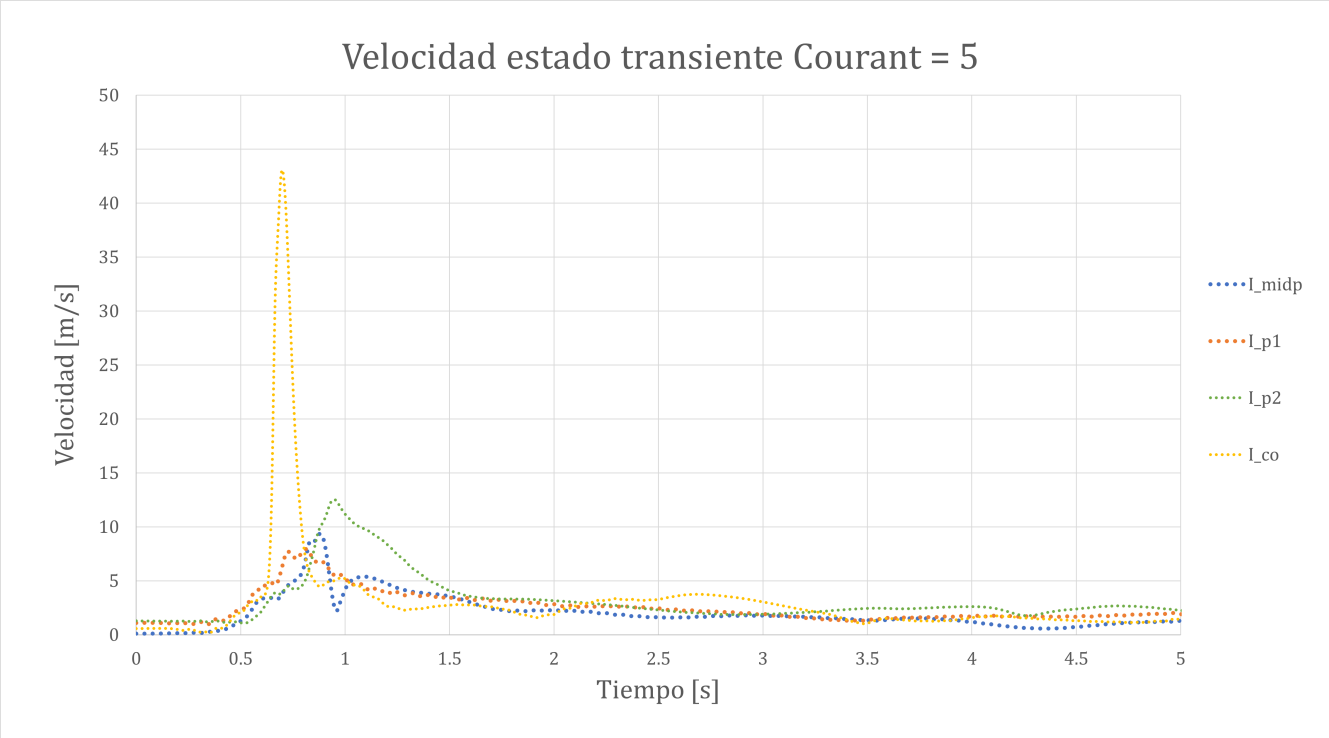


Figura B.6: Velocidad simulación I

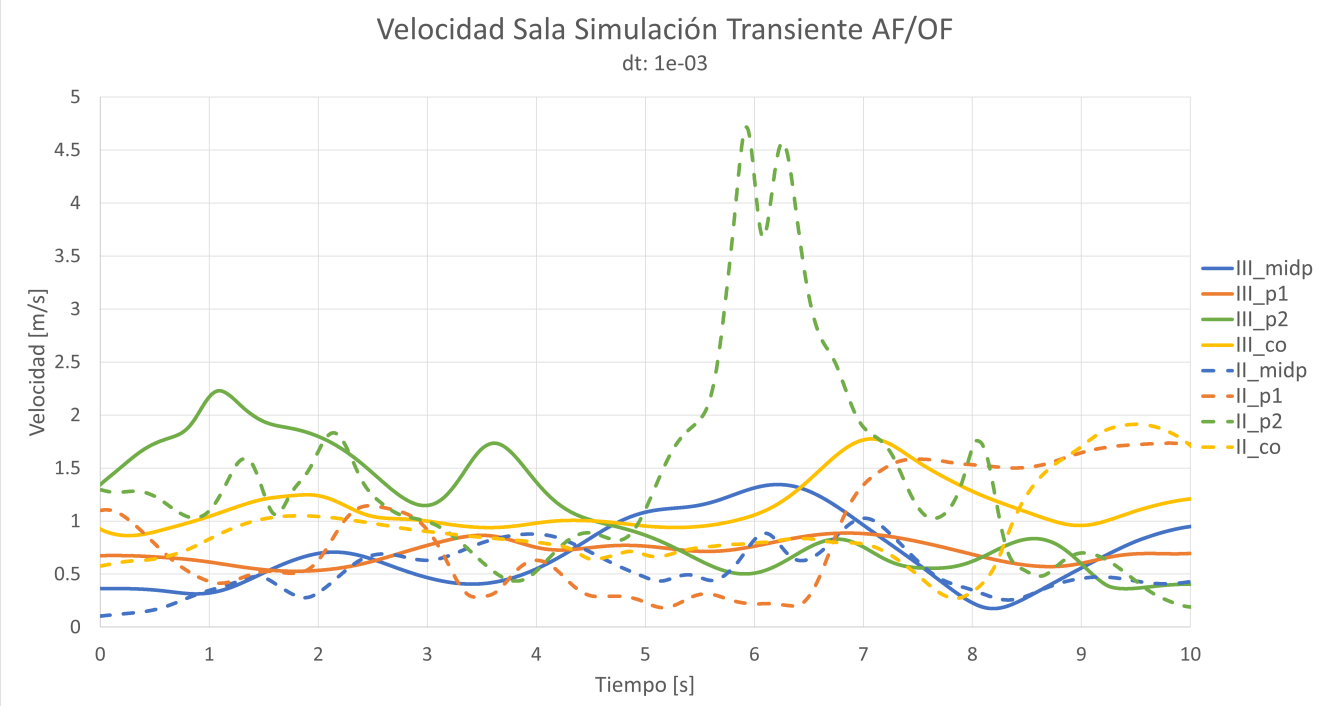


Figura B.7: Velocidad simulación II y III

Anexo C

Metodología de uso OpenFOAM

Metodología OpenFOAM v7

Instructivo para la realización de simulaciones CFD orientadas a ventilación industrial

Pablo Pogorelow Morales
Santiago de Chile, 2021

Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Mecánica

C.1. Introducción

Como parte de la realización de un análisis comparativo de las herramientas de simulación CFD ANSYS Fluent y OpenFOAM en un caso de ventilación industrial aplicado, existen una serie de aprendizajes tanto en el uso de estas herramientas como en las capacidades y desempeños mostrados por estos programas. Es de interés entonces, realizar una metodología de uso que permita a un usuario con conocimientos básicos de mecánica de fluidos computacional, utilizar el software OpenFOAM para realizar simulaciones computacionales logradas en otros programas del rubro.

Este documento es una recopilación de información y experiencias asociadas principalmente a la realización de un caso de estudio de ventilación modelada empleando el modelo de turbulencia K-epsilon, tanto para estado estacionario como para transiente, con y sin la resolución de ecuación de energía. Ha sido desarrollado utilizando la séptima versión de este software en dos sistemas operativos en base a la distribución linux, Ubuntu 18.04 y CAE Linux.

OpenFOAM carece de una interfaz de usuario, quizás la mayor diferencia con los programas comerciales más conocidos, sin embargo, el usuario podrá ver que la lista de de instrucciones para llevar a cabo una simulación es acotada y breve. Por otra parte, es posible automatizar instrucciones y modificar la totalidad de los archivos empleados en el caso.

C.2. Estructura y funcionamiento

En esta sección se pretende dar cuenta del funcionamiento práctico de OpenFOAM, su estructura y los principales elementos presentes en su funcionamiento a la hora de modelar casos de ventilación con transferencia de calor. El lector encontrará que la estructura aquí descrita será una guía suficiente para implementar variados modelos numéricos y formulaciones de la mecánica de fluidos computacional, permitiendo expandir las aplicaciones descritas en este manual de uso.

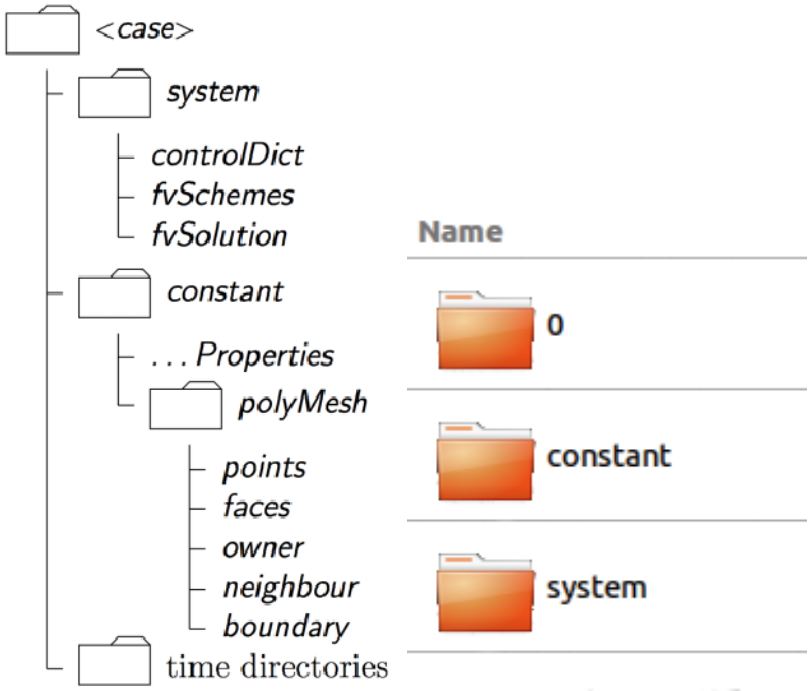
Quizás la mayor barrera de entrada de OpenFOAM está ligada a la ausencia de una interfaz de usuario como otros programas del rubro. Sin embargo, el usuario podrá ver que la lista de de instrucciones para llevar a cabo una simulación es acotada y breve, encontrando todos los comandos necesarios para ejecutar cada fase de la simulación en esta guía. Por otra parte, es posible automatizar instrucciones y modificar la totalidad de los archivos empleados en el caso de estudio, sin importar lo fundamental del archivo, el usuario tendrá acceso a la edición de este y la opción de realizar cualquier mejora que considere conveniente.

Finalmente, siendo que la visualización de resultados y el post procesamiento es posible realizarla de numerosas formas, la instalación del programa ofrece adicionalmente la adquisición de otro

programa de código libre diseñado para estas funciones, ParaView. En esta sección se revisa la importación y/o lectura de datos, el post procesamiento es revisado en mayor detalle en el capítulo C.8.

C.2.1. OpenFOAM

Independiente del proceso de instalación implementado por el usuario, la estructura de OpenFOAM aquí descrita será la base para cualquiera de sus implementaciones. En primera instancia se aconseja determinar con precisión el objetivo, características y funciones requeridas para el proceso de simulación. Para este proceso se sugiere al usuario hacer uso de lo descrito en las secciones C.6, C.6. Luego de la determinación del *cómo* se simulará lo deseado, se deberá tener una carpeta principal del caso[1]. En esta carpeta, deberán estar al menos presentes tres sub carpetas: *system*, *constant* y *0*. Estos directorios contienen a su vez archivos mínimos para la operación de un caso en OpenFOAM. Tanto las carpetas como sus archivos se aprecian en la figura C.1.



(a) (b)

Figura C.1: Estructura general, carpetas principales

La carpeta *0* es el directorio que contiene las condiciones de borde e iniciales de cada parámetro a resolver por el modelo determinado por el usuario. Dentro del proceso de simulación se adicionan diversos directorios dependiendo de la configuración de registro de datos establecida por el usuario.

[1] Se recomienda generar nombres continuos, sin espacios entre palabras. OpenFOAM no es flexible en este aspecto.

El directorio *systems* contendrá aquellos archivos que controlan el proceso de simulación, es decir, incluye la información del solver empleado, esquemas de discretización empleados por el solver, parámetros de control de este y parámetros de simulación globales. Además de esta información, este es el directorio que contiene los archivos opcionales de procesamiento y post procesamiento.

Constant será la dirección a donde recurre el programa en búsqueda de la información de carácter constante de la simulación. Aquí se almacena el mallado a emplear, propiedades físicas, de transporte y aquellas requeridas por el modelo a utilizar, como lo serían las propiedades del modelo de turbulencia por ejemplo.

Los tres directorios aquí descritos representan la base del caso de OpenFOAM y **todos los archivos presentes en estos directorios son modificables en cualquier momento de la simulación** (a excepción del mallado). Es decir, si se desea modificar cierta propiedad del fluido, tolerancia del algoritmo, parámetro de registro, o cualquier otro archivo, es posible realizar dicha modificación y el guardar la nueva versión del archivo modificado, implicará automáticamente la modificación en el desarrollo de la simulación, esta posibilidad de modificación se conoce como modificación *on the fly*. La variación será aplicada en el siguiente paso de la resolución. Dicho lo anterior, es importante que el usuario comprenda los riesgos que esto implica. Un error en la modificación de cualquier parámetro o instrucción puede desembocar en la detención inmediata de la simulación.

C.2.1.1. Construcción de un caso

Como se mencionó anteriormente, para iniciar una simulación debe tenerse en cuenta una serie de objetivos en cuenta. Una vez determinado el método deseado a implementar para resolver el problema, es aconsejable construir la carpeta del nuevo caso a partir de un tutorial ya existente en OpenFOAM que contenga aquellos elementos buscados por el usuario. Dentro de los archivos del programa existirá un directorio que abarca los principales y más utilizados métodos de resolución ordenados por categorías. Para esta versión, la carpeta de tutoriales se encuentra dentro de la carpeta de instalación principal.

```
cd $FOAM_TUTORIALS$
```

Aquí se encuentran las siguientes secciones

```
Allclean    combustion      electromagnetics   IO          resources
Allrun      compressible    financial          lagrangian  stressAnalysis
Alltest     discreteMethods heatTransfer       mesh
basic       DNS             incompressible     multiphase
```

Dentro de estas opciones el usuario encontrará tutoriales específicos para diversos casos aplicados (con geometrías incluidas). Para la ejecucuín de cualquiera de estos casos se deberá copiar el archivo a una carpeta externa, luego se podrá ejecutar siguiendo los archivos y directrices de cada tutorial.

En el caso de un fluido incompresible el usuario tendrá las siguientes opciones para iniciar su carpeta de proyecto:

```
1  adjointShapeOptimizationFoam  pimpleFoam          simpleFoam
2  boundaryFoam                  pisoFoam            SRFPimpleFoam
3  icoFoam                       porousSimpleFoam    SRFSimpleFoam
4  nonNewtonianIcoFoam           shallowWaterFoam
```

A partir de este punto entonces, el usuario es capaz de encontrar el mejor punto de partida en la formulación de su propio proyecto. Así, importando su respectiva malla, imponiendo sus condiciones de borde y sus parámetros de control, será capaz de formular su propia simulación.

A continuación se presentan una serie de aspectos a tener en cuenta para comprender el funcionamiento de OpenFOAM. Se muestra y detallan los tipos de archivos mostrados en la figura C.1, su funcionamiento global y su estructura interna.

C.2.1.2. Carpetas temporales y estructura básica de archivos tipo FOAM

Dependiendo del propósito de la simulación se tendrán diversos parámetros a trabajar. Cada parámetro sea o no de interés, deberá tener un archivo *FOAM* con un nombre único, donde en su interior deberán ser especificadas las condiciones iniciales y de borde de dicho parámetro. En la figura C.2 de muestra el set de archivos para el caso de una simulación transiente con transferencia de calor y sin ella.

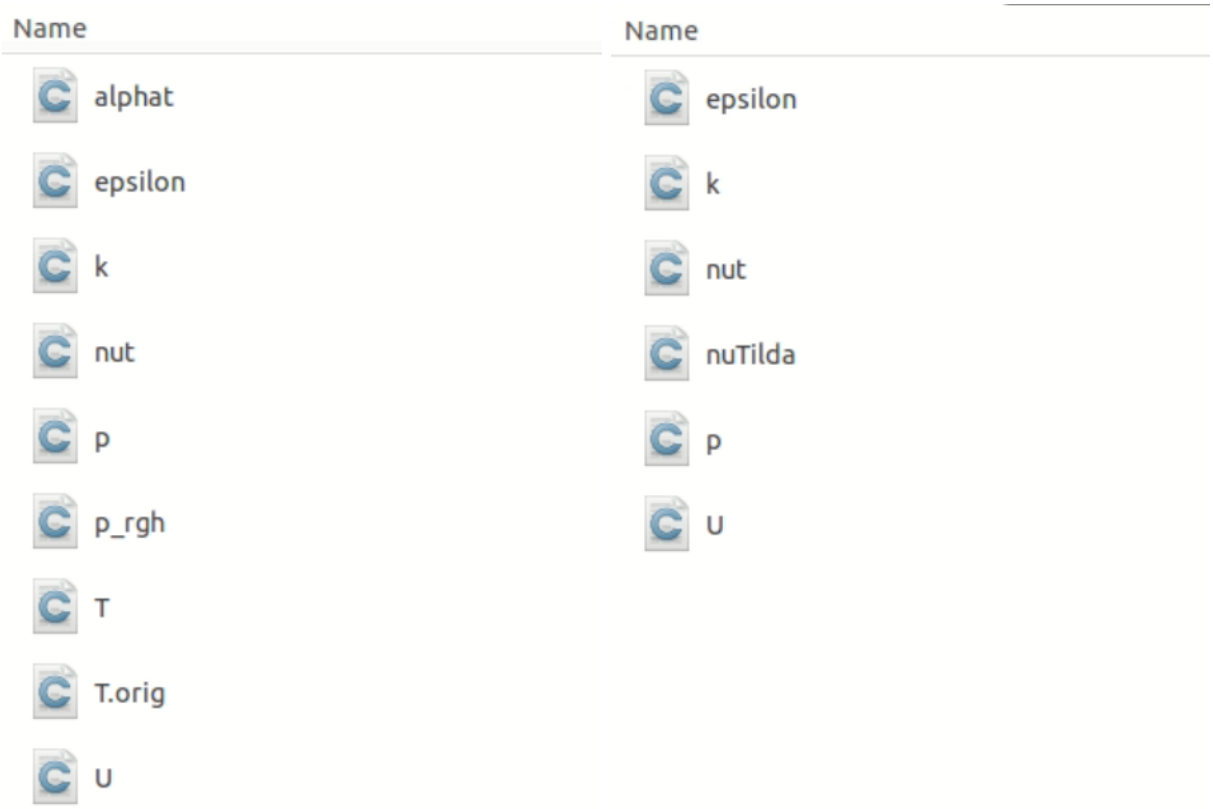
Para explicar la estructura de un archivo tipo *FOAM* se tomará como referencia el archivo de velocidad 'U' perteneciente al tutorial de flujo incompresible en estado transiente PitzDaily (código C.1). Para esta geometría se tiene una entrada (inlet), una salida (outlet), una pared superior (upperWall) y una pared inferior (lowerWall). **Por convención, todos los nombres y comandos en OpenFOAM se escriben unidos diferenciando con letra capital la palabra siguiente de la antecesora**. En este caso el objeto o parte (patch en inglés) de nombre *frontAndBack* establece que la geometría es bidimensional.

En el código C.1 se diferencian cinco secciones: Encabezado, diccionario principal, dimensiones, campo interno o condición inicial y un diccionario de condiciones de borde. Los diccionarios se identifican mediante los paréntesis de corchete, mientras que para las dimensiones se establecerá un arreglo designado por corchetes cuadrados. Finalmente, cuando una variable presente más de una componente, como el caso de la velocidad, se tendrán paréntesis curvos como los mostrados en las lineas 15 y 21. A continuación se ahonda un poco más en estos elementos.

- Encabezado: Parte inicial del archivo, da cuenta de la versión del programa que se está utilizando.
- Diccionario principal: Establece el tipo de archivo FOAM y entrega información respecto al

tipo de variable presente en el archivo.

OpenFOAM presenta varias ediciones y con el desarrollo del software ciertas mejoras han modificado la implementación ded variables, solvers y archivos de diversa índole. La versión será un indicador de la versión del archivo en tanto al global de versiones de OpenFOAM.



(a) buoyantPimpleFoam (b) pimpleFoam

Figura C.2: Ejemplo de archivos carpeta *0*

La clase del archivo determinará el tipo de variable que representa el documento. En este caso *volVectorField* es la representación de una variable de carácter vectorial. En el caso de la presión por ejemplo, el usuario encontrará una clase del tipo escalar (*volScalarField*). En general para el resto de archivos presentes en el caso se tendrá la clase *dictionary*, representando netamente un diccionario de información.

La última variable del diccionario FoamFile establece el nombre del parámetro del archivo en cuestión. Ambos nombres irán siempre en sintonía.

- Dimensiones: Este arreglo dará cuenta de las dimensiones que presenta la variable representada por el archivo. Las dimensiones son acordes al sistema internacional de medidas y su distribución en el arreglo se presenta de la siguiente forma: $\mathrm{kg}^{\alpha}\mathrm{m}^{\beta}\mathrm{s}^{\gamma}\mathrm{K}^{\delta}\mathrm{mol}^{\epsilon}\mathrm{A}^{\zeta}\mathrm{cd}^{\eta}$

  El arreglo de dimensiones deberá ser llenado con las potencias de las unidades. Es decir, para

el caso de la velocidad se tiene β igual a 1 y γ igual a -1, ya que las unidades de esta variable son m^1 y s^{-1}. De esta forma se construye el arreglo mostrado en la linea 14 del siguiente código.

Es de suma importancia prestar atención a las unidades que emplea cada solver y por ende cada variable de dimensiones de cada parámetro a utilizar. Existirán claras diferencias de unidades, incluso para una misma variable, al emplear un solver distinto en OpenFOAM.

- Campo interno: El campo de valores inicial del parámetro en cuestión, estará determinado por la variable *internalField*. Aquí se deberá respetar la naturaleza de la variable tal como se comenta anteriormente en la definición de clase.

- Diccionario de condiciones de borde: Este diccionario presenta un número de entradas igual a la cantidad de objetos generados por el proceso de mallado. Cada objeto tendrá asignada una condición acorde al requerimiento de la variable del archivo principal. Siguiendo el ejemplo del código anterior, se aprecia que para cada objeto habrá un diccionario adicional con al menos la entrada *type*. En este caso la entrada del fluido está impuesta mediante una velocidad constante de $10[m/s]$ en la dirección de la componente x; la condición de borde a la salida se impone como gradiente cero y para todas las paredes de la geometría se impone la condición de deslizamiento nulo o *noSlip*.

Código C.1: Archivo FOAM de velocidad

```
1
2  \*---------------------------------------------------------------------------*/
3  FoamFile                    {
4     version     2.0;
5     format      ascii;
6     class       volVectorField;
7     object      U;            }
8  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
9  dimensions      [0 1 -1 0 0 0];
10 internalField   uniform (0 0 0);
11 boundaryField{
12    inlet    {
13       type          fixedValue;
14       value         uniform (10 0 0);    }
15    outlet    {
16       type          zeroGradient;    }
17    upperWall    {
18       type          noSlip;    }
19    lowerWall    {
20       type          noSlip;    }
21    frontAndBack    {
22       type          empty;    }
23 }
24 // ************************************************************************* //
```

No está demás notar que cada linea de texto culmina con un punto y coma ';'. No respetar esta

norma se incurre en problemas de lectura del archivo y en futuros errores al momento compilar la simulación.

C.2.1.3. Constant

Dependiendo de la estructura del caso a simular, esta carpeta podrá almacenar una serie de archivos del programa. Como se menciona en un inicio, los archivos aquí presentes son aquellos que no presentan modificaciones en el transcurso de la simulación. Este directorio almacenará al menos el mallado de la geometría a simular, un archivo con las propiedades de transporte y otro con las propiedades de turbulencia del modelo.

- PolyMesh: En esta carpeta se almacena la malla a utilizar. OpenFOAM requerirá de una serie de archivos tipo FOAM que den cuenta del mallado, no emplea un único archivo como otros programas. Más información respecto a este punto se encontrará en la sección C.4.

- Turbulence properties: En este archivo se especificará el tipo de modelación de turbulencia a realizar. A grandes rasgos se separará en RAS (Reynold Average) y LES (Large Eddy Simulation). Dentro de las opciones más comunes de la primera familia se encuentran: kEpsilon, realizableKE, kOmega, kOmegaSST, v2f.

  En este archivo también se especifica la impresión en consola de los coeficientes del modelo.

- Transport properties: En OpenFOAM se define la viscosidad cinemática ν como parámetro de transporte bajo la relación C.2.1.3 . Esta definición remplaza lo que se aprecia en otros programas por separado como son la densidad (ρ) y la viscosidad μ.

$$\nu = \frac{\mu}{\rho} \tag{C.1}$$

- Thermophysical properties: Este archivo contendrá la información necesaria adicional a las propiedades de transporte para hacer uso de modelaciones con transferencia de calor. Aquí se definen parámetros como Cp, peso molar, número de Prandtl, entre otros. También se describe el método de cálculo para resolver la ecuación de energía, más información respecto a esto se encuentra en la sección C.6.

- Gravedad: Dentro del abanico de archivos presentes en esta carpeta se podrá almacenar un archivo de nombre g, cuya finalidad será proporcionar al modelo el dato numérico de la aceleración de gravedad y la orientación que debe tener esta acorde a la disposición de la geometría.

Código C.2: Ejemplo implementación aceleración de gravedad

```
1    FoamFile  {
```

```
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    location    "constant";
    object      g;   }
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 1 -2 0 0 0 0];
value           (0 0 -9.81);
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

C.2.1.4. System

Los archivos dispuestos en esta carpeta controlan todo el proceso de simulación. Los archivos aquí almacenados son las herramientas de las que dispone el usuario para poner en ejecución, controlar y registrar el proceso de simulación. Los archivos mínimos a encontrar en este directorio serán: *controlDict*, *fvOptions* y *fvSchemes*.

A continuación se hará un desglose de las variables presentes en cada uno de estos archivos, con el fin de conocer exactamente qué implica cada cada una y sus posibles variaciones. Se muestran extractos de códigos reales empleados en simulaciones de ventilación industrial.

- *controlDict*: Es la sala de control de la simulación. Determina qué solver se emplea, el control de tiempo, registro de datos y funciones externas y definidas por el usuario. Si vemos el código a continuación nos encontramos con que la *application* será el solver empleado por el programa, el mismo que será llamado a través de la terminal al momento de ejecutar la simulación. El resto de entradas se explica en las tablas C.1 y C.2.

Código C.3: Ejemplo diccionario de control para buoyantSimpleFoam

```
FoamFile    { (...) }
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
application     buoyantSimpleFoam;
startFrom     latestTime; // variación común: startTime
startTime     500;
stopAt        endTime;
endTime       3000;
deltaT           1;
writeControl          timeStep; // variación común: adjustableRunTime
writeInterval         100;
purgeWrite            0;
writeFormat           ascii;
writePrecision        6;
writeCompression      off;
timeFormat            general;
timePrecision         6;
runTimeModifiable     true;
adjustTimeStep        yes;
maxCo                 5;
functions    {
    #includeFunc  probes
    #includeFunc  residuals    }
```

Tabla C.1: Complemento controlDict: Tiempos de inicio y término

Entrada	Valor	Descripción
application	-	Cualquier solver disponible en OF
startFrom	firstTime	El primer tiempo encontrado en los registros
	startTime	Tiempo específico
	latestTime	Tiempo más reciente encontrado
startTime	número	Valor de inicio
stopAt	endTime	Valor de término
	writeNow	Detiene la simulación al terminar el tiempo actual y registrar valores
	noWriteNow	Detiene la simulación al terminar el tiempo pero no registra
	nextWrite	Detiene la simulación al siguiente periodo de registro
endTime	número	Valor de término
deltaT	número	Paso de tiempo de la simulación

Tabla C.2: Complemento controlDict: Registro y opciones adicionales

Entrada	Valor	Descripción
writeControl	timeStep	Registro de datos de intervalo timeStep
	runTime	Registro de datos cada runTime de simulación
	adjustableRunTime	Usado en timpo variable. Registra según el intervalo de tiempo especificado
	cpuTime	Registro cada cpuTime segundos
	clockTime	Registro de datos cada clockTime segundos de tiempo real
writeInterval	número	parámetro usado para writeControl
purgeWrite	número	Número de registros consecutivos en proceso de sobreescritura
writeFormat	ascii	Formato ASCII
	binary	Formato binario
writePrecision	integer	Parámetro usado para writeFormat
writeCompression	on/off	Establece compresión de datos al registrarse
timeFormat	fixed	Los registros se realizan con la precición de timePrecision
	scientific	Registros en notación científica
	general	Especifica el tiempo en notación científica si es muy pequeño
timePrecision	integer	Parámetro usado para timeFormat
graphFormat	raw	Valor por defecto: registra en ASCII
	gnuplot	Registra en formato gnuplot
	xmgr	Registra en formato xmgr
	jplot	Registra en formato jplot
adjustTimeStep	yes/no	Habilita la modificación del paso de tiempo
maxCo	número	Número máximo de Courant
runTimeModifiable	true/false	Hablilita la modificación de parámetros on the fly
libs	Lista	Lista de librarías externas a importar
functions	diccionario	Diccionario de funciones adicionales, creadas in situ o importadas

- *fvSchemes*: El prefijo *fv* corresponde a 'finite volume', implicando que este archivo es quien contiene los esquemas de discretización del método de volúmenes de control empleado por OpenFOAM. OpenFOAM realiza una discretización para cada término de las ecuaciones que resuelve el solver principal de la simulación, es decir, dependiendo de cada caso se tendrán que discretizar diversas combinaciones de variables.

  Si el usuario realiza el procedimiento recomendado empezando desde un tutorial proporcio-

nado por OpenFOAM para la construcción del caso de estudio, se encontrará con ajustes menores a realizar en este documento. Si por el contrario, se no se encuentran explícitamente las combinaciones de discretizaciones necesarias por el solver, openFOAM generará una alerta en la terminal exigiendo la inclusión de este término en el diccionario correspondiente.

Código C.4: Ejemplo esquemas de discretización para transferencia de calor

```
FoamFile    { (...) }
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
ddtSchemes    {
#ifeq \${FOAM_APPLICATION}   buoyantSimpleFoam
 default       steadyState;
#else
 default       Euler;
#endif   }

gradSchemes    {
   default        cellMDLimited leastSquares 1.0;    }

divSchemes    {
#ifeq \${FOAM_APPLICATION}   buoyantSimpleFoam
 default          none;
 div(phi,U)        bounded Gauss linearUpwind grad(U);
 div(phi,K)        bounded Gauss linear;
 div(phi,h)        bounded Gauss upwind;
 div(phi,k)        bounded Gauss upwind;
 div(phi,epsilon)  bounded Gauss upwind;
 div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
#else
 default          none;
 div(phi,U)        Gauss linearUpwind grad(U);
 div(phi,k)        Gauss upwind;
 div(phi,epsilon)  Gauss upwind;
 div(phi,R)        Gauss upwind;
 div(R)           Gauss linear;
 div((nuEff*dev2(T(grad(U))))) Gauss linear;
 div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
 div(phi,K)      Gauss linear;
 div(phi,Ekp)    Gauss linear;
 div(phi,h)      Gauss upwind;
#endif    }

laplacianSchemes    {
   default        Gauss linear limited 1.0;    }

interpolationSchemes    {
   default        linear;    }

snGradSchemes    {
   default        limited corrected 1.0;    }
```

En la práctica, el usuario puede hacer uso de más de un solver en un mismo caso. Para mantener el orden en la estructura del caso, se recomienda programar una condición sencilla en aquellos diccionarios que no compartan todas las discretizaciones a emplear. El código a

continuación presenta una buena guía de esta alternativa, buscando que el usuario no deba intercambiar constantemente un documento de esquemas al emplear buoyantSimpleFoam y su complemento en transiente buoyantPimpleFoam. OpenFOAM no restringe el empleo de diversos esquemas de discretización, en las tablas C.3 y C.4 se presenta un resumen de las opciones más útiles y sus principales características.

A continuación se desglosa el código anterior en cuanto a las opciones de discretización que presenta OpenFOAM. Es importante notar que ciertas entradas en los diccionarios presentan términos por defecto y otros no, esta decisión quedará supeditada a las intenciones del usuario. En cualquier diccionario además de presentar términos por defecto se pueden adicionar discretizaciones específicas, las cuales sobre escribirán la condición por defecto del nuevo parámetro.

- Esquemas de tiempo *timeSchemes*: Los esquemas temporales representan la discretización de las derivadas temporales de primer y segundo orden $\partial/\partial t, \partial^2/\partial^2 t$.
  - *steadyState*: En estado estacionario las derivadas temporales son seteadas a cero.
  - *Euler*: En transiente, primer orden implícito no acotado
  - *CrankNicolson*: En transiente, segundo orden implícito acotado por el parámetro ψ, donde ψ igual a 0 corresponde a Euler. Generalmente es seteado a $\psi = 0.9$
  - *localEuler*: Pseudo transiente utilizado para acelerar la solución en estado estacionario. Primer orden implícito.
- Esquemas de gradiente *gradSchemes* ∇: La entrada típica de discretización de los gradientes es la discretización Gausiana, sin embargo el desarrollo empírico muestra que otros esquemas presentan mayor estabilidad ante mallas no estructuradas. En la tabla C.3 se detallan los principales métodos para discretizar el término del gradiente.

  A grosso modo *leastSquares* realiza una discretización de segundo orden empleando mínimos cuadrados con todas las celdas vecinas.
- Esquemas de divergencia *divSchemes* $\nabla\cdot$: Este diccionario determina la discretización de todos los términos de la forma $\nabla\cdot$. Aquí se incluyen tanto los términos advectivos como los difusivos, teniendo en consideración la relevancia de estas discretizaciones es que normalmente no se impone una discretización por defecto.

  La mayoría de los esquemas son basados en la integración gaussiana, usando el flujo ϕ y el campo de advección para interpolar las celdas en virtud de las caras adjacentes a un nodo mediante el esquema escogido. Para los esquemas de este diccionario se suele complementar el abanico de posibles elecciones con términos específicos que buscan acotar las interpolaciones (*bounded*) y evitar la divergencia en el proceso de cálculo. En la tabla C.4 se encuentran las opciones típicas para problemas de turbulencia como el abordado en esta guía, sin embargo el espectro es mucho mayor.

- Esquemas de laplaciano *laplacianSchemes* ∇^2: Los términos típicos a discretizar por este diccionario corresponden a los términos de difusión de las ecuaciones de momentum. Estos términos del estilo $\nabla \cdot (\nu \nabla \mathbf{U})$ son discretizados empleando el esquema Gausiano en conjunto con un esquema de interpolación y un término de interpolación de gradiente normal.
- Esquemas de interpolación *interpolationSchemes*: Este diccionario determina la interpolación entre valores alojados en los centroides de las celdas y sus caras adjacentes. Son contados los casos en donde se emplea un sistema de interpolación que no sea lineal (The OpenFOAM Foundation, 2021).
- Esquemas de gradiente normal *snGradSchemes*: Este diccionario contiene los términos de gradiente normales a la superficie. Este tipo de términos son empleados en los esquemas laplacianos dado que se requieren para evaluar estos términos en la integración Gausiana.

En las tablas C.3 y C.4 se establecen los principales esquemas de primer y segundo orden empleados en la discretización de términos. Muchos de los esquemas de convección (divergencia) se basan en los métodos de Variable Normalizada (NV) y Disminución de variación total (TVD). Estos métodos proporcionan una mezcla entre un esquema de bajo orden y uno de orden mayor gracias a una variable de limitación entre ambos (parámetro opcional para el usuario).

Para los esquemas de divergencia el acotamiento de valores está garantizado solo para problemas de una dimensión, sin embargo se mejora este aspecto en problemas 2D y 3D acotando el término del gradiente. Junto con esto, en estado estacionario se suele anteponer el término *bounded* que a grosso modo, ayuda con la convergencia al orientar el cálculo a un valor acotado (OpenCFD Ltd 2019, 2019).

Tabla C.3: Principales esquemas de discretización para términos de gradiente

Nombre	Descripción
Gauss	Gradiente de la celda calculado utilizando el teorema de Gauss
Least-squares	Gradiente calculado empleando método de mínimos cuadrados
Cell-limited	Limita el cálculo del gradiente a los valores de las celdas vecinas empleando un coefficiente de 0 a 1. Donde 1 significa máximo acotamiento de valores.
Multi-directional CL	Equivalente al método anterior solo que incluye a todas las direcciones posibles
Face-limited	Limita el cálculo del gradiente a los valores de las caras adjacentes al centroide. Emplea un coefficiente de 0 a 1. Donde 1 significa máximo acotamiento de valores.
Multi-directional CL	Equivalente al método anterior solo que incluye a todas las direcciones posibles

Tabla C.4: Principales esquemas de discretización para términos de divergencia

Nombre	Descripción
Limited linear	Primer y segundo orden, no acotado
Linear	Segundo orden, no acotado. Empleado en LES y para mallas isotrópicas
Linear-upwind	Segundo orden, no acotado. Usa upwind para la interpolació con corrección explicita basada en el gradiente de cada celda
Mid-point	Primer y segundo orden, no acotado. Equivale a Linear en mallas isotrópicas
MUSCL	Segundo orden, no acotado. Monotone Upstream-centered Scheme for Conservation Laws
QUICK	Segundo orden, no acotado. Quadratic Upstream Interpolation for Convective Kinematics
UMIST	Segundo orden, no acotado. Upstream Monotonic Interpolation for Scalar Transport
Upwind	Primer orden, acotado. El valor de la cara de la celda es calculado según la dirección de flujo y la celda adjacente
Van Leer	Segundo orden, no acotado.

- *fvSolution*: Este diccionario al igual que los anteriores estará supeditado a la elección del solver empleado por la simulación. Aquí, cada parámetro a resolver en las ecuaciones gobernantes del caso deberá tener un método de resolución numérico, junto con esto se establecerán los algoritmos de resolución a utilizar de forma global (SIMPLE, PISO, etc) y sus parámetros de control. Por último se encontrarán una serie de opciones adicionales de control, siendo los factores de relajación quizás el parámetro de mayor importancia.

  A continuación se presenta un extracto de un archivo *fvSolution* empleado en conjunto con los archivos expuestos en las secciones anteriores. En el primer segmento se puede ver el sub diccionario de la variable de presión dentro de *solvers*. Aquí, se distinguen los siguientes parámetros:

  - solver: Este parámetro corresponde al solver lineal empleado en resolver la ecuación matricial donde está presente la variable de interés. La elección de este solver estará principalmente determinada por la naturaleza de la variable y su simetría matricial en la ecuación a resolver. El programa hace la distinción entre matrices simétricas y asimétricas, si el usuario emplea un solver de matriz simétrica para una matriz asimétrica se obtendrá un error indicando para qué variable el solver no es el adecuado.

    En la práctica un solver lineal para matrices asimétricas puede ser empleado para la resolución matriz simétricas, existen diferencias en el tiempo de cómputo y en la estabilidad numérica de la solución. El usuario deberá buscar aquel solver lineal que mejor se ajuste al tipo de variable a resolver. Entre las principales opciones de solvers lineales se encuen-

tran los Preconditioned conjugate gradient (PCG) y los Smooth solvers. Los primeros requieren la utilización de un precondicionador de matrices, mientras que los últimos un suavizador. Dependiendo la elección del usuario las opciones serán las siguientes:

Preconditioner:

- DIC/DILU: Diagonal incomplete-Cholesky, emplea la decomposición de Cholesky para matrices simétricas y LU para matrices asimétricas.
- FDIC: Faster diagonal incomplete-Cholesky.
- diagonal: Precondicionador diagonal, diagonaliza la matriz.
- GAMG: Geometric-algebraic multi-grid. Suele ser el mejor solver a empĺear en el cálculo de la presión.

Smoother

- GaussSeidel: Emplea el método de Gauss-Seidel para la resolución del sistema de ecuaciones. Junto con symGaussSeidel es de los solver más utilizados para el cálculo de la variable de velocidad.
- symGaussSeidel: Método de Gauss-Seidel en matrices simétricas.
- DIC/DILU: Diagonal incomplete-Cholesky, emplea la decomposición de Cholesky para matrices simétricas y LU para matrices asimétricas.
- DICGaussSeidel: Emplea Diagonal incomplete-Cholesky junto con el método Gauss-Seidel tanto para matrices simétricas como asimétricas.

Existen tres formas de detener la iteración de un solver. Se puede lograr un nivel de precisión tal que el nivel de residuos de la operación sea menor que el nivel de tolerancia *tolerance*; la razón entre el residuo inicial respecto al logrado a través de las sucesivas iteraciones sea menor a una tolerancia relativa *relTol* o bien se puede alcanzar un máximo de iteraciones (*maxIter)* intentando cumplir con el nivel de precisión estipulado por el usuario.

En la práctica, condiciones más permisivas (tolerancias y tolerancias relativas altas) implican que el proceso se ejecute más rápido, sin embargo, se condiciona la estabilidad numérica del proceso. Se aconseja ser estricto con aquellos parámetros que impliquen otros cálculos dentro del algoritmo y ante una muestra de estabilidad de la solución, de a poco ir relajando las tolerancias si es que se busca mayor velocidad de cómputo.

En el lenguaje $C++$ se puede llamar a una variable empleando el caracter \$. Como se aprecia en la linea 10 del siguiente código, se define un sub diccionario del mismo parámetro *p_rgh* con la palabra FINAL. Esta definición especifica la última resolución numérica de la variable *p_rgh* en cada loop de iteración. Es decir, si el algoritmo general estipula 4 ciclos de resolución de cierta variable, el último ciclo (*p_rghFINAL* en este caso) se define como el

diccionario *p_rgh* con el parámetro *relTol* sobreescrito en 0. De esta forma se busca asegurar la convergencia antes de continuar con el siguiente paso del algoritmo.

Código C.5: Ejemplo fvSolution: Parámetros

```
1  FoamFile  { (...) }
2  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
3  solvers  {
4      p_rgh      {
5      solver          GAMG;
6      tolerance       1e-8;
7      relTol          0.01;
8      maxIter         1000;
9      smoother        DICGaussSeidel;       }
10     p_rghFinal      {
11       \$p_rgh;
12       relTol         0;  } }
```

El algoritmo general o método de resolución también se detalla en esta sección. Dependiendo de la elección del usuario (para problemas RANS suele ser SIMPLE, PISO o PIMPLE), en un nuevo sub diccionario se especificarán las condiciones de operación y ciertos parámetros propios de cada algoritmo. Por ejemplo, en el caso de utilizar el algoritmo PIMPLE, el usuario deberá especificar la cantidad de correctores (loop de resolución de presión) que desea implementar. En caso del algoritmo SIMPLE, el usuario deberá especificar si desea emplear SIMPLEC a través de la variable *consistent*.

Como se aprecia a continuación, dentro del sub diccionario del algoritmo se encuentran también un diccionario de control de residuos. Aquí se determina el nivel de precisión de cada variable a resolver por los solvers llamados por el algoritmo principal. Las entradas de parámetros (al igual que el los otros archivos FOAM) pueden ser nominadas en conjunto o bien por separado, tal como se aprecia en la linea 10, donde se comparte el mismo valor para los residuos de k y ϵ.

Código C.6: Ejemplo fvSolution: Algoritmo general

```
1  SIMPLE  {
2    consistent  yes;
3     momentumPredictor no;
4     nNonOrthogonalCorrectors 2;
5
6     residualControl      {
7        p_rgh       1e-4;
8        U           1e-4;
9        h           1e-4;
10       "(k|epsilon)" 1e-3;  }  }
```

Por último, a continuación se presenta el último diccionario típico en el archivo $fvSolution$. Aquí se define un sub diccionario con los factores de relajación presentes en las ecuaciones a resolver. Es posible generar entradas de parámetros externos a las ecuaciones, sin embargo

este no es el caso. Se puede en la linea 8 el término .∗, este término determina que para cualquier variable cuyo factor de relajación no se encuentre dentro del diccionario, adoptará el valor 0.9.

Código C.7: Ejemplo fvSolution: Otras opciones de control, factores de relajación

```
1
2       relaxationFactors   {
3       equations       {
4          p_rgh        0.3;
5          U            0.7;
6          k            0.7;
7          epsilon      0.7;
8          ".*"         0.9;   }  }
```

C.2.2. ParaView

El programa ParaView es un programa multi-plataforma de análisis y visualización de datos. Es importante hacer énfasis en que no solo opera en conjunto con OpenFOAM, si no que con un abanico mucho mayor de programas. Dicho esto, ParaView requiere que el usuario le proporcione la información a ser graficada o analizada y gestionada posteriormente.

En pos de construir una estructura y no confundir al lector, ParaView requiere que el usuario le proporcione tanto el mallado como los pasos de tiempo del caso de estudio. Este proceso de importación no es automático. Si bien ParaView es capaz de visualizar datos mientras que la simulación está en curso, el proceso de importación debe realizarse manualmente cada vez[2].

Para hacer posible la visualización del caso mediante ParaView el programa debe ser ejecutado en la carpeta principal de la simulación mediante el siguiente comando:

```
paraFoam
```

Este comando creará un archivo temporal de extensión *"nombre_carpeta.OpenFOAM"* y a continuación se abrirá el programa.

Si por alguna razón el comando anterior no funciona, es necesario crear un archivo de extensión *.foam* para luego poder abrir ParaView y así proceder con la importación de los datos. Para crear un archivo vacío que cumpla con esto puede utilizarse el comando *touch* en linux. Luego se debe arrancar la aplicación y abrir el archivo creado.

```
touch nombre_alternativo.foam
paraview
```

Una vez abierto el programa con un archivo temporal, se está en condiciones de importar la malla del caso de estudio. Para esto, debe estar disponible dicha información en el directorio */constant/polymesh*. ParaView generará la opción de aplicar la malla al canvas de visualización tal como se presenta en la figura C.3.

[2] No se descarta que al implementar OpenFOAM a través de python o algún otro gestor externo este proceso si pueda automatizarse, sin embargo para la realización de este manual aquella opción se ve descartada.

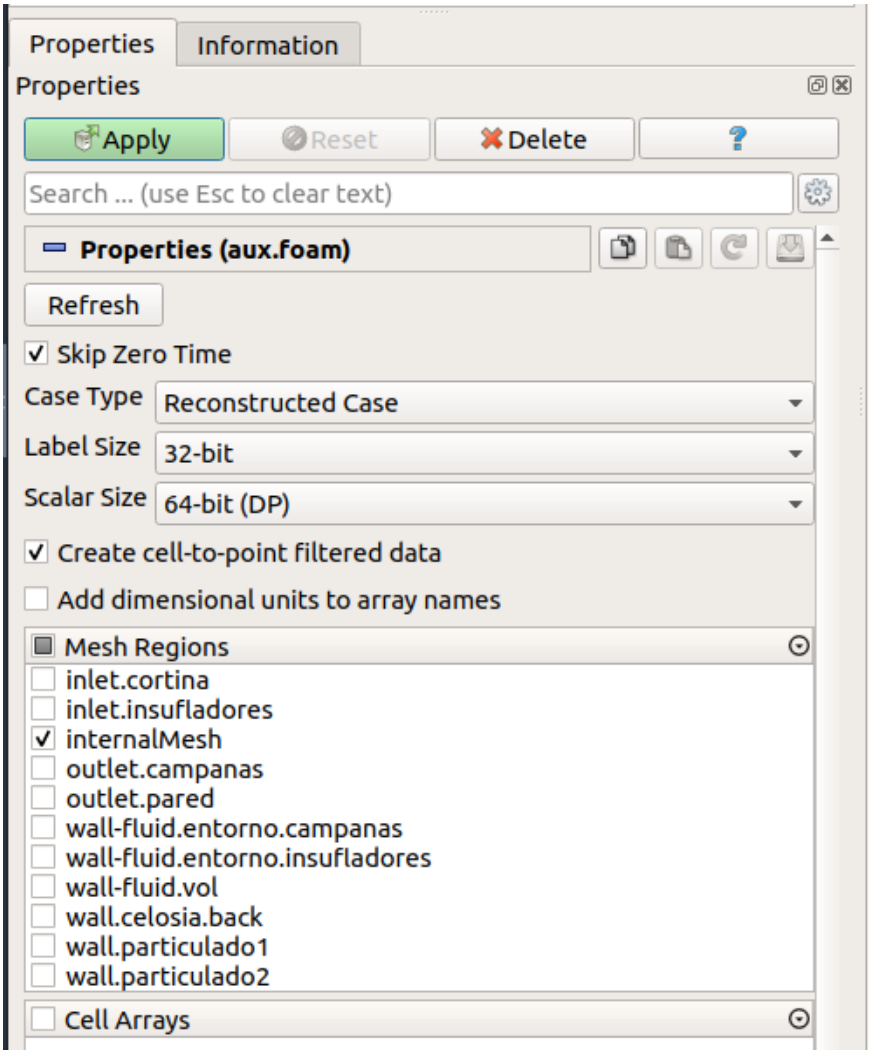


Figura C.3: Menú de propiedades e información ParaView

En la figura anterior es posible notar que ParaView ha identificado cada región de nuestra malla. Cada región mostrada otorga la posibilidad de selección al momento de importar la información. Se recomienda al usuario tener cuidado al seleccionar las entradas a importar, ya que en el proceso de visualización de datos, es posible confundir aquellos valores presentes como condiciones de borde con aquellos obtenidos producto del cálculo en la simulación.

Finalmente, al presionar la opción de aplicar, se cargarán en el programa todos los espacios de tiempo disponibles para visualización.

C.3. Instalación OpenFOAM

En esta sección de detalla el procedimiento para instalar y configurar OpenFOAM v7 en el sistema operativo Ubuntu o sistema semejante de 64 bit. En la sección C.3.1 se aborda el proceso de instalación junto con algunos pasos adicionales y problemas típicos. Se debe contar con permisos de administrador y/o superusuario (sudo) junto con acceso a internet. La Información aquí presentada es provista por *OpenFOAM Foundation* a través de su sitio web.

C.3.1. Instalación V1

Al momento de realizar este manual, la siguiente instalación está disponible para las siguientes versiones de Ubuntu: 16.04 LTS, 18.04 LTS, 19.04, 19.10 y 20.04 LTS.

1. En una terminal del sistema se debe ingresar el siguiente código una única vez. Estos comandos buscan habilitar la lectura de paquetes y claves a través de la herramienta *apt*. Al finalizar el proceso, el software quedará instalado en la carpeta por defecto de los proogramas externos al sistema operativo, la dirección */opt*.

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

2. Luego se debe actualizar *apt* con los nuevos permisos de acceso, para eso ejecutar la siguiente linea de comandos:

```
sudo apt-get update
```

3. Para instalar, ejecutar el siguiente comando. Se instalará tanto OpenFOAM como el complemento de Paraview.

```
sudo apt-get -y install openfoam7
```

C.3.1.1. Configuración usuario

Para que el programa pueda emplearse a través de cualquier terminal es necesario modificar el archivo *bashrc* del sistema. De esta forma OpenFOAM queda habilitado para su uso desde que arranca el sistema operativo a través de cualquier terminal.

1. Se abre el archivo bashrc presente el la carpeta origen del sistema operativo. Se emplea *gedit* como gestor de archivos de texto por default, pero el usuario puede hacer uso de cualquier programa de este estilo.

```
gedit ~/.bashrc
```

   Una vez abierto el archivo se debe agregar la siguiente linea de texto al final de este. Si existe un comando similar (de alguna versión anterior por ejemplo) debe eliminarse o en su defecto comentar dicha linea de texto.

```
source /opt/openfoam7/etc/bashrc
```

2. Para comprobar que la instalación funciona se llama a algún solver de OpenFOAM, por ejemplo:

```
simpleFoam -help
```

C.3.1.2. Mantención recurrente

OpenFOAM recibe constantes actualizaciones y ajustes, por lo que una buena práctica es realizar su actualización. Para esto se deben ejecutar los siguientes comandos:

```
sudo apt-get update
sudo apt-get install --only-upgrade openfoam7
```

C.3.1.3. Posibles problemas

- Si luego de la instalación en una máquina virtual ParaView muestra pantalla en blanco, se aconseja usar un escritorio alternativo como Gnome Classic o MATE. Se muestra continuación el comando de instalación del primero. Para hacer efectiva la instalación se debe reiniciar la máquina virtual.

```
sudo apt-get install gnome-panel gnome-flashback gnome-session-flashback indicator-applet-appmenu
```

- Se reportan algunos errores durante la instalación cuyos mensaje se encuentran a continuación:

```
Some packages could not be installed. This may mean that you
have requested an impossible situation...

The following information may help to resolve the situation:

The following packages have unmet dependencies: openfoam7 :
Depends: csh but it is not installable...
```

  La razón probable es que no estén debidamente habilitados los repositorios *universe*. Para habilitarlos, ejecutar el siguiente comando:

```
sudo apt-add-repository universe
sudo apt-get update
```

- Si MPICH está instalado en el sistema, durante la configuración de usuario puede presentarse el siguiente error: "*gcc: error: unrecognized command line option '–showme:link'*". Ante esto se debe localizar la instalación de MPICH y luego setear otro MPI por default. Para esto, ejecutar los siguientes comandos:

```
sudo update-alternatives --list mpi
sudo update-alternatives --set mpi /usr/lib/openmpi/include
```

C.3.2. Instalación V2

La siguiente guía de instalación se propone como alternativa a los pasos mostrados anteriormente. La guía expuesta a continuación es compleja (por la cantidad de pasos) y de lenta ejecución, sin embargo presenta una alternativa a cualquier problema de funcionamiento que pueda presentarse

con el método anterior. Permite operar con varias versiones de OpenFOAM en un mismo sistema operativo. Los pasos aquí presentados son provistos por *OpenFOAM Wiki* a través de su sitio web, donde la adquisición de los archivos es a través de un repositorio git.

A través de una terminal del sistema se ingresa al modo administrador (root) donde instalan los siguientes paquetes de datos:

```
sudo -s #Si no está permitido este comando, entonces emplear: su -
apt-get update
apt-get install git-core build-essential cmake libfl-dev bison zlib1g-dev qttools5-dev qtbase5-dev libqt5x11extras5-dev gnuplot
  ↪ libreadline-dev libncurses-dev libxt-dev libopenmpi-dev openmpi-bin libboost-system-dev libboost-thread-dev libgmp-dev
  ↪ libmpfr-dev python python-dev libcgal-dev curl
apt-get install libglu1-mesa-dev
exit
```

1. Descarga OpenFOAM

   ```
   cd ~
   mkdir OpenFOAM
   cd OpenFOAM
   git clone https://github.com/OpenFOAM/OpenFOAM-7.git
   git clone https://github.com/OpenFOAM/ThirdParty-7.git
   ```

   De forma opcional se realizan los siguientes enlaces para asegurar el correcto funcionamiento de MPI empleado por OpenFOAM. También evita problemas si existe presencia de otro paquete MPI instalado en el sistema

   ```
   ln -s /usr/bin/mpicc.openmpi OpenFOAM-7/bin/mpicc
   ln -s /usr/bin/mpirun.openmpi OpenFOAM-7/bin/mpirun
   ```

2. Luego se realiza la construcción de OpenFOAM. Se diferencia en las arquitecturas i686 o x86_64. Para conocer esta información emplear:

   ```
   uname -m
   ```

   Para i686:

   ```
   source \$HOME/OpenFOAM/OpenFOAM-7/etc/bashrc WM_ARCH_OPTION=32 FOAMY_HEX_MESH=yes
   ```

   Para x86_64 se distingue entre el máximo de información que almacena el integer. El caso de un sistema normal de 32-bit, se soporta un máximo de 2.147Ö109 celdas, caras o puntos, en ese caso ingresar el siguiente comando.

   ```
   source \$HOME/OpenFOAM/OpenFOAM-7/etc/bashrc FOAMY_HEX_MESH=yes
   ```

   Para 64-bit, el máximo soportado corresponde a 9.22Ö1018 celdas, caras o puntos. Ingresar lo siguiente:

   ```
   source \$HOME/OpenFOAM/OpenFOAM-7/etc/bashrc WM_LABEL_SIZE=64 FOAMY_HEX_MESH=yes
   ```

3. Para ejecutar el programa se debe proporcionar un **Alias**. En este caso se proporciona *of7* como alias.

   ```
   echo "alias of7='source \$HOME/OpenFOAM/OpenFOAM-7/etc/bashrc $FOAM_SETTINGS'" >> $HOME/.bashrc
   ```

4. Para la instalación de ParaView 5.6.0 se requieren una serie de pasos para asegurar el entendimiento con python y MPI. El primero es comprobar la versión del programa *Qt* nativo de linux y setear la dirección *ThirdParty*, para esto ingresar lo siguiete:

```
cd \$WM_THIRD_PARTY_DIR
export QT_SELECT=qt5
```

   Luego, nuevamente dependiendo de la arquitectura del SO se construye ParaView. Este proceso puede durar un par de horas. Para i686:

```
./makeParaView -python -mpi -python-lib /usr/lib/i386-linux-gnu/libpython2.7.so.1.0 > log.makePV 2>&1
```

   Para x86_64:

```
./makeParaView -python -mpi -python-lib /usr/lib/x86_64-linux-gnu/libpython2.7.so.1.0 > log.makePV 2>&1
```

   Finalmente se actualiza el entorno:

```
wmRefresh
```

5. Finalmente se construye OpenFOAM con los archivos descargados y ParaView. Primero se comprueba que en la carpeta origen *Qt* esté definido como antes, luego se ejecutan los comandos de instalación. Este proceso puede durar bastante (hasta 6 horas según el instructivo) Según la cantidad de nucleos que se deseen emplear se debe modificar el número 4 en la siguiente linea.

```
./Allwmake -j 4 > log.make 2>&1
```

   Se recomienda correr de nuevo el comando anterior, de estar todo correcto entregará un resumen de la instalación. Se existir algún problema será sencillo volver a obtener el mensaje ahora en de forma directa.

C.3.2.1. Revisión instalación

Para revisar el proceso de instalación se puede hacer uso de la búsqueda de un solver empleado por OpenFOAM. El siguiente comando debería proporcionar como respuesta las direcciones de información del solver.

```
icoFoam -help
```

C.3.2.2. Posibles problemas

Si el comando anterior no arroja la información descrita debe revisarse el archivo de registro de la instalación. Este por defecto fue designado como "log.make". Este archivo puede ser abierto empleando cualquier gestor de archivos del sistema operativo, por ejemplo gedit.

```
gedit log.make
```

C.4. Mallado

El proceso de mallado es sin duda un paso clave y determinante al momento de emplear OpenFOAM. Si bien existe un sinnúmero de opciones tanto de código abierto como pagadas para realizar un proceso de mallado a partir de un archivo CAD, para la realización de este instructivo se opta por emplear el mallador de ANSYS (ANSYS Meshing 18.2).

La realización de esta guía en pos del desarrollo de simulaciones de ventilación industrial tiene como resultado algunas prácticas y parámetros de calidad de malla que se alejan de los valores usuales manejados en el área. En las siguientes secciones se enuncian algunos malladores de código libre de uso común en OpenFOAM para luego mostrar un procedimiento general para la implementación de mallas en ANSYS Meshing, su exportación, importación y manejo en OpenFOAM.

C.4.1. Malladores de código libre utilizados en OpenFOAM

Como se hace mención en el capítulo C.2, la malla en OpenFOAM será leída desde múltiples archivos tipo FOAM, los archivos principales del mallado serán: points, faces, owner, neighbour y boundary. Se explica a continuación su composición.

- *points*: Es una lista de vectores que describen la posición de todos los vértices del mallado. El primer vector corresponde al vértice cero, luego el vértice 1 y así.
- *faces*: Es una lista de todas las caras, donde cada cara es una lista los índices de vértices que la conforman. La lista guarda el mismo orden que el archivo anterior, iniciando de la cara cero.
- *owner*: Lista de caras de celdas, donde una cara .ºwner.es aquella referenciada como la cara de donde nacen las caras vecinas.
- *neighbour*: Lista de caras vecinas con su cara .ºwner"de la forma: Owner(vecino1 vecino2 ... vecino n)
- *boundary*: Lista de todos los nombres de los "parches.º nombres de objetos asignados en el proceso de mallado. Se presenta un diccionario por cada nombre con la cantidad de caras que contiene el parche así como el índice de la celda que lo inicia.

BlockMesh

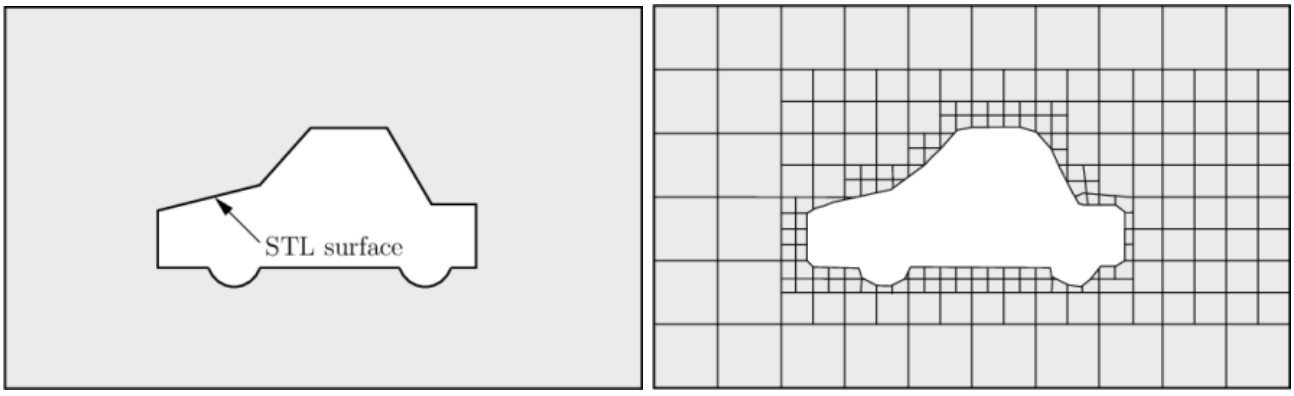
BlockMesh es quizás la forma más sencilla de realizar un mallado. Se requiere básicamente de un archivo que contenga una serie de vértices y comandos que guían la unión de dichos puntos para la generación de elementos tridimensionales. Luego, es posible refinar el mallado y asignar ponderaciones a los objetos de forma de amplificar o disminuir el tamaño de la malla.

Para la generación del mallado debe importarse el diccionario *blockMeshDict* presente en OpenFOAM a la carpeta del caso de estudio y en este archivo describir la malla deseada. Luego, desde la terminal se ingresa el comando *blockMesh* y OpenFOAM lee el diccionario y genera el conjunto de archivos que requiere dentro de la carpeta *polyMesh*.

Para geometrías sencillas BlockMesh presenta gran utilidad, sin embargo al tener geometrías complejas emplear esta herramienta se hace contraproducente, no obstante, el usuario encontrará una buena guía en el siguiente link Introduction to Meshing in OpenFOAM - Cardiff, Philip.

SnappyHexMesh

SnappyHexMesh es un mallador también incluido en la instalación de OpenFOAM pero su funcionamiento es completamente distinto a BlockMesh. Aquí, se debe proporcionar un archivo STL (Stereolithography) o del formato OBJ (Wavefront Object) desde el cual se realiza un proceso de refinación del contorno otorgado por el usuario.



(a) Inicio (b) Término

Figura C.4: Ejemplo de proceso de mallado mediante SnappyHexMesh

El proceso de refinamiento se realiza de forma automatizada como especificada por el usuario, siendo una gran herramienta la opción de añadir capas de refinamiento a los contornos, de manera que desde el contorno se refine el mallado hacia los extremos con un mayor detalle. Las subdivisiones del mallado pueden realazarse en un inicio con BlockMesh, sin embargo SnappyHexMesh emplea el uso de diccionarios más complejos para realizar esta tarea. La forma de operación consiste al igual que con BlockMesh, en un diccionario principal, cuyos sub diccionarios determinarán acciones sobre el mallado a realizar. En este caso, el usuario deberá importar el archivo *snappyHexMeshDict* desde el cual se realizarán todas las etapas del procedimiento.

C.4.2. Malla en ANSYS Meshing

En la siguiente sección se da por descontado que el usuario conoce como realizar e importar una geometría CAD en ANSYS Meshing a través de SpaceClaim o DesignModeler (ambos programas incluídos en ANSYS Workbench 18.2), así como que se está en conocimiento de los parámetros de calidad de malla provistos por ANSYS Meshing. Se recomienda al usuario utilizar SpaceClaim como gestor de la geometría ya que en la práctica se encuentran ciertas deficiencias en geometrías provenientes de DesignModeler.

Si la geometría es compleja o bien presenta grandes diferencias en proporción de los detalles a mallar, el usuario debe considerar la opción de subdividir el volumen en cuerpos más pequeños. El objetivo de este procedimiento será maximizar el control sobre el refinamiento de la geometría sin aumentar desproporcionadamente el refinamiento en todo el volumen, manteniendo así el número de elementos de malla al mínimo sin perjudicar la calidad de esta.

Es imprescindible que de haber subdividido la geometría en tantas partes se haya considerado pertinente, el usuario deba compartir topología entre todos los nuevos cuerpos del volumen como se muestra en la figura C.5. El no realizar este procedimiento puede desencadenar en problemas serios del mallado. Incluso si se cuenta con buenos parámetros de malla es posible que luego de la importación a OpenFOAM el usuario detecte superficies de contacto entre volúmenes que no reconozca.

Como se expone en el capítulo de estructura, cada parámetro a ser resuelto por el programa debe contener condiciones iniciales y de borde en cada una de las superficies o regiones especificadas en el proceso de mallado. De no nombrarse todas las superficies en esta etapa, OpenFOAM asignará un nombre arbitrario a las superficies que detecte, por esto **es de vital importancia que el usuario asigne nombres a todos los volúmenes y superficies de interés o que alojen alguna condición en el futuro.**

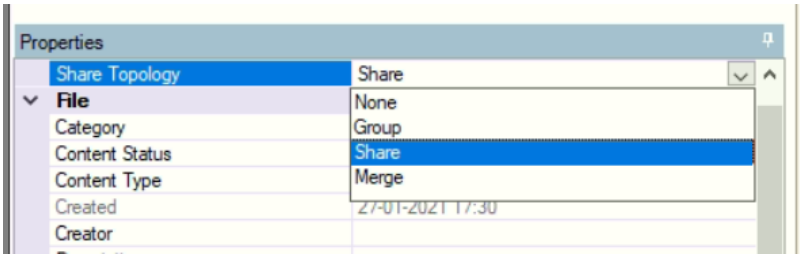


Figura C.5: Share Topology SpaceClaim

Como parámetro de referencia de calidad se mencionará la deformación de las celdas como parámetro principal bajo el cual construir el mallado (skewness). Tamaños de elementos mínimos y

máximos así como tasa de crecimiento y algoritmo para mallar las diferentes superficies y volúmenes se consideran material específico de cada geometría.

Parámetros globales de calidad de malla propuestos por ANSYS y otras guías estipulan valores máximos y mínimos mostrados en la tabla C.5. La práctica ha demostrado que mallas con dichos valores obtienen un buen desempeño en ANSYS Fluent, sin embargo OpenFOAM es notoriamente más sensible a mallas no estructuradas y por ende requiere de parámetros más estrictos. Se sugieren parámetros como aquellos obtenidos en las mallas empleadas para realizar esta metodología. En particular se muestran los parámetros del caso de estudio principal (CE), los cuales demuestran un buen desempeño. Si desea simplificar este análisis bajo la mirada de la deformación máxima de celdas, el skewness máximo deberá permanecer bajo 0.7.

Tabla C.5: Parámetros de calidad de malla deseables

Parámetro	máximo/mínimo	Max/min CE	Promedio CE
Skewness	$< 0{,}9$	0,6	0,26
Razón de aspecto	~1	4,4	1,89
Calidad Ortogonal	$> 0{,}15$	0,4	0,74

Si bien una buena calidad de malla ayudará enormemente a la estabilidad del problema, se debe comprender que esto no solo depende de este único elemento. Las condiciones de borde, esquemas de discretización y tolerancias de los métodos numéricos deben ser vistas como un todo.

C.4.2.1. Conversión de topología de malla mediante ANSYS Fluent

Si se desea mejorar aún más la calidad de malla y de desempeño de esta en OpenFOAM puede realizarse un procedimiento de conversión de la topología del mallado empleando ANSYS Fluent. Se busca transformar un mallado principalmente tetraédrico en uno con elementos poliédricos ya que con toda certeza el cambio de topología mejora la calidad del mallado. El procedimiento a continuación realiza esta función empleando ANSYS Meshing y Fluent.

1. Obtener la geometría mallada en ANSYS Meshing son elementos tetraédricos.
2. Importar la geometría en Fluent modo mallado y emplear la opción "Make Polyhedra".
3. Guardar el caso de fluent con la nueva malla.
4. Abrir un nuevo Fluent con opción de mallado e importar el caso recién guardado.
5. Exportar la malla con extensión .msh y en formato ASCII (NO BINARIO).

Es posible mejorar además la nueva malla. Para esto se debe ingresar a la consola del programa Fluent en el paso 4. Aquí al presionar 'Enter se desplegarán una serie de opciones. Ir a

/mesh/improve-repair/improve-quality. Este comando realizará una mejora de la calidad ortogonal del 0.1 % de los elementos de la malla. Realizar este proceso repetidas veces mejorará sustancialmente los peores elementos del mallado.

C.4.3. Importación a OpenFOAM

Tanto si se opta por realizar la conversión de topología como si no, la exportación de la malla debe realizarse en formato ASCII. Para esto, desde ANSYS Meshing se puede ir a *Tools* > *Options* > *Meshing* > *Export*. Aquí debe modificarse la entrada *Format of input File (*.msh)* y asignar el valor ASCII.

OpenFOAM trae consigo un comando de transformación que permite al usuario implementar cualquier malla de extensión *.msh* en el conjunto de archivos de los cuales se hecho mención anteriormente. Para realizar este procedimiento debe localizarse la malla en la carpeta principal del caso, luego abrir una terminal y proceder con el comando de la linea 1. La comprobación de la malla puede realizarse mediante el comando *checkMesh*. Esta instrucción generará un reporte de la malla, finalizando con una conclusión generada por el programa. Si todo ha ido bien, se desplegará el mensaje "Mesh OK.".

```
1  fluentMeshToFoam nombre_del_archivo.msh
2  checkMesh
```

Luego de estos pasos la malla ha sido correctamente importada, sin embargo aún se requiere modificar las condiciones de borde para poder desplegarla mediante ParaView. La razón de esto es que deben coincidir las entradas de los archivos del directorio 0 con aquellos nombres del mallado. Se recomienda obtener el detalle de los elementos del mallado mediante la dirección */constant/polymesh/boundary*. En este archivo el usuario encontrará todas las entradas necesarias a definir mediante condiciones de borde.

C.5. Condiciones de borde

En OpenFOAM el usuario debe especificar una gran cantidad de parámetros en cuanto a condiciones de borde se refiere. En el código C.1 se muestra un típico archivo de velocidad. Aquí se nota como cada entrada del diccionario *boundaryField* debe llevar un parámetro. Esta última aseveración se extiende a todas las entradas de todos los archivos presentes en el caso de estudio, si bien es posible que alguno de estos parámetros no se emplee en el transcurso de la simulación, OpenFOAM y ParaView requiere que al menos exista un valor para cada condición de borde. En esta sección se muestran las condiciones de borde (BC's) generales en la implementación de cualquier parámetro y luego las más utilizadas para simulaciones empleando SIMPLE/PIMPLE FOAM y buoyantSIMPLE/PIMPLE.

Tabla C.6: Condiciones de borde generales

Condición	Descripción
calculated	Esta condición de borde no evalúa directamente. Fuerza al solver a obtener desde el campo de valores el valor necesario
fixedValue	Proporciona un valor constante y es la base para otras BC's
fixedGradient	Proporciona la condición de gradiente
zeroGradient	Proporciona la condicion de gradiente cero desde una superficie externa a los valores de caras internas
mixed	Otorga la capacidad de mezclar BC's. Ej: fixedValue y zeroGradient
directionMixed	Mismo que anterior pero otorgando la capacidad de aplicar una BC respecto a la dirección del flujo
empty	Condición de borde empleada en problemas 2D. Establece que no existe malla en el eje z
symmetry	Proporciona simetría a la superficie
symmetryPlane	Proporciona la condición de plano de simetría

C.5.1. SIMPLE/PIMPLE

Se debe tener presente en todo momento que hay parámetros intrínsecamente relacionados entre sí y este aspecto debe respetarse en la elección de las condiciones de borde. Para las condiciones de borde mostradas en este capítulo debe tenerse en principal consideración la relación existente entre Velocidad (U) y Presión (p o p_rgh), así como la energía cinética turbulenta (K) y la tasa de disipación turbulenta (ϵ).

A continuación se describen las principales condiciones de borde a ser empleadas para los parámetros U, p, K, ϵ, nut y otras condiciones de uso común y transversales a los parámetros.

Tabla C.7: Condiciones de borde para Velocidad U

Condición U	Descripción
flowRateInletVelocity	Corrige valores extrapolados o crea un campo de valores uniformes normal a la dirección de la superficie y ajustado al valor del fluido circundante
freestreamVelocity	Condición de flujo libre para velocidad
pressureInlet-OutletVelocity	Condición de velocidad cuando es especificada la presión de entrada Salida: zeroGradient, entrada: Valor estimado de celdas vecinas
pressureInlet-UniformVelocity	Similar a la condición anterior pero especifica un valor de velocidad uniforme para el flujo de entrada
surfaceNormalFixedValue	Proporciona un valor constante y normal a la superficie

Tabla C.8: Condiciones de borde para Presión

Condición p	Descripción
fanPressure	Condición de presión absoluta
fixedFluxPressure	Proporciona un gradiente de presión acorde a un valor de flujo de entrada especificado en la BC de velocidad
freestreamPressure	Condición de flujo libre para presión
totalPressure	Provee condición de presión total sobre la superficie

Tabla C.9: Condiciones de borde adicionales

Otros	Descripción
fixedMean	Extrapola el campo de la variable a la superficie usando los valores de las celdas vecinas. Ajusta la distribución para generar valores promedio
freestream	Condición de flujo libre
outletInlet	Proporciona una BC mixta con valor genérico sobre la entrada y valor específico ante la salida
inletOutlet	Proporciona una BC mixta con valor específico para la entrada y genérico para la salida
turbulentIntensity-KineticEnergyInlet	Proporciona condición para K a partir de intensidad turbulenta entregada por el usuario como acción de la velocidad media de la BC.
noSlip	BC para paredes. Impone velocidad cero.
slip	Provee condición de deslizamiento para la superficie

Tabla C.10: Funciones de pared para condiciones de borde típicas

Variable	BC	Descripción
nut	nutkWallFunction	Función por defecto para paredes
	nutRoughWall-Function	Función para paredes rugosas
k	kqRWallFunction	Función genérica a aplicar para k, q o Reynold Stress
epsilon	epsilonWallFunction	Función generica para condición de borde en paredes
alphat	alphatWallFunction	Función que proporciona diffusividad termal según nut y Pr

Para el caso de buoyantSIMPLE o buoyantPIMPLE FOAM la implementación de los parámetros varía. La principal adición al haber un fenómeno de transferencia de calor es la introducción del parámetro de difusividad termal *alphat*, cuya condición se aprecia en la tabla **??**. Junto con esto, las variables y sus condiciones de borde se mantienen similares a la implementación SIMPLE/PIMPLE a excepción de la presión.

Para caracterizar el fenómeno de compresión se adiciona el parámetro de presión hidrostática p_rgh. Esta presión será el simil en cuanto a condiciones de borde a la variable de presión p en SIMPLE/PIMPLE, mientras que el parámetro de presión p se mantiene como un campo a calcular. Dada la importancia de esta implementación, en los códigos C.11, C.9 y C.10 se ilustra la diferencia.

Código C.8: SIMPLE/PIMPLE: p

```
1   FoamFile {
2   version     2.0;
3   format      ascii;
4   class       volScalarField;
5   object      p;}
6 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
7 dimensions      [0 2 -2 0 0 0 0];
8 internalField   uniform 0;
9 boundaryField{
10    wall-vol{
11       type           zeroGradient;}
12    out1{
13       type           fixedValue;
14       value          uniform 0;}
15    out2{
16       type           fixedValue;
17       value          uniform 0;}
18    inlet{
19       type           zeroGradient;}
20    bloque2{
21       type           zeroGradient;}
22    bloque1{
23       type           zeroGradient;}
24 }
```

Código C.9: buoyantSIMPLE/PIMPLE: p

```
    FoamFile{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [1 -1 -2 0 0 0 0];
internalField   uniform 1e5;
boundaryField{
    wall-vol{
        type            calculated;
        value           $internalField;}
    out1{
        type            calculated;
        value           $internalField;}
    out2{
        type            calculated;
        value           $internalField;}
    inlet{
        type            calculated;
        value           $internalField;}
    bloque2{
        type            calculated;
        value           $internalField;}
    bloque1{
        type            calculated;
        value           $internalField;}
}
```

Código C.10: buoyantSIMPLE/PIMPLE: p_rgh

```
    FoamFile{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p_rgh;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [1 -1 -2 0 0 0 0];
internalField   uniform 1e5;
boundaryField{
    wall-vol{
        type            fixedFluxPressure;
        value           uniform 1e5;}
    out1{
        type            fixedValue;
        value           uniform 0;}
    out2{
        type            fixedValue;
        value           uniform 0;}
    inlet{
        type            zeroGradient;}
    bloque2{
        type            fixedFluxPressure;
        value           uniform 1e5;}
    bloque1{
        type            fixedFluxPressure;
        value           uniform 1e5;}
}
```

C.6. Aproximación computacional

OpenFOAM ofrece una muy amplia oferta de solver, sin mencionar claro que cada uno de ellos puede servir de punto de partida para la creación de un solver propio. En este capítulo se enuncian algunos de los solver disponibles en OpenFOAM y se analizan la implementación de los solver SIMPLE/PIMPLE y sus principales diferencias con buoyantSIMPLE/PIMPLE, en tanto las ecuaciones gobernantes a resolver como su implementación,

C.6.1. Solver de OpenFOAM

Dado el amplio espectro de solver presentes en OpenFOAM es que cada uno ha sido nombrado de forma sencilla y explícita en cuanto a lo que resuelve cada algoritmo. Para hacer una búsqueda acerca de cual es el indicado se recomienda acudir al manual, a los solver descritos a continuación o hacer uso de las opciones de búsqueda del programa.

Para esto último, el usuario puede ingresar el comando a continuación para identificar el directorio que contiene esta información o bien digitar el código *sol* en una terminal con acceso a las funciones de OpenFOAM.

Código C.11: SIMPLE/PIMPLE: p

```
1 $FOAM_SOLVERS
```

En OpenFOAM cada solver posee una función en particular, dentro de las categorías principales se encuentran:

FOTO SOLVERS SOL basic discreteMethods financial lagrangian combustion DNS heatTransfer multiphase compressible electromagnetics incompressible stressAnalysis

Por los fines de este manual, se muestran a continuación una breve descripción de los solver relevantes pertenecientes a *basic*, *compressible*, *heatTransfer* y *incompressible*.

basic

- laplacianFoam: Resuelve la ecuación de Laplace.
- potentialFoam: Resuelve el potencial de velocidad sobre el dominio para determinar el flujo desde donde se obtiene en campo de velocidades. Se suele emplear a modo de inicialización del dominio para problemas de turbulencia.
- scalarTransportFoam: Resuelve la ecuación de transporte en estado transiente o estacionario para un escalar pasivo.

compressible

- rhoCentralFoam: Solver basado en la densidad para flujos compresibles. Emplea esquema central-upwind. Soporta mallas móviles y cambios de topologías.
- rhoSimpleFoam: Solver en estado estacionario para fluidos turbulentos.
- rhoPorousSimpleFoam: Solver en estado estacionario para fluidos turbulentos con manejo de porosidad explicita o implícita.
- rhoPimpleFoam: Solver transiente para fluidos compresibles. Soporta opcionalmente mallas móviles y cambios de topologías.

incompressible

- icoFoam: Solver transiente para fluidos laminares Newtonianos (posee simil para fluidos no Newtonianos)
- simpleFoam: Solver de estado estacionario y turbulencia que resuelve el algoritmo SIMPLE.
- porousSimpleFoam: Solver de estado estacionario y turbulencia con manejo de porosidad explícita o implícita. Soporta multiples cuadros de referencia MRF (Multi Reference Frames).
- pisoFoam: Solver transiente para flujo turbulento que resuelve el algoritmo PISO.
- pimpleFoam: Solver transiente para flujo turbulento de fluidos Newtonianos. Resuelve mallas móviles y admite cambios de topología.

heatTransfer

El intercambio de calor se presenta al resolver la ecuación de energía C.2, esta da cuenta del intercambio de calor en el medio de trabajo. Los términos de la derecha de esta expresión representan: la transferencia de energía mediante conducción (donde k_{eff} será la conductividad efectiva producto de la adición entre la conductividad del medio y el término de conductividad turbulento definido por el modelo de turbulencia empleado); el término difusivo de transporte de especies con h_j la entalpía y J_j el flujo de la especie j y la disipación viscosa. Además, se define S_h como calor externo producto de una fuente volumétrica o reacción química.

$$\frac{\partial}{\partial t}(\rho E) + \nabla \cdot (\vec{v}(\rho E + p)) = \nabla \cdot \left(k_{eff} \nabla T - \sum_j h_j \vec{J}_j + (\bar{\tau}_{eff} \cdot \vec{v}) \right) + S_h \tag{C.2}$$

A la izquierda de la igualdad se plantea la variación interna de energía en el volumen de control. Aquí, E está dada por la ecuación 2.5, donde h corresponde a la entalpía sensible cuyo cálculo variará dependiendo de las condiciones del fluido de trabajo.

Simplificación de Boussinesq: Esta aproximación sostiene que la variación de la temperatura de un fluido se puede ignorar excepto por la densidad y que la dependencia del fluido con la densidad solo se considera cuando da lugar a una convección del flujo del tipo convección natural (Zhang et al., 2016). Para caracterizar esta aproximación se emplea la siguiente relación, en donde ρ_0 es la densidad de referencia del fluido a una temperatura T_0. β es el coeficiente de expansión volumétrica.

$$\rho = \rho_0 \left[1 - \beta \left(T - T_0\right)\right] \tag{C.3}$$

En OpenFOAM existen tres niveles de aplicación de la resolución de la ecuación de energía en la transferencia de calor. A continuación se describen las tres formas que prsenta OpenFOAM de caracterizar este fenómeno.

- Buoyant Boussinesq SIMPLE/PIMPLE Foam: Estos métodos emplean la aproximación de Boussinesq descrita en las ecuación C.3. Serán idóneos para fluidos turbulentos en estado estacionario (SIMPLE) y transiente (PIMPLE) y bajo la suposición de incompresibilidad. Se sugiere el uso de este método ante inestabilidades en el desarrollo de la simulación empleando métodos más complejos.
- Buoyant SIMPLE/PIMPLE Foam: Es por defecto el método a utilizar para ventilación ya que tanto para fluidos transientes como estacionarios se asume compresibilidad en el fluido, incluyendo los efectos de convección generados por las diferencias de temperatura y densidad. Opera netamente para un mismo fluido, cuyas propiedades termo-físicas deben ser entregadas al igual que en los otros métodos.
- chtMultiRegion Foam: Es el mecanismo de resolución más capaz para la modelación de transferencia de calor en OpenFOAM. Permite la interacción de diversos medios y fases entre sólidos y fluidos. Opera bajo la lógica de subdividir el cálculo de la dinámica de cada fluido, teniendo que el usuario especificar parámetros de simulación sobre cada elemento distinto que componga el caso.

Variaciones en thermophysicalProperties

Como se expone en esta sección, existen diferentes niveles de profundidad en cuanto a transferencia de calor se refiere. Este instructivo abarca la implementación de los dos primeros niveles anteriormente mencionados, por lo que a continuación se detallan los cambios a realizar en el archivo */constant/thermophysicalProperties* de forma de realizar la transición de un modelo a otro.

La primeria diferencia a notar es la entrada de nombre '*thermo*', esta variable determina si se asume constante C_p (en caso de *hConst*) o C_v. Junto con esto la ecuación de estado cambia a Boussinesq, habilitando la ecuación C.3, por lo que en el diccionario de parámetros *mixture* ahora es necesario el ingreso de los valores antes descritos.

Código C.12: thermophysicalProperties buoyantSIMPLE/PIMPLE

```
   FoamFile{(...)}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
thermoType{
   type           heRhoThermo;
   mixture         pureMixture;
   transport       const;
   thermo          hConst;
   equationOfState perfectGas;
   specie          specie;
   energy          sensibleEnthalpy;
}
pRef           100000;

mixture //propiedades para el aire
{
   specie{
      molWeight       28.96;}
   thermodynamics{
      Cp              1006.43;
      Hf              0;}
   transport{
      mu              1.831e-05;
      Pr              0.705;}
}
```

Código C.13: thermophysicalProperties buoyantBoussinesqSIMPLE/PIMPLE

```
   FoamFile{(...)}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
thermoType{
   type           heRhoThermo;
   mixture         pureMixture;
   transport       const;
   thermo          eConst;
   equationOfState Boussinesq;
   specie          specie;
   energy          sensibleInternalEnergy;
}
mixture
{
   specie{
      molWeight       28.9;}
   equationOfState{
      rho0            1;
      T0              300;
      beta            3e-03;}
   thermodynamics{
      Cv              712;
      Hf              0;}
   transport{
      mu              1e-05;
      Pr              0.7;}
}
```

C.7. Ejecución, control y monitoreo

Como todo lo controlado por OpenFOAM, la orden de ejecución y detención de una simulación será a través de la terminal de comandos. En esta sección se presentan los comandos en orden cronológico para llevar a cabo el proceso de simulación de un caso. Se verá que existen formas más limpias de llevar el proceso y otras que mostrarán todo lo que sucede en pantalla. En esta sección también se mostrará como desarrollar una simulación en paralelo en una misma máquina[3].

Antes de iniciar el proceso de simulación, el usuario debe recordar que el archivo de control *system/controlDict* es el encargado de almacenar la información de la simulación. Como usuario, se deben ejecutar los comandos que inicien la simulación del método alojado como *application* en *controlDict*. Sin ir más lejos, basta para que el usuario digite *simpleFoam* (suponiendo que aquello es lo que contiene el archivo de control) para que la simulación se ejecute.

A menos que la impresión de los datos esté desactivada, al ejecutar la simulación en la terminal se mostrará un encabezado con toda la información relevante del caso. Luego de esta impresión comenzarán los ciclos de tiempo. Cada ciclo de tiempo corresponderá a la simulación de un paso de tiempo según el solver empleado. Para lo que sería una simulación de SIMPLE cada paso de tiempo se presentará información semejante a la figura C.6.

```
File  Edit  View  Search  Terminal  Help
Starting time loop

Time = 0.0001

smoothSolver:  Solving for Ux, Initial residual = 0.574761, Final residual = 0.0548666, No Iterations 2
smoothSolver:  Solving for Uy, Initial residual = 0.765132, Final residual = 0.0688806, No Iterations 2
smoothSolver:  Solving for Uz, Initial residual = 0.574456, Final residual = 0.0544578, No Iterations 2
GAMG:  Solving for p, Initial residual = 1, Final residual = 0.0702406, No Iterations 1
time step continuity errors : sum local = 7.55233e-05, global = 2.30268e-07, cumulative = 2.30268e-07
smoothSolver:  Solving for epsilon, Initial residual = 0.415763, Final residual = 0.0169958, No Iterations 2
bounding epsilon, min: -5.98257 max: 92.436 average: 14.7189
smoothSolver:  Solving for k, Initial residual = 1, Final residual = 0.070108, No Iterations 2
ExecutionTime = 3.28 s  ClockTime = 3 s
```

Figura C.6: Ejemplo ejecución en terminal: simpleFoam

Como se aprecia a simple vista el nivel de residuos de la simulación se presentan solo en consola, graficarlos es posible pero luego de haberlos almacenado y ejecutar un script de código adicional de forma paralela a la simulación. Este proceso se muestra en la sección ASDASDASDASDAS-DASDADSA.

Para almacenar los datos presentados en la figura C.6 el comando de ejecución debe ir acompaña-

[3] La opción de realizar simulaciones en más de un equipo es perfectamente viable pero escapa de los alcances de este manual.

do del caracter '>' como se muestra en el código siguiente (linea 1). Aquí la palabra a continuación define el nombre del archivo de registro. Al ejecutar un comando con registro de datos el usuario notará que la ventana de la terminal ya no se ve modificada constante mente, el programa está simulando en segundo plano. Para ver el desarrollo de la simulación, es decir lo que sería el archivo 'log_simple_01', se debe abrir **otra terminal de comandos** con origen en el directorio donde se está ejecutando el registro e ingresar la segunda linea del código a continuación. De esta forma la información será registrada hasta que se detenga la simulación, por error o porque se ha alcanzado el tiempo final.

Código C.14: Comando de ejecución simpleFoam

```
1 simpleFoam > log_simple_01
2 tail -f log_simple_01
```

C.7.1. Ejecución en paralelo

Para realizar una simulación empleando más de un procesador es necesaria la subdivisión del mallado (también conocido como descomposición del dominio). Cada procesador tendrá a cargo una sección de la subdivisión y resolverá el problema de forma independiente. Para luego unificar la información se deberá reconstruir el dominio, este proceso puede ser realizado a través de ParaView o bien manualmente.

Para la descomposición del dominio se emplea el diccionario provisto por OpenFOAM *decomposeParDict*. Al interior de este documento se encuentran los métodos de división (revisados a continuación) y un par de parámetros que darán cuenta de la opción escogida por el usuario. La variable *numberOfSubdomains* dará cuenta en cuantas partes se desea realizar la división, mientras que *method* dará cuenta del método escogido para este fin.

- simpleCoeff: Método sencillo de descomposición donde se ingresa las subdivisiones según las coordenadas cartesianas. La entrada mostrada en el código a continuación representaría 2 divisiones en la dirección x, 1 división en y y 3 en la dirección z.
- hierarchicalCoeffs: Procedimiento idéntico a simple pero admite la elección del orden de la subdivisión.
- scotch: Es quizás el método más sencillo de implementar ante geometrías complejas. No se requiere instrucción geométrica ya que el programa realiza la subdivisión considerando la cantidad deseada por el usuario. Presenta la opción de 'balancear' la cantidad del dominio si se emplean procesadores de distintas velocidades.
- manual: Aquí el usuario debe proporcionar la dirección de cada celda a un procesador en particular empleando un archivo externo.

En el ejemplo a continuación se muestra el contenido de un diccionario de descomposición. El método seleccionado corresponde a *scotch*, subdividiendo el dominio en ocho partes más menos idénticas.

```
FoamFile{ (...) }
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
numberOfSubdomains 8;
method         scotch;

simpleCoeffs    {
    n            (2 1 3);
    delta         0.001;    }

hierarchicalCoeffs    {
    n            (1 1 1);
    delta         0.001;
    order         xyz;    }

manualCoeffs    {
    dataFile       "";    }

distributed     no;
//Indica si los datos están distribuidos en más de un disco
roots          ( );
```

Es importante mencionar que la descomposición del dominio debe realizarse en una etapa previa a la ejecución de la simulación. Para realizar esto el usuario deberá ingresar el siguiente comando en la terminal. Esta acción generará nuevos directorios en la carpeta principal de la simulación. Cada carpeta contendrá también la subdivisión de los directorios temporales ya simulados. De no existir ninguno, se encontrará el directorio de las condiciones de borde 0.

```
decomposePar
```

Luego de haber subdividido el dominio, el comando de ejecución C.14 debe ser modificado también. Ahora, se requiere que el usuario especifique la cantidad de procesadores a emplear en la instrucción así como que el proceso ha de realizarse en paralelo, la instrucción que da cuenta de aquello se muestra en el código C.15.

Código C.15: Comando de ejecución en paralelo simpleFoam

```
mpirun -np 8 simpleFoam -parallel > log_simple_01
```

En la imagen siguiente se muestra la opción de ParaView que permite al usuario no tener que reconstruir manualmente el caso de estudio para la visualización de datos. Si la simulación no se ha realizado en paralelo o el caso ya ha sido importado esta opción no estará disponible.

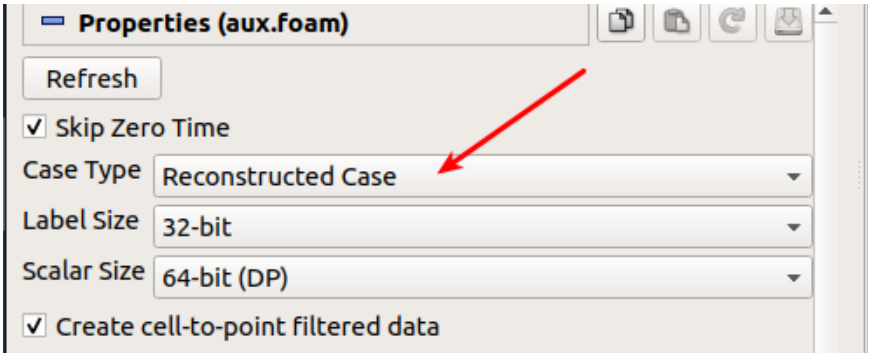


Figura C.7: Menú de propiedades e información ParaView

La reconstrucción del caso se realiza haciendo uso del comando *reconstructPar* junto con un ingreso de los registros temporales que desean unificarse. Si el usuario no ingresa esta información, la instrucción ejecutará la unificación de todos los pasos de tiempo. A continuación se muestran ejemplos de esto.

```
1  reconstructPar
2  reconstructPar  -latestTime       //Toma el último paso de tiempo
3  reconstructPar  -time 10          //Reconstruye solo el paso de tiempo igual a 10
4  reconstructPar  -time '10, 20, 30'  //Reconstruye los pasos de tiempo 10, 20 y 30
```

C.7.2. Archivos ejecutables

Existe la opción de configurar instrucciones para que la simulación opere de forma independiente. Si bien esta cualidad presenta gran potencial, aquí se limita a la operación más sencilla de la automatización, ya que es imprescindible conocer este tipo de archivos para el manejo de tutoriales. La forma de ejecutarlos es sencilla, se deben ingresar los siguientes comandos.

```
1  ./Allclean
2  ./Allrun
```

C.7.2.1. Allclean

Este archivo (mostrado a continuación) por defecto hará uso de la función *cleanCase* de OpenFOAM, eliminando todos los directorios temporales distintos de 0, la carpeta de mallado *constant/polyMesh* y todos los archivos de registro de datos que se pudieran haber generado mediante una simulación.

Gracias a la naturaleza de código abierto de OpenFOAM es posible generar funciones de limpieza propias de forma de cumplir con cualquiera sea la necesidad específica del usuario. Para esto deberá ser modificado el archivo */bin/tools/CleanFunctions*. Otras funciones útiles presentes en este archivo se muestran en las lineas 8 y 9.

Código C.16: Ejecutable Allclean

```
1  #!/bin/sh
```

```
cd ${0\%*} || exit 1    # Run from this directory

# Source tutorial clean functions
. $WM_PROJECT_DIR/bin/tools/CleanFunctions

cleanCase
//cleanTimeDirectories: Remueve las carpetas temporales
//cleanPostProcessing: Remueve la carpeta postProcessing
#------------------------------------------------------------------------------
```

C.7.2.2. Allrun

La función de este archivo será conducir las diversas entradas que el usuario realiza a través de la terminal de comandos. Si bien muchos tutoriales lo presentan, en la práctica puede que el usuario lo modifique a su conveniencia de a cuerdo a lo que desee ejecutar y registrar. Como muestra de esto, se muestra a continuación un archivo Allrun que realiza y registra todos los comandos mostrados en este capítulo.

Código C.17: Ejecutable Allrun

```
#!/bin/sh
cd \${0\*} || exit 1    # Run from this directory
//Copia de respaldo de las condiciones iniciales
cp -r 0 0.backup
//se parte de cero
cleanCase
//se genera la malla
fluent3DMeshToFoam mallado_aux.msh > log.meshConversion 2>&1
//Se guarda la información de calidad de la malla
checkMesh   > log.meshQuality 2>&1
//Se descompone el dominio según decomposeParDict
decomposePar > log.decomposePar 2>&1
//Se ejecuta la applicación simpleFoam en paralelo y se registran los valores
mpirun -np 8 simpleFoam -parallel > log.simple_01 2>&1


# Más funciones se encuentran en:
#. \$WM_PROJECT_DIR/bin/tools/RunFunctions

#------------------------------------------------------------------------------
```

Nota: La extensión *2>&1* es incluida por defecto en archivos de $C++$ al momento de registrar información, da cuenta de la sobreescritura del archivo si este es encontrado por el programa.

C.8. Visualización de resultados y post procesamiento

En esta sección se describe brevemente el funcionamiento de ParaView junto con mostrar al usuario las principales herramientas del programa. Se muestra la visualización sencilla de datos, generación de planos de corte, manejo de unidades y generación de lineas de flujo a partir de un set de datos.

C.8.1. Disposición general ParaView

Para iniciar el programa ParaView una vez instalado es necesario el empleo de la terminal de comandos. Aquí se puede iniciar el programa de dos formas distintas, una asociada al proyecto al cual se desea realizar el análisis y otra de forma independiente. Para el primer caso se hace uso del comando de la linea 1 a continuación. En caso de que se desee ejecutar el programa de forma independiente, es decir, sin generar un archivo automático en la carpeta del caso, se debe emplear la segunda instrucción.

```
1 paraFoam
2 paraview
```

Independiente de la elección de inicio del programa se iniciará una ventana como la mostrada en la figura C.8. Aquí se aprecian tres secciones principales: sección superior, lateral izquierda y central. En la sección central como es de esperarse se visualizará la geometría del caso de estudio. La sección lateral izquierda tendrá que ver con todo el manejo de la geometría, desde la importación de parámetros a la visualización de estos. Se cuenta con un panel superior cuya función es desplegar el árbol del proyecto, aquí se podrá llevar la cuenta, organizar y renombrar todas las funcionalidades aplicadas por el programa.

Finalmente, en la parte superior se tendrá el banco de control. Aquí se podrán seleccionar las funciones del programa, escoger el paso de tiempo deseado y determinar las escalas de valores a mostrar para cierto parámetro escogido. Es importante notar que aparecerán solo aquellos pasos de tiempo y parámetros previamente importados. Para la importación de las carpetas temporales debe seguirse lo mostrado por la figura C.3. Para la selección de parámetros de la simulación debe seleccionarse la importación de cada uno de ellos según lo mostrado por la figura C.9 (a).

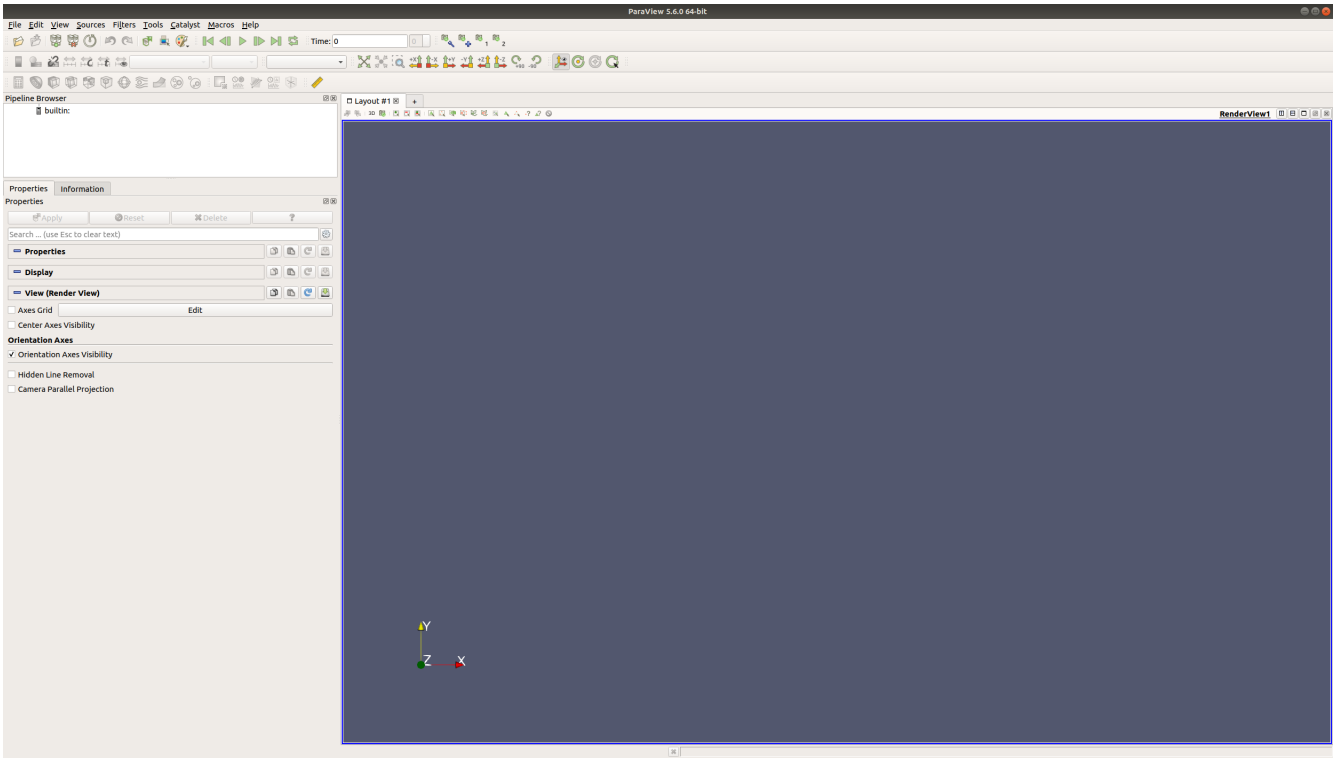
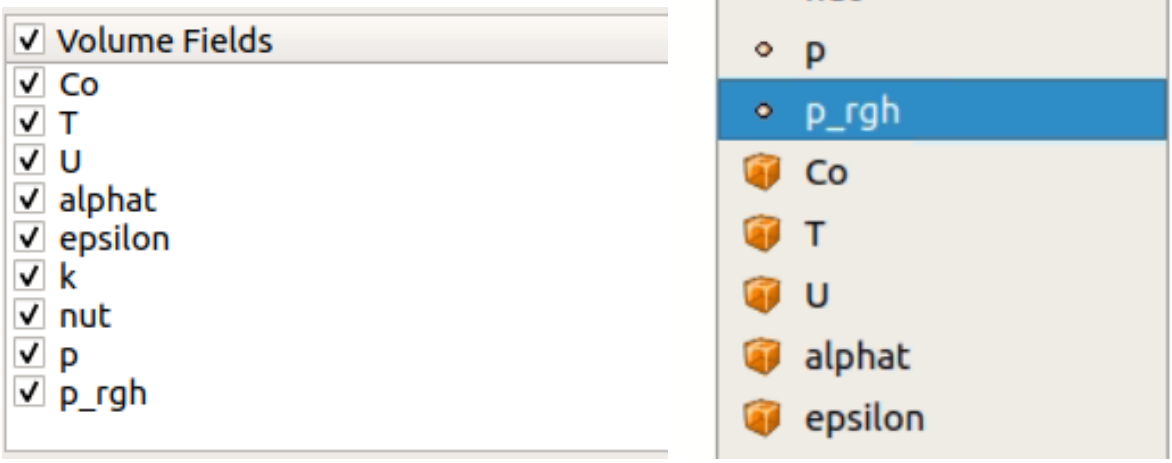


Figura C.8: ParaView: Ventana principal



(a) Selección parámetros

(b) Selección continuidad

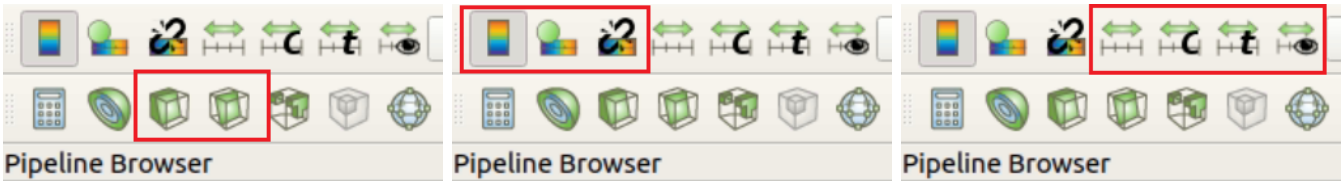
Figura C.9: ParaView: Selección de parámetros y visualización

Una vez realizada la importación de la geometría, los parámetros deseables a visualizar y los pasos de tiempo que contienen esta información, es posible seleccionar un parámetro y graficar los resultados de este de forma inmediata en la ventana de visualización. El proceso puede demorar al-

gunos segundos dependiendo de la cantidad de elementos del mallado y la capacidad computacional del equipo.

La opción de visualización de parámetros desplegará dos opciones para cada uno en conjunto con un icono como se muestra el la figura C.9 (b). El icono de bloque visualizará los datos en la geometría de forma discontinua, mientras que el icono de 'punto' realizará un proceso de suavizado entre los valores del las celdas, mostrando un resultado de contornos suavizados.

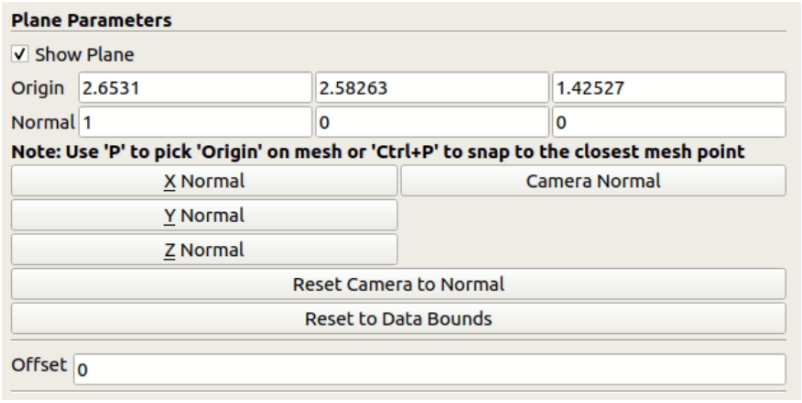
C.8.2. Visualización y generación de cortes



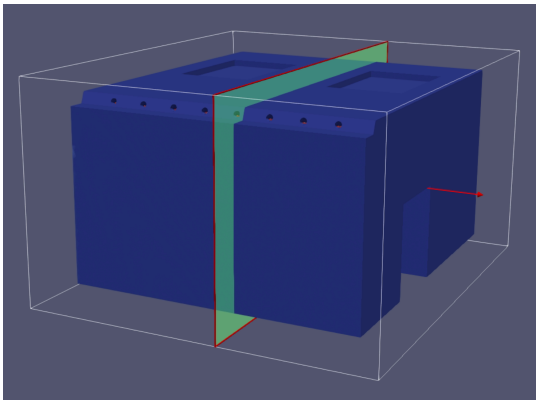
(a) (b) (c)

Figura C.10: ParaView: Selección de parámetros y visualización

Existen dos funciones principales para generar vistas en corte para una geometría: *slice* y *clip*. La función *slice* (C.10 (a - izq)) realizará un corte en la geometría de espesor unitario, es decir, se podrán ver los valores del plano escogido desde ambos lados de este. La función *clip* (C.10 (a - der)) por otro lado, realizará un corte de la geometría principal, dejando al descubierto el plano indicado por el usuario. Ambas funciones requieren que el usuario determine el plano de corte deseado empleando el panel de control (panel lateral izquierdo) o bien utilizando el recuadro rojo desplegado en la geometría. Para esto es necesario utilizar el mouse y determinar la posición deseada. Los valores en el espacio del plano se mostrarán en el panel de información del plano como se aprecia en la figura C.11.



(a) Panel de plano



(b) Selección

Figura C.11: ParaView: Selección de vista en corte

Tras la realización del corte y en general, tras cada nueva disposición de la geometría, es posible modificar la escala de visualización de los datos, de esta forma se mostrarán aquellos rangos de valores presentes o deseados por el usuario. Para esto existen cuatro herramientas: Reescalar para el rango de datos de la selección, rango de datos propio, rango de datos en todo el espectro temporal y reescalar para el rango de datos visibles. Los botones a cada una de estas funciones se encuentran en la figura C.10 (c) ordenados de izquierda a derecha.

- Rango de datos: Fija la escala de datos independiente de si existen datos no visibles.
- Rango de datos propio: Fija el rango de datos con valores otorgados por el usuario.
- Rango de en todo el espectro de tiempo: De forma automática ParaView recorrerá todos los conjuntos de datos disponibles y sobre ese rango de datos establecerá un rango de datos para los mostrados en la ventana.
- Rango de datos visibles: Quizás la opción de más fácil acceso, fija el rango de datos para el parámetro deseado según los datos visibles. Variará la escala si se rota o modifica la geometría, pero antes debe aplicarse nuevamente la herramienta presionando el botón.

En todo momento de la selección de datos para un determinado espacio de tiempo se puede visibilizar el rango de valores de cada parámetro. Para esto es necesario seleccionar la ventana *Information* a un costado de la pestaña *Properties* en el panel de control justo debajo del árbol de trabajo.

Luego haber seleccionado tanto los datos a mostrar como el espectro de colores segun el rango de valores de la selección, es posible generar otro mapa de colores, como modificar sus propiedades,

fuentes, colores, etc. Estas funciones son accedidas mediante los botones presentes en la figura C.10 (b). Aquí el menú principal es mostrado presionando el botón central.

Para modificar los aspectos estéticos del *Color map*, es necesario acceder a un pequeño botón con una letra 'e' dispuesto en el extremo derecho superior del menú de propiedades de la barra de color.

C.8.3. Generación de lineas de flujo

La generación de lineas de flujo es un proceso sencillo pero requiere la realización de algunos pasos en el orden correcto. Si bien ParaView cuenta con una función generadora de lineas de flujo, se encuentra que en la práctica el método es engorroso y carece de potencial para algunos fines. El método presentado a continuación permite al usuario realizar lineas de flujo a partir de cualquier superficie nombrada en la geometría.

1. Primero es necesario hacer la importación de los datos a muestrear. Se requiere importar todos los parámetros necesarios y seleccionar el paso de tiempo a gusto.

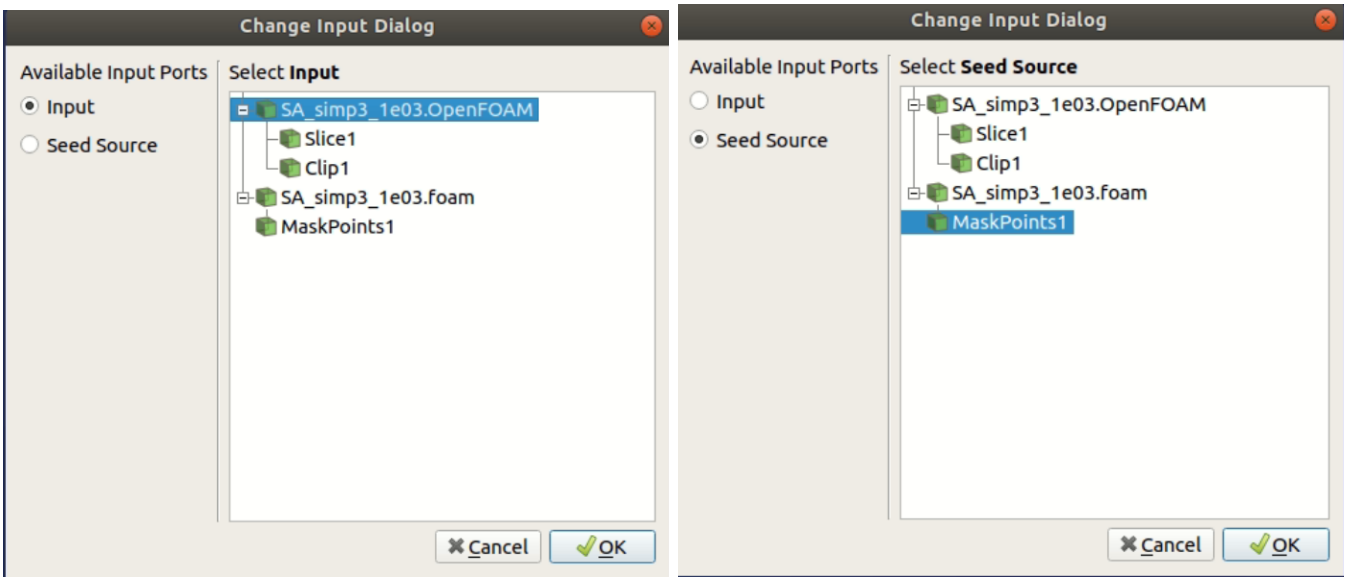
2. Luego, se debe realizar una segunda importación del caso. Abrir → Seleccionar archivo.foam. En esta importación se debe solo importar la superficie de donde se desean generar las lineas de flujo, por ejemplo: inlet[4].

3. Se ocupa el filtro 'Mask Points' para determinar la selección de puntos de emisión de las lineas de flujo. Para esto, teniendo seleccionada la última geometría importada en el árbol de trabajo, se ingresa a la pestaña 'Filters' y en la barra de búsqueda se busca la opción 'Mask Points'.

4. En la nueva tarea del árbol se debe seleccionar un rango de puntos y una cantidad. Luego, se buscará el filtro 'Stream tracer With Custom Source'.

   Una ventana como la mostrada a continuación aparecerá. Deberá seleccionarse como 'input' los datos del caso previamente importados. Luego se debe seleccionar un origen ('Seed Source'), para lo cual se debe seleccionar la función de puntos generada previamente ('Mask Points')

[4] Para observar mejor la superficie puede seleccionarse la opacidad de la figura: click derecho, 'Set Block Opacity'



(a) Panel de plano (b) Selección

Figura C.12: ParaView: Generación de lineas de flujo

El producto final será algo como lo mostrado en la figura C.13.

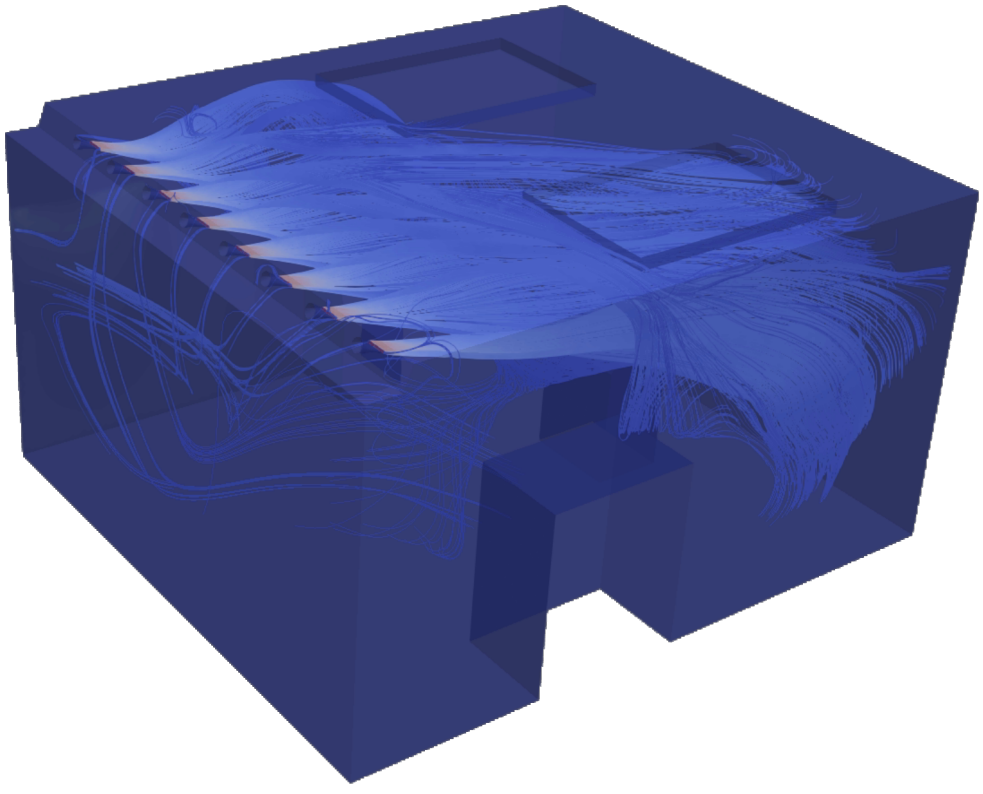


Figura C.13: ParaView: Resultado Steamlines personalizado

C.9. Comandos útiles linux

Tabla C.11: Comandos sencillos Ubuntu/Unix

Comando	Definición	Descripción/uso
cd	Cambiar directorio	cd ../ –> sale un nivel ../.. sale dos. terminando con * ante una duda, lista el resto.
ls	Enlistar archivos	ls –lt muestra las propiedades de los archivos (ll –t lo mismo para otras distribuciones).
mkdir	Crear directorio	Crea carpetas.
cp	Copiar archivos	Cp 'nombre.tipo' carpeta/'nuevonombre.tipo' copia el archivo viejo y lo renombra cp –r para copiar directorios.
mv	Mover archivos	Para copiar directorios y renombrarlos se puede hacer mv directorio directorio2 y queda todo en el mismo lugar pero con otro nombre.
rm	Remover archivos	
touch	Crear archivo comodín	Permite crear todo tipo de archivos y al aplicarlo sobre un archivo preexistente queda modificado como recién hecho.
pwd	-	Imprime directorio actual.
clear	Limpiar ventana	Limpia la ventana. Comandos antiguos quedan hacia .ªrriba"
exit	Salir/cerrar	-
sudo -s	root/adm	Otorga permisos administrador
-	Script	Para correr script "./Allrun"