Trabajo Fin de Máster en Ingeniería Aeronáutica

MODELADO CFD DEL FLUJO NO REACTIVO EN LA CÁMARA DE COMBUSTIÓN DE MOTORES AERODERIVADOS MEDIANTE MALLADO ADAPTATIVO

Autor

Carlos Moreno Montagud

Tutor

Marcos Carreres Talens

Cotutor

Mario Belmar Gil

Universidad Politécnica de Valencia

Escuela Técnica Superior de Ingeniería del Diseño

Departamento de Máquinas y Motores Térmicos

Valencia - Septiembre de 2018

Resumen

En la sociedad actual existe cada vez mayor preocupación por las emisiones contaminantes, especialmente el NO_x, y eso se traduce en la creciente concienciación de la población sobre dicho problema así como en leyes cada vez más estrictas. En este contexto surgen estrategias de inyección y combustión diferentes que tratan de corregir este problema, como la Lean Direct Injection (LDI) para motores de flujo contínuo.

En el presente Trabajo Fin de Máster se pretende avanzar en el estudio de dicha estrategia mediante resolución numérica con códigos CFD. Los pasos a seguir son: analizar la inyección de combustible, su atomización y posterior evaporación, su interacción con el flujo turbulento en la cámara de combustión, la eficiencia global y las emisiones vinculadas a este proceso. A partir de unos resultados experimentales, obtenidos de la cámara de combustión LDI diseñada en CORIA para el proyecto KIAI, se tratará de caracterizar el campo de velocidades turbulento que se produce en el interior. Para ello se simularán y compararán casos no reactivos, premezclado (inyectando una mezcla pobre de aire y combustible) y posteriormente no premezclado (inyectando combustible gaseoso directamente sobre el flujo de aire turbulento). Los códigos CFD utilizados son CONVERGE (con mallado adaptativo) y OpenFOAM (con mallado tradicional), haciendo uso de aproximaciones RANS y LES.

Resum

En la societat actual hi ha cada vegada una major preocupació per les emissions contaminants, especialment el NO_x, fet que es tradueix en la concienciació de la població sobre aquest problema així com en lleis cada vegada més estrictes. En aquest context, sorgeixen estratègies d'injecció i combustió diferents que tracten de corregir el problema, com la Lean Direct Injection (LDI) per a motors de flux continu.

En el present Treball Fi de Màster es pretén avançar en l'estudi d'aquesta estratègia mitjançant resolució numèrica amb codis CFD. Els passos a seguir són: analitzar la injecció de combustible, la seua atomització i posterior evaporació, la seua interacció amb el flux turbulent a la cambra de combustió, l'eficiència global i les emissions vinculades a aquest procés. A partir d'uns resultats experimentals, obtinguts de la cambra de combustió LDI dissenyada en CORIA per al projecte KIAI, es tractarà de caracteritzar el camp de velocitats turbulent que es produeix a l'interior. Per a això es simularan i compararan casos no reactius, premesclat (injectant una barreja pobra d'aire i combustible) i posteriorment no premesclat (injectant combustible gasós directament sobre el flux d'aire turbulent). Els codis CFD utilitzats són CONVERGE (amb mallat adaptatiu) i OpenFOAM (amb mallat tradicional), fent ús d'aproximacions RANS i LES.

Abstract

There is growing concern in today's society about polluting emissions, especially the NO_x, and in consequence people are growing in awareness of this problem as well as competent authorities are making more restrictive laws. In this context different injection and combustion strategies arise trying to fix this problem, such as the Lean Direct Injection (LDI) for continuous flow engines.

In this Final Master's Project, we intend to make progress in the study of this strategy through numerical resolution with CFD codes. The steps to follow are: analyzing the fuel injection, its atomization and subsequent evaporation, its interaction with the turbulent flow in the combustion chamber, the global efficiency and the emissions produced in this process. Based on experimental results, obtained from the LDI combustion chamber designed in CORIA for the KIAI project, an attempt will be made to characterize the turbulent velocity field that is produced internally. To do this, non-reactive cases will be simulated and compared, premixed (injecting a poor mixture of air and fuel) and not premixed (injecting gaseous fuel directly into the turbulent air flow). The CFD codes used are CONVERGE (with adaptive mesh) and OpenFOAM (with traditional mesh), using RANS and LES approaches.

Great works are performed not by strength
but by perseverance.

Samuel Johnson

If you think you're too small to make a difference,
you haven't spent a night with a mosquito.

African Proverb

Agradecimientos

En primer lugar, al tutor Marcos Carreres y cotutor Mario Belmar por guiarme durante este proyecto, así como en mi estancia en el CMT. Me han brindado las herramientas necesarias para sacarlo adelante, me han ayudado con sus acertadas correcciones y gracias a ellos recordaré siempre con cariño la experiencia vivida y los conocimientos adquiridos en ella.

En segundo lugar, a mis compañeros de carrera Jose Manuel Sellés e Iván Olmeda que han hecho más llevaderos estos años de universidad, los trabajos en grupo, el período de prácticas, los descansos a mediodía para comer y las horas de gimnasio.

En tercer lugar, también quiero agradecer a mi perpetuo grupo de amigos (Adri, Carlos, Gabi, JC, JS, Juan y Tono) su paciencia cada vez que he tenido que rechazar un plan por trabajar y/o redactar, así como de hacerme disfrutar de su compañía en cada quedada. También a Lidia por soportar mis dolores de cabeza, por abrirme su casa, por su sonrisa perenne y su inagotable capacidad de darme conversación para distraerme de tanto trabajo; y a Irene por su apoyo incondicional, por creer en mí siempre dándome la fuerza y ánimo necesarios para no abandonar y seguir esforzándome.

Y finalmente, pero no menos importante, a mis padres Diego y Ana, así como a mi hermano Diego, por su dedicación e insistencia para que pueda llegar lo más lejos posible en la vida.

Índice general

Índice de figuras

Documento I

MEMORIA

Índice de documento

Capítulo 1

Introducción

1.1. CONTEXTO HISTÓRICO

Los primeros motores a reacción de uso aeronáutico fueron concebidos antes de la II Guerra Mundial, y en sus primeros desarrollos no eran capaces de proporcionar un gran empuje. Por este motivo, el principal desafío durante la época fue el de incrementar la cantidad de empuje obtenido en ellos, tratando de reducir su peso y tamaño, así como de hacerlos funcionar a altitudes elevadas.
Más adelante, en los años 80, un acusado incremento en el precio del combustible llevó a la industria a tratar de descubrir nuevas ideas para mejorar la eficiencia de los motores, pudiendo así disminuir el consumo de este tipo de máquinas.
Sin embargo, a partir de los años 90 y hasta la actualidad, el objetivo fundamental de la investigación en este campo pasa por reducir los niveles de emisiones contaminantes por debajo de los límites regulados por las autoridades.

Este cambio de perspectiva se debe a organizaciones como la OACI que, desde la primera regulación de emisiones NO_x en 1981, ha publicado a lo largo de los años diferentes estándares cada vez más exigentes para controlar la huella ambiental [1].

1.1.1. Impacto Ambiental De Las Aeronaves

No hay duda del inmenso beneficio económico y social que ha reportado el desarrollo de la aviación a nivel mundial. La conquista de los cielos ha supuesto un gran progreso en el transporte de pasajeros y bienes a gran velocidad, tumbando las fronteras geográficas y abriendo nuevas rutas a la economía global y las relaciones internacionales. [2]

No obstante, todas estas ventajas conllevan un coste, la repercusión en el medio ambiente y la salud. Ya en el año 1999 se advertía del peligro que suponía la aviación para el clima y la capa de ozono, teniendo en cuenta diferentes escenarios de crecimiento del sector y estimando las consecuencias hasta el año 2050 (figura 1.1). [3]

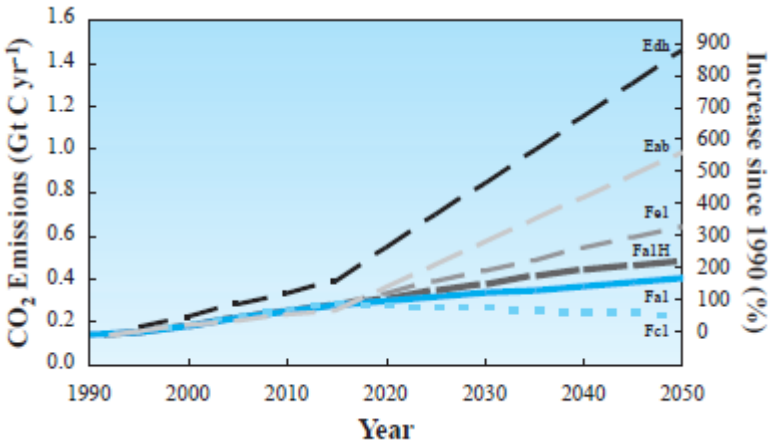


Figura 1.1: Emisiones de CO_2 del sector de aviación en seis escenarios diferentes según estimaciones del uso de combustible. Las emisiones se dan en Gt C (10^9 toneladas de carbono) al año, que se pueden convertir a Gt CO_2 multiplicando por 3'67.

En la actualidad se estima que el tráfico aéreo continuará creciendo durante al menos 10 años más, como lo ha estado haciendo desde 2010 (ver figura 1.2), con tasas de crecimiento anuales mayores del 5 % debidas al impulso económico de países en desarrollo y a la reducción de tarifas. Este hecho contribuiría así al aumento de emisiones y seguiría afectando de forma negativa al entorno.

Figura 1.2: Número de vuelos realizados por la industria global de aerolíneas desde 2004 hasta 2018 (en millones) [4].

Aunque las aeronaves únicamente participan en un pequeño porcentaje sobre el total de emisiones NO_x generadas en el planeta como se puede observar en la figura 1.3, su efecto se multiplica debido a que son emitidas en regiones muy concentradas: en las cercanías de los aeropuertos provocando nieblas tóxicas y en la atmósfera a grandes altitudes [5]. Además, como muestra la tabla 1.1, las emisiones están relacionadas con los regímenes de operación del motor.

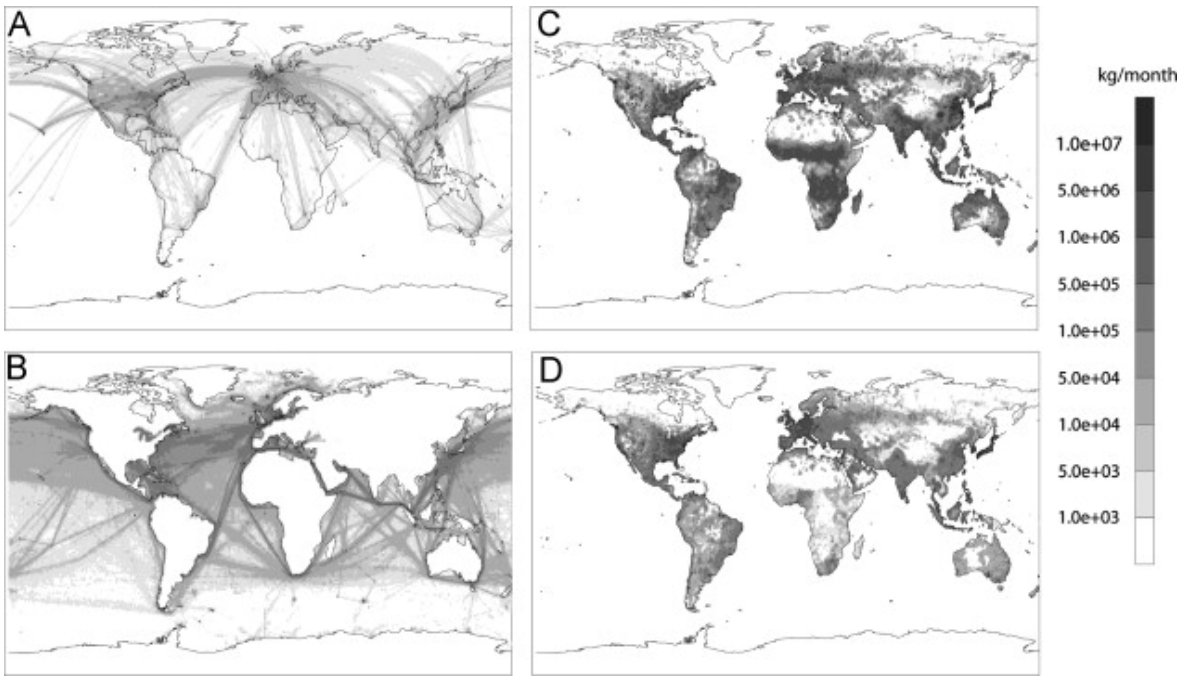


Figura 1.3: Distribución geográfica de flujos NO_x en el año 2000 en kg/mes para: (A) aviación, (B) envíos internacionales, (C) emisiones antropogénicas no relacionadas con el transporte y (D transporte por carretera [6, 7].

Tabla 1.1: Emisiones que más afectan de los motores aeroderivados [8]

Categoría	Operación del motor a la que se asocia principalmente
Humo (hollín)	Despegue y ascenso
Hidrocarburos sin quemar (HC) Monóxido de carbono (CO)	Baja potencia, especialmente ralentí en tierra
Óxidos de nitrógeno (NO_x)	Alta potencia, incluyendo crucero
Dióxido de carbono (CO_2)	Todos los ajustes de operación

Por estos motivos es importante contribuir al desarrollo de motores de aviación más eficientes, silenciosos y seguros, capaces de emitir menos contaminantes sin disminuir sus prestaciones.

1.1.2. Reducción De Emisiones Contaminantes

Recientemente, en la 10ª reunión celebrada en 2016 por su Comité en Protección Ambiental en Aviación (CAEP/10), se ha incluido una limitación en las emisiones de partículas de hollín ultra-finas y CO_2 de las aeronaves, además de una propuesta de reducción anual del 2 % en el consumo de combustible hasta 2050 [9].
De la misma manera, el ACARE en su programa Visión 2020, ha impuesto objetivos ambiciosos para el crecimiento sostenible del transporte aéreo: reducción del 50 % en emisiones de CO_2 y del 80 % en NO_x desde 2001 hasta 2020 [2].

Una de las líneas de investigación que más se está imponiendo en la actualidad en cuanto a combustión en los motores turbina de gas es el LDI (Lean Direct Injection), de la que se hablará en la sección 2.2.4, ya que ha conseguido alcanzar unos objetivos ambiciosos en la reducción de emisiones de NO_x. Esto la convierte en una tecnología apropiada para continuar con el desarrollo del transporte aéreo.

1.2. OBJETIVOS

Los objetivos del presente trabajo se enumeran a continuación:

- La calibración y validación de dos modelos CFD, uno con mallado tradicional (OpenFOAM©) y otro con técnicas más modernas (CONVERGE©), mediante comparación de resultados con los obtenidos de manera experimental.
- Caracterización de estructuras turbulentas del flujo no reactivo en el interior de una cámara de combustión en quemadores LDI. Las simulaciones se llevarán a cabo con modelos RANS y LES, obteniendo información sobre flujo premezclado y no premezclado para el posterior estudio de la atomización, mezcla y combustión.
- Sentar las bases de una herramienta potente que permita el estudio de atomización, mezcla y combustión con relativamente pocos recursos. El uso de esta herramienta podrá utilizarse en la búsqueda de geometrías y mecanismos que resulten en reducción de emisiones y mejor eficiencia para los objetivos que la legislación propone en el futuro de la aviación.

Capítulo 2

Fundamentos de combustión en motores aeroderivados

La tecnología aplicada al proceso de combustión se ha ido desarrollando de forma gradual y continuada, sin cambios drásticos, dejando tras de sí motores aeronáuticos cuyas cámaras de combustión son muy similares en tamaño, forma y apariencia general.

2.1. PRODUCTOS DE LA COMBUSTIÓN

El escape de una turbina de gas se compone de CO, dióxido de carbono (CO_2), vapor de agua (H_2O), hidrocarburos sin quemar (UHC), partículas (principalmente carbón), óxidos de nitrógeno (NO_x) y exceso de oxígeno y nitrógeno atmosféricos.

El CO_2 y H_2O no son considerados contaminantes porque son una consecuencia natural de la combustión de hidrocarburos. Sin embargo, ambos contribuyen al calentamiento global y sólo pueden reducirse quemando menos combustible. Ahí reside la importancia de mejorar la eficiencia térmica, que además de disminuir la contaminación tiene el mismo efecto sobre los costes.
El CO reduce la capacidad de la sangre de absorber oxígeno y puede causar asfixia y hasta la muerte en altas concentraciones.
Los UHC, además de ser tóxicos, se combinan con los NO_x formando niebla al nivel del suelo, causando daños a la vegetación y contribuyendo a la lluvia ácida.
Las partículas (hollín y humo) causan problemas de visibilidad y suciedad en la atmóstfera. Aunque no es considerado tóxico a los pequeños niveles que se emiten, existen estudios que relacionan el asma y otras enfermedades respiratorias a la concentración de estas partículas en el aire [10]. Además, algunos supresores de humo contienen metales pesados (como el bario) que suponen más polución por los gases de escape.

2.1.1. Mecanismos De Formación NO_x

La mayoría del óxido nítrico (NO) que se forma en la combustión se oxida posteriormente generando NO_2, de modo que conviene juntar ambos y expresar los resultados en términos de NO_x. Éste puede ser producido por 3 mecanismos diferentes: thermal NO_x, prompt NO_x y fuel NO_x. [11, 12]

Thermal NO_x. Es producido por la oxidación del nitrógeno atmosférico en regiones con gases a alta temperatura. Es un proceso endotérmico y ocurre de manera significativa por encima de 1850K [13]. La reacción se representa mediante el Mecanismo Extendido de Zeldovich:

Óxido Nítrico	Óxido Nitroso
$O_2 \leftrightarrow 2\ O$	$N_2 + O \leftrightarrow N_2O$
$N_2 + O \leftrightarrow NO + N$	$N_2O + O \leftrightarrow NO + NO$
$N + O_2 \leftrightarrow NO + O$	$N_2O + H \leftrightarrow NO + NH$
$N + OH \leftrightarrow NO + H$	$N_2O + CO \leftrightarrow NO + NCO$

Prompt NO_x. Es una forma de thermal NO_x que se forma en lugares cercanos al frente de llama cuando se oxidan productos intermedios de la combustión como el HCN.

Mecanismo Prompt
$N_2 + CH \leftrightarrow HCN + N$
$N + O_2 \leftrightarrow NO + O$
$HCN + OH \leftrightarrow CN + H_2O$
$CN + O_2 \leftrightarrow NO + CO$

Fuel NO_x. Se produce al quemar combustible que contiene nitrógeno. Algunos ejemplos típicos son tipos de gas natural que llevan nitrógeno molecular y algunos combustibles sintéticos que lo contienen en forma de amoníaco. Cuando combustionan, los enlaces del nitrógeno se rompen y algunos pueden formar NO_x.

2.2. ESQUEMAS DE COMBUSTIÓN CON BAJAS EMISIONES

En la Figura 2.1 se observa que el índice de emisiones NO_x es función de la temperatura de llama, del dosado relativo de la mezcla aire-combustible y del tiempo de residencia. De esta manera, se pueden tomar algunas estrategias para reducirlo: utilizando mezclas aire-combustible ricas ($\phi > 1.2$) o pobres ($\phi < 0.8$), o reduciendo el

tiempo de residencia en condiciones estequiométricas. Por su parte, el efecto de la presión en la producción de NO_x es bastante menor y función de la configuración particular del motor, tal y como ha quedado constatado en diversos estudios [14, 15].
Otra forma de disminuir las emisiones de NO_x consiste en evitar altas temperaturas, añadiendo más aire a la mezcla o consiguiendo homogeneizarla para eliminar los puntos calientes. Sin embargo, aunque son muy útiles y está probada su eficacia, estas estrategias no son ideales ya que generalmente pueden reavivar los problemas de emisiones de CO y UHC [11].

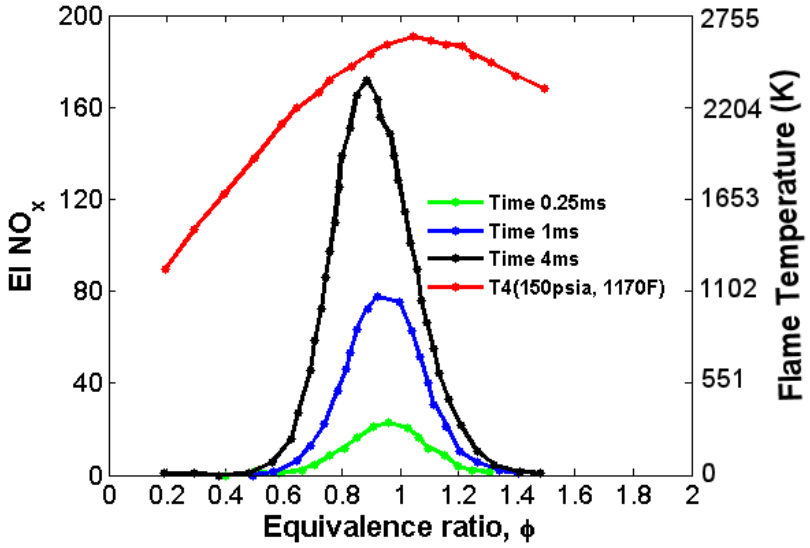


Figura 2.1: Índice de emisiones NO_x en relación al dosado relativo, temperatura de llama, y tiempo de residencia (cortesía de Pratt & Whitney).

2.2.1. Rich Burn/Quick-Quench/Lean-Burn (RQL) Combustion

Este diseño es el más común y fue desarrollado en 1980 [16] con el objetivo de reducir las emisiones de NO_x en las turbinas de gas empleadas en aviación.

El principio de la combustión RQL se muestra en la figura 2.2. En primer lugar se genera una zona de dosado rico ($1.2 < \phi < 1.8$) para asegurar la estabilidad de las reacciones, que por las concentraciones de oxígeno y temperaturas relativamente bajas, consigue minimizar la producción de NO_x *thermal* y *prompt*.
A continuación, la mezcla abandona esta zona primaria e interacciona con chorros de aire secundario que la enfrían rápidamente y reducen el dosado. Si se hace con suficiente eficacia, se consigue evitar la formación de regiones localmente estequiométricas, y con ello se mantiene baja la generación de NO_x.

Finalmente, en la zona de mezcla pobre ($0.5 < \phi < 0.7$) se oxidan las altas concentraciones de hidrocarburos y otros productos intermedios de la combustión, tratando de no sobrepasar temperaturas de 1900K que conllevarían una elevadísima formación de NO_x. No obstante, se debe asegurar una temperatura lo suficientemente alta que permita consumir los restos de CO, UHC y hollín que no pueden ser expulsados en el escape [17].

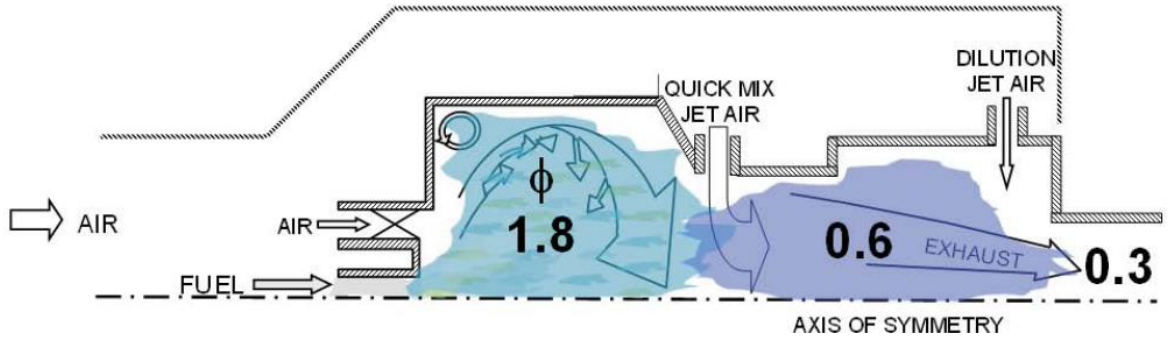


Figura 2.2: Esquema quemador RQL.

Sin embargo, este esquema presenta algunos inconvenientes. En primer lugar su tamaño, peso y complejidad no son factores beneficiosos para la aeronáutica. También tienen un potencial limitado para cumplir con las futuras normativas, ya que aumentar el dosado rico podría resultar más beneficioso en cuanto a emisiones de NO_x, pero el exceso de combustible perjudicaría las emisiones de hollín. Además, la fase de mezcla rápida es crucial para mantener bajos los índices de NO_x, y este proceso no es trivial. Otra desventaja es la dificultad para mantener un elevado rendimiento en el ciclo termodinámico del motor, debiéndose encontrar un compromiso entre el rendimiento y las emisiones.

2.2.2. Combustión Escalonada (Staged Combustion)

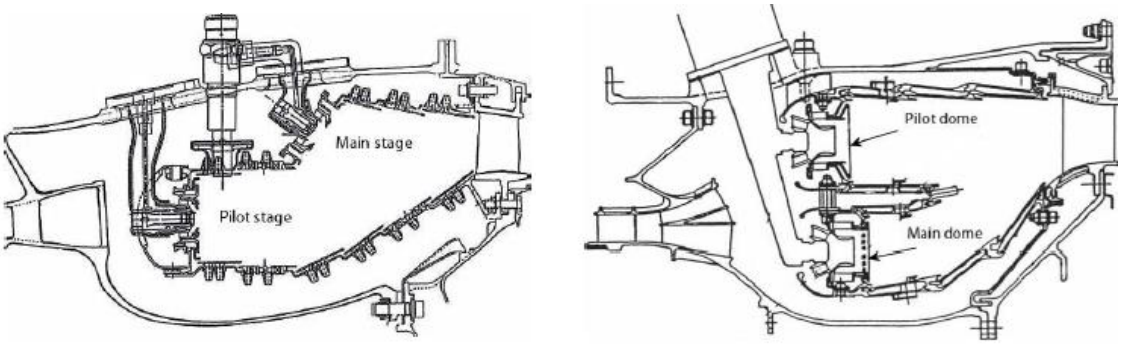
En este tipo de esquemas la distribución del flujo de aire es constante a lo largo de la cámara de combustión, no así el flujo de combustible que varía para tratar de mantener la temperatura fija en cada etapa. Su principal modelo de inyección es conocido como Inyección de Combustible Selectiva [18], por el cual se suministra el combustible mediante combinaciones concretas de inyectores.

Este método permite reducir la cantidad mínima de combustible necesaria para mantener una llama estable (lean blowout limit), permitiendo operar con mezclas más pobres y reduciendo de esta manera las emisiones de CO y UHC, aunque no las de NO_x. Además, esta técnica puede causar un enfriamiento en las regiones exteriores de las zonas de combustión, provocando una distribución no uniforme de la temperatura de salida y afectando por tanto a la eficiencia del motor.

Con el objetivo de paliar estas limitaciones, se desarrollaron cámaras de combustión escalonadas en las que se emplean dos o más etapas:

- En serie, como muestra la figura 2.3(a). [19, 20]
- En paralelo, como muestra la figura 2.3(b). [21, 22]

Cada una está diseñada para optimizar aspectos concretos del rendimiento de la cámara de combustión.



(a) En serie (b) En paralelo

Figura 2.3: Cámaras de combustión escalonadas.

2.2.3. Lean Premixed Combustion

Este esquema comparte con los anteriores la combustión a un dosado bajo en su objetivo de reducir las emisiones de NO_x. Cuanto más cercano es el umbral para una llama estable al límite de apagado de llama, menores son las posibilidades de producir óxidos de nitrógeno. A diferencia de los otros esquemas, éste incluye el concepto de la premezcla, por el cual se introduce una mezcla homogénea de aire y combustible en la zona de combustión. [12]
En las turbinas de gas industriales se ha demostrado su eficacia en comparación con el resto de filosofías que operan con llamas de difusión, no obstante, esto ha sido posible con combustible gaseoso (concepto LPM). Como la mayoría de motores de aviación operan con combustible líquido y funcionan en condiciones de operación más exigentes, la aplicación de este esquema requiere de algunas adaptaciones.

Lean Premixed Prevaporized. Fue concebido a principios de los años 1970, e incluye una sección de premezcla justo antes de la zona primaria de combustión. En ella se mezcla aire y combustible, en lugar de tenerlo ya almacenado, para después inyectarlo junto. Sin embargo, los quemadores que utilizan LPP requieren una conversión de llama premezclada a difusiva en condiciones de intensidad reducida para

mantenerla estable. Esta capacidad para cambiar de un tipo de operación a otro introduce complejidad en el diseño y aumenta los costes. Además existen varios peligros potenciales, nada deseables en aviación general y menos todavía en aviación comercial: un límite de estabilidad estrecho, alta generación de ruido y posibilidad de auto-ignición y retroceso de llama en la zona premezclada. [23]

2.2.4. Lean Direct Injection

Este esquema surge en 1988 [24, 25, 26, 27] como una alternativa a la combustión LPP y los riesgos operacionales que entraña. En ella se inyectan el aire y el combustible directamente en la cámara de combustión, sin premezcla previa, de nuevo con un dosado relativo bajo (entre 0.6 y 0.65) cercano al límite de apagado de llama [28]. De esta manera se elimina la zona de premezclado, y con ello la posibilidad de que se produzcan autoencendidos o retrocesos de llama.

El swirler (sección 2.4.2) toma una importancia notable ya que el proceso de mezcla y combustión depende casi en su totalidad del campo de velocidades generado. El verdadero desafío de los quemadores LDI es conseguir que la mezcla sea lo más homogénea posible, evitando picos de temperatura locales en la cámara que disparen las emisiones de NO_x.

Al introducir una gran proporción de aire de manera coaxial a la inyección de combustible, se puede prescindir de las zonas de dilución presentes en las cámaras de combustión convencionales, reduciendo significativamente las dimensiones del motor [29]. El diseño de estos quemadores es mucho más simple y ligero, pero requieren análisis detallados de la dinámica del flujo y la liberación de calor producida en las regiones de mezclado. [23]

El potencial de bajas emisiones NO_x de los quemadores LDI ha sido demostrado también por [30, 31, 32]. Sus resultados mostraron que las emisiones eran comparables a las de los quemadores LPP.

Una manera de mejorar el proceso de mezcla es incrementar la diferencia de presión a través del inyector de combustible, generando mayor turbulencia que lo descomponga. No obstante, esto supondría una pérdida de eficiencia en el sistema.

Multi-Point Lean Direct Injection. No es de extrañar que, habiendo comprobado la eficacia de un quemador LDI, surja la idea de ir más allá y probar con varias entradas para la inyección del combustible. Este concepto es el conocido como MPLDI (figura 2.4) que ha sido sometido a experimentos en las instalaciones del NASA Glenn Research Center con varias configuraciones [33, 34, 35].

Dividiendo un inyector de combustible grande en varios pequeños, es posible incrementar la eficacia y uniformidad de la mezcla. Cada uno de ellos está rodeado por un

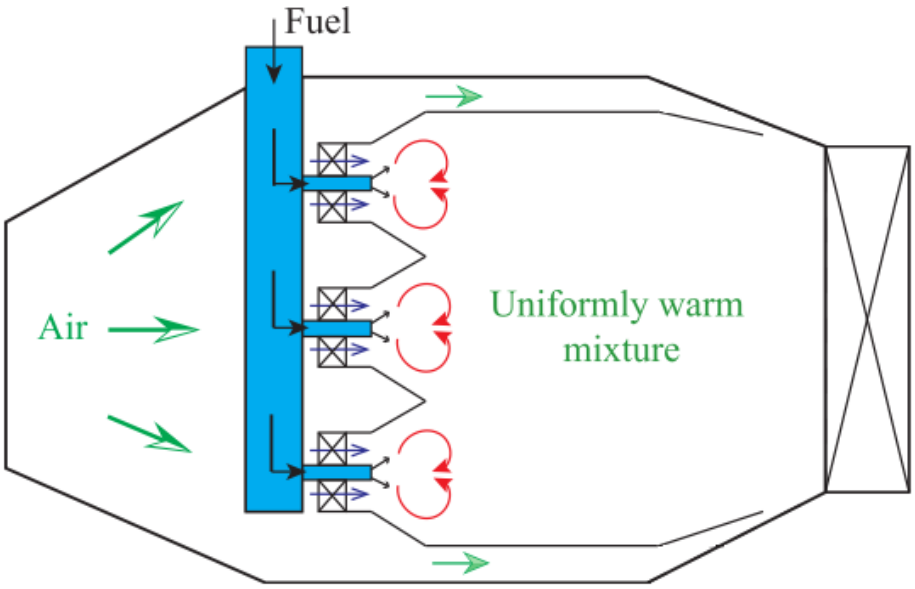


Figura 2.4: Esquema de combustor MPLDI.

swirler que crea un alto grado de turbulencia para atomizar los chorros de combustible en gotas muy finas. La dispersión de éstas y las zonas de cizalladura resultantes de los swirlers adyacentes contribuyen a la homogeneización de la mezcla, al contrario que en los LDI de un único inyector limitados por la pared de la cámara. La configuración también permite controlar los inyectores individualmente para redistribuir el combustible y reducir inestabilidades. [36]

2.3. REQUERIMIENTOS DE LA CÁMARA DE COMBUSTIÓN

Los requerimientos generales según Lefebvre [11] se detallan a continuación:

- Máxima eficiencia en la combustión.
- Ignición suave y fiable, tanto en tierra como tras el apagado de llama a gran altitud.
- Amplio rango de estabilidad (para variaciones grandes de la presión y de los dosados locales).
- Pérdidas de presión reducidas.
- Distribución de temperatura a la salida de la cámara controlada, para que no afecte negativamente a la vida de los álabes de turbina y la tobera.
- Bajas emisiones contaminantes.
- Tamaño y forma adecuados del motor para su aplicación concreta.

- Diseño para costes mínimos de fabricación y mantenimiento.
- Durabilidad
- Capacidad de funcionamiento con combustibles variados.

Concretamente, en el caso de los motores de aviación el peso y tamaño son muy importantes, mientras que para motores industriales lo es la capacidad para funcionar con diferentes combustibles. Además, independientemente de su aplicación se consideran primordiales las bajas emisiones contaminantes.

2.4. ELEMENTOS BÁSICOS DE LA CÁMARA DE COMBUSTIÓN

En este apartado se dará una visión general de los elementos básicos de la cámara de combustión en turbinas de gas. Existen otros elementos a parte de los mencionados pero que no son relevantes en referencia a los objetivos que se plantean en el presente trabajo.

2.4.1. Difusor

La pérdida de presión en la cámara de combustión ocurre fundamentalmente cuando se añade calor a un flujo de gas según la fórmula 2.1, donde T_3 y T_4 son respectivamente las temperaturas a la entrada y salida de la cámara. Como ésta es proporcional al cuadrado de la velocidad del aire, con velocidades a la salida del compresor de 170 m/s y un ratio de temperaturas de 2.5, las pérdidas originadas pueden llegar a suponer el 25 % de la presión alcanzada. Para reducirlas a un nivel aceptable (hasta la quinta parte) se utiliza el difusor [11].

$$\Delta P_{hot} = 0.5\rho U^2 \left(\frac{T_4}{T_3} - 1\right) \tag{2.1}$$

En su forma más simple, el difusor es un pasaje divergente por donde el flujo se decelera, convirtiendo la velocidad en presión estática. La eficiencia de este proceso es muy importante ya que las pérdidas ocurridas se manifiestan en una pérdida de presión total, y por tanto, en el rendimiento del ciclo.

Por una parte, en difusores largos con ángulo de divergencia pequeño, la pérdida de presión se debe principalmente a la fricción con las paredes. Por otro lado, los difusores cortos con mayor ángulo de divergencia, tienen menor longitud y fricción, pero sufren principalmente las pérdidas por separación de la capa límite. Lo ideal en el diseño de estos componentes es un equilibrio entre ambas geometrías, ya que para una relación de áreas determinada existe un ángulo de divegencia óptimo con pérdidas mínimas que suele estar entre 6-12°.

2.4.2. Swirler

Para posibilitar la combustión es necesario un flujo recirculatorio de aire y productos quemados que cumple dos funciones: evitar el desprendimiento de llama (por el campo de velocidades, figura 2.5) y mantener una fuente contínua de ignición para la mezcla entrante [37]. Esto puede obtenerse mediante varias soluciones, pero concretamente en este trabajo se optará por el swirler, porque es el que dispone la geometría a estudiar y ofrece ventajas como regiones de fuertes tensiones cortantes, alta turbulencia y tasas de mezcla rápidas.

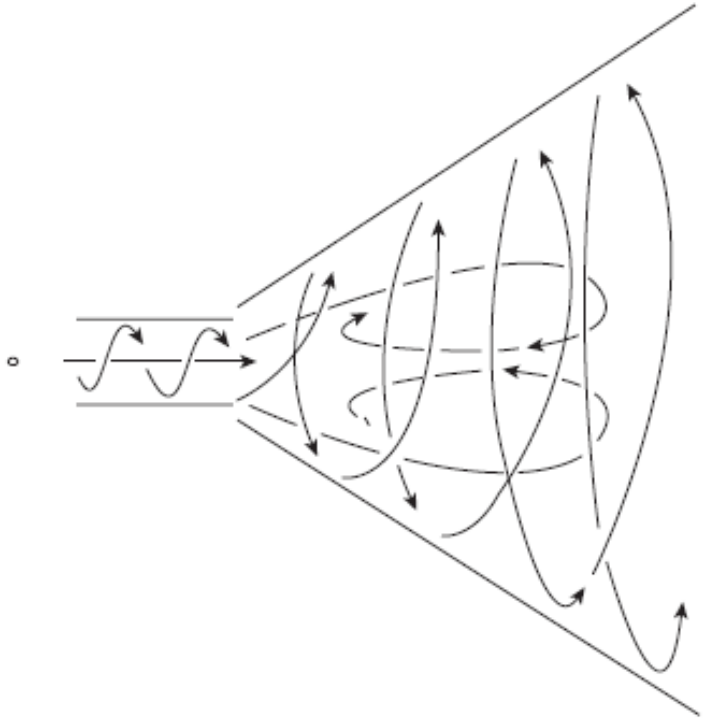


Figura 2.5: Campo de velocidades provocado por el swirler. El flujo en forma de espiral es capaz de producir una zona central toroidal de recirculación con alta turbulencia.

Las características del flujo de aire producido por el swirler y sus aplicaciones prácticas han sido estudiadas en [38, 39], siendo la última de especial interés porque se dirige directamente a los tipos de swirlers más relevantes en turbinas de gas, proporcionando información valiosa para el diseño de estos componentes.

Número De Swirl. Las características del flujo rotatorio presente a la salida del swirler dependen del número de palas, del ángulo de salida de éstas y de la pérdida de presión experimentada. En [38] se propone un criterio no dimensional para caracterizar la cantidad de rotación que se transmite al flujo, el número de swirl.

El número de swirl se define como la proporción entre las componentes tangencial y axial de la cantidad de movimiento del flujo axial normalizado con un radio característico (ecuación 2.2)

$$S = \frac{G_\phi}{G_x \; r_t} \tag{2.2}$$

donde las componentes de cantidad de movimiento son (ecuaciones 2.3 y 2.4)

$$\begin{aligned} G_\phi &= \int_{r_h}^{r_t} (Wr)\rho U(2\pi r)dr \\ &= (U \; tan\theta)2\pi\rho U \int_{r_h}^{r_t} r^2 dr \\ &= \frac{2}{3}\pi\rho U^2(r_t^3 - r_h^3) \end{aligned} \tag{2.3}$$

$$G_x = \int_{r_h}^{r_t} \rho U^2(2\pi r)dr + \int_{r_h}^{r_t} p(2\pi r)dr \tag{2.4}$$

y r_t y r_h son, respectivamente, el radio exterior e interior del swirler.

Esta expresión contiene en G_x el término de presión, que por la dificultad que plantea de ser calculado se suele omitir, definiendo el número de swirl de esta manera

$$S = \frac{G_\phi}{G'_x \; r_t} \tag{2.5}$$

donde la nueva G'_x se calcula como

$$\begin{aligned} G'_x &= \int_{r_h}^{r_t} \rho U^2(2\pi r)dr \\ &= 2\pi\rho U^2 \int_{r_h}^{r_t} r dr \\ &= \pi\rho U^2(r_t^2 - r_h^2) \end{aligned} \tag{2.6}$$

Finalmente, operando estos términos y simplificando se obtiene

$$S = \frac{2}{3} tan\theta \left[\frac{1 - (r_h/r_t)^3}{1 - (r_h/r_t)^2}\right] \tag{2.7}$$

donde θ corresponde al ángulo que forman las palas del swirler respecto a la dirección axial. En los swirlers axiales de álabes planos este ángulo es constante, mientras que en aquellos con álabes torsionados el ángulo es nulo en la sección de entrada y θ en la

sección de salida, presentando un valor intermedio las secciones centrales en función de la ley de torsión.

Basándose en el número de swirl, se puede dividir el flujo en dos regiones: débil ($S < 0.6$) y fuerte ($S > 0.6$). En el flujo rotatorio débil los gradientes de presión axiales son insuficientes para causar recirculación. Opuestamente, el flujo rotatorio fuerte tiene gradientes altos de presión en dirección radial y axial cerca de la salida del swirler, consiguiendo originar dicha zona central de recirculación.

Tipos De Swirler. Existen varios tipos de swirler, pero los dos más comunes son axial y radial, cuyos esquemas se muestran en la figura 2.6 y se suelen encontrar dispuestos de manera individual o concéntrica.

El primero de ellos se emplea generalmente en quemadores convencionales de combustible líquido y es relativamente sencillo de fabricar ya que suele ser plano. Sin embargo añadir curvatura a sus palas puede mejorar su desempeño y propiedades aerodinámicas. Por otro lado el swirler radial se enfoca tanto a quemadores convencionales como a los DLE (dry low emissions) también llamadas cámaras de combustión con emisiones bajas.

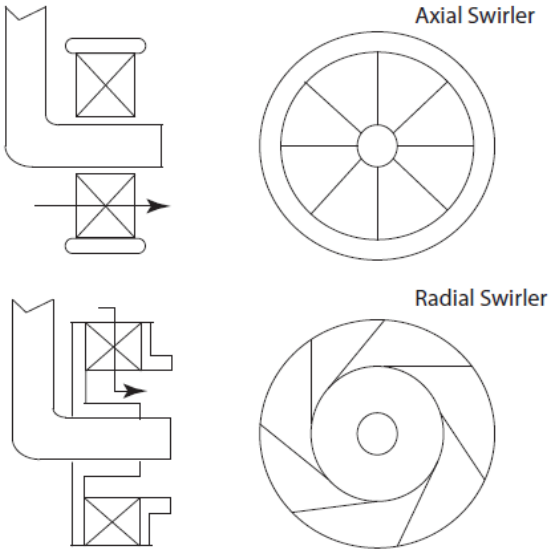


Figura 2.6: Esquema de swirler axial y radial.

Existen estudios relevantes sobre swirlers axiales como [40], donde emplean 6 palas con un ángulo 60 grados en combinación con tobera convergente-divergente.

En el caso de los swirlers radiales hay estudios como [24, 25, 26, 27] con aplicación directa en quemadores LDI. Sus investigaciones incluyen el efecto de uno y varios swirlers contrarrotantes, y de la inyección del combustible en las emisiones de NO_x, aunque utilizan gas natural en lugar de combustible líquido.

Sin embargo, ambos tipos de swirlers se utilizan comúnmente en quemadores de turbinas de gas. Algunas configuraciones han sido evaluadas experimentalmente en [41] usando gas natural y en [42] con combustible líquido. En ellos se observa que en cuanto a emisiones, la actuación del swirler radial es mejor que el axial. Las propiedades de la zona de recirculación generada por cada uno podría ser el factor que más contribuye. Aunque ambos tenían el mismo número de swirl, el radial producía una zona central de recirculación más consistente que su homólogo axial, debido a las diferencias en la relación de áreas, altura del canal y velocidad de salida del swirler.
Por otro lado, en [43] se lleva a cabo un estudio numérico sobre el efecto de ambas configuraciones de swirler. En él se advierte que, al contrario que el modelo estrecho y centralizado que produce el swirler axial, el radial provee un patrón de remolinos más disperso, haciendo más probable que el combustible llegue a chocar con la pared.

2.4.3. Inyector

Aunque en la sección 2.4.2 se ha hablado de ambos tipos de combustible (gas y líquido), la gran mayoría de los motores turbina de gas en la industria aeronáutica operan con combustibles líquidos debido a su elevada densidad energética; el más utilizado es el keroseno por su alto poder calorífico y bajo punto de fusión. El combustible se inyecta en forma de spray, proceso que puede tener lugar en la corriente del flujo de recirculación (inyección directa) o en una zona anterior a la entrada de la cámara de combustión (premezclado).

Los sistemas de inyección tienen como objetivos garantizar un suministro de combustible constante e ininterrumpido a la cámara de combustión, generar las condiciones adecuadas para facilitar la evaporación de las gotas líquidas y agilizar el proceso de mezclado con el aire para una combustión eficiente.

Teniendo en cuenta lo anterior, una forma de lograr mejorar la transferencia de masa y calor entre la fase líquida y gaseosa (la evaporación del combustible) consiste en producir una nube de gotas, maximizando la superficie de contacto entre las fases. Se precisa un tamaño de gotas reducido, lo que provee un descenso importante de la energía necesaria para la ignición. Por ello se debe generar una zona altamente turbulenta en la región cercana a la inyección que facilite los procesos de atomización y mezclado de ambas fases.

Existen dos tipos de atomización:

- La atomización primaria consiste en la desestabilización de la película líquida emergente del inyector para generar ligamentos.
- La atomización secundaria se basa en la rotura de los ligamentos mencionados en pequeñas gotas.

Según los procesos mecánicos mediante los cuales se realizan la atomización primaria y secundaria, existen diversas tecnologías de inyectores:

- Pressure Swirl Atomiser (figura 2.7). Esta tipología aprovecha la diferencia de presión existente entre la línea de combustible presurizada y la cámara de combustión para forzar el paso de la película líquida a través de un pequeño orificio. Su principal ventaja radica en que el grado de atomización conseguido no es función del flujo de aire, es decir, el tamaño final de las gotas es independiente de las condiciones de operación del motor. Por el contrario, entre las desventajas se encuentra la posible obstrucción de los orificios de inyección debido a su reducido tamaño, requiriendo de sistemas de presurización adyacentes que elevan la complejidad y el peso del motor.

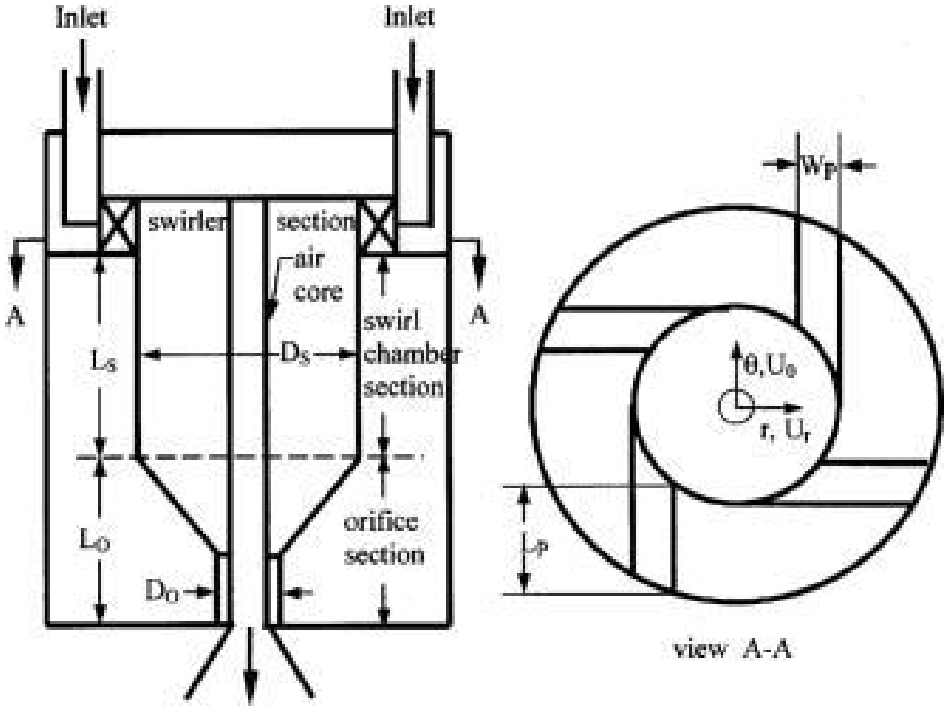


Figura 2.7: Variante del Pressure Swirl Atomiser más sencilla, conocida como SIMPLEX. [44]

- Vaporizadores. Se trata de unos tubos localizados en la zona de combustión primaria a través de los cuales se inyecta el combustible evaporado a causa del flujo de calor proveniente de la llama. No obstante, su uso es muy limitado debido a la corta vida útil del sistema fruto de los elevados esfuerzos térmicos a los que se encuentra sometido.
- Air-blast Atomiser (figura 2.8). El uso de esta tecnología está ampliamente extendido en la industria por tratarse de un sistema muy simple que solamente depende del flujo de aire para operar. El combustible líquido se inyecta de forma

cónica sobre una pared cilíndrica donde se encuentra con un primer chorro de aire torbellinado que lo rompe en ligamentos y lo arrastra hasta el orificio de salida. En la sección de salida del inyector un segundo chorro de aire torbellinado en sentido contrario al primero genera una fuerte zona de cortadura y completa la atomización secundaria. Por tanto, el combustible inyectado en la cámara de combustión se encuentra totalmente atomizado y parcialmente mezclado, mejorando así las prestaciones respecto a los inyectores pressure swirl.

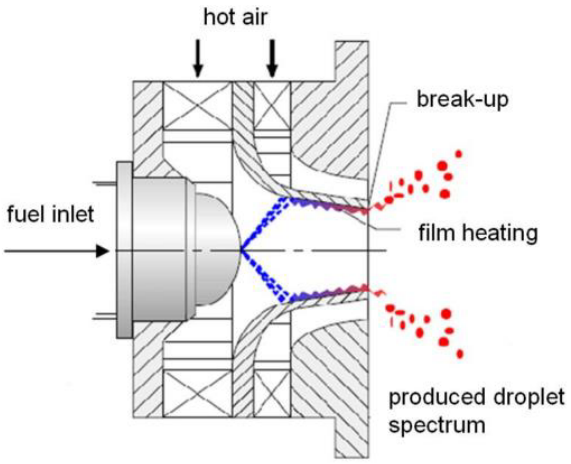


Figura 2.8: Air-blast Atomiser. [45]

Capítulo 3

Modelado mediante Mecánica de Fluidos Computacional

La dinámica de fluidos computacional o CFD por sus siglas en inglés (Computational Fluid Dynamics) es una técnica de simulación mediante métodos numéricos para resolver y analizar problemas sobre mecánica de fluidos. Esta disciplina se encuentra en auge y es muy utilizada en casos donde resulta muy complicado o costoso realizar experimentos.

3.1. CONTEXTO HISTÓRICO

Comenzó a utilizarse en los años 60, enfocado sobre todo para el diseño y fabricación en el sector de la industria aeroespacial, pero rápidamente se extendió a otras áreas de investigación durante los años 80-90 llegando a establecerse como parte esencial en el desarrollo tecnológico e industrial.

Algunos ejemplos donde se aplican estas simulaciones numéricas son: en el desarrollo de motores tanto de aviación (figura 3.1) como automoción, en la Fórmula 1 para mejorar el diseño aerodinámico de los automotores, en climatología para predecir el movimiento de las corrientes eólicas, en la medicina para investigar el flujo sanguíneo en vasos y arterias, entre muchas otras áreas.

Se espera que en un futuro éstas sean capaces de reemplazar a los métodos experimentales debido a que normalmente son muy costosos, a menudo son irrepetibles y generalmente son muy dependientes de la instrumentación utilizada. Opuestamente, una simulación puede repetirse innumerables veces, se puede tomar medidas en cualquier punto del dominio y se puede generar múltiples visualizaciones muy útiles para ingenieros y diseñadores.

Sin embargo, existe un importante límite en esta tecnología, y es que el poder computacional actual es limitado. A pesar de estar en constante evolución desde hace

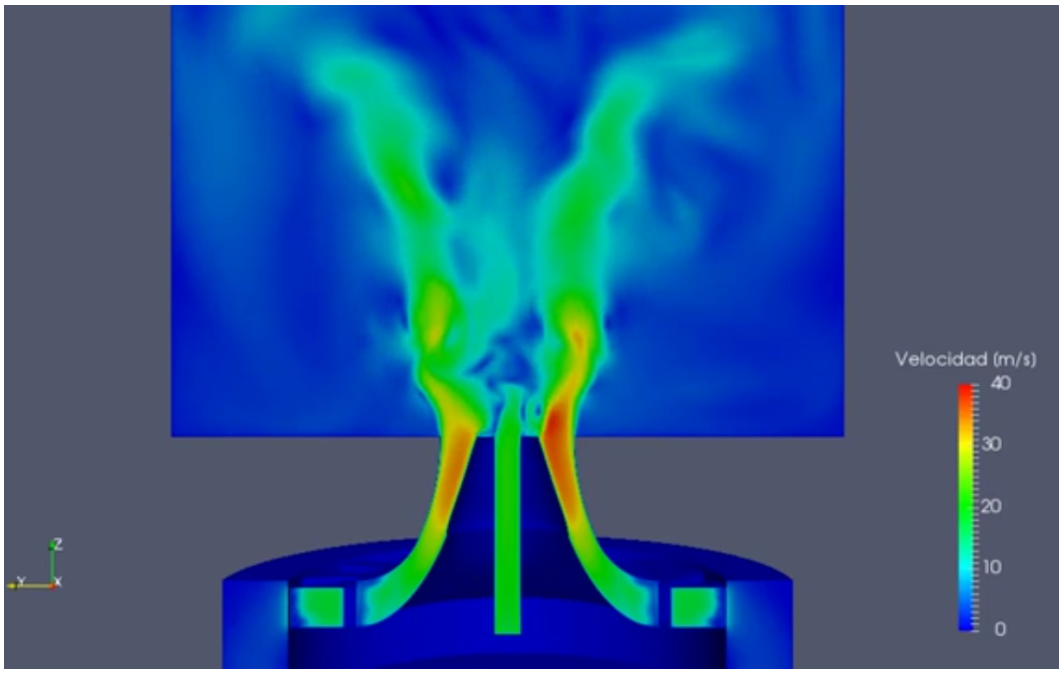


Figura 3.1: Campo de velocidades del flujo a la entrada de una cámara de combustión calculado mediante CFD.

décadas, las simulaciones de CFD necesitan manejar grandes cantidades de datos (la geometría del problema se divide en millones de elementos, con sus respectivas variables físicas y las ecuaciones complejas que se deben resolver en cada uno de ellos) haciendo que lleguen a durar semanas, meses e incluso años.

3.2. ECUACIONES FUNDAMENTALES

Las ecuaciones que gobiernan pueden obtenerse en diferentes formas (diferencial, integral, etc). Para la mayor parte de la teoría aerodinámica no hay grandes diferencias entre unas y otras, pero para un algoritmo concreto de CFD puede ser vital utilizar la más adecuada. La convergencia, las oscilaciones, la estabilidad o los errores pueden depender exclusivamente de la forma escogida para representar las ecuaciones. [46]

3.2.1. Conservación De La Masa

La ecuación de conservación de la masa o ecuación de continuidad, en su forma no estacionaria y tridimensional en un flujo compresible se plasma en la ecuación 3.1.

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot (\rho \vec{V}) = \frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y} + \frac{\partial (\rho w)}{\partial z} = 0 \tag{3.1}$$

El primer término representa la tasa de variación de la densidad en el tiempo, masa por unidad de volumen, mientras que el segundo (que se divide en tres, uno por cada dimensión) representa el término convectivo, esto es, el flujo másico neto saliente del elemento a través de sus fronteras.

3.2.2. Cantidad De Movimiento

La Segunda Ley de Newton enuncia que la tasa de cambio de la cantidad de movimiento de una partícula fluida equivale a la suma de las fuerzas ejercidas sobre ella. Éstas pueden ser tensiones normales o de presión p, tensiones viscosas τ y fuerzas volumétricas. De ahí se extraen las 3 ecuaciones de cantidad de movimiento (3.2-3.4), una por cada dimensión.

$$\rho\frac{Du}{Dt} = \frac{\partial(-p+\tau_{xx})}{\partial x} + \frac{\partial\tau_{yx}}{\partial y} + \frac{\partial\tau_{zx}}{\partial z} + S_{Mx} \tag{3.2}$$

$$\rho\frac{Dv}{Dt} = \frac{\partial\tau_{xy}}{\partial x} + \frac{\partial(-p+\tau_{yy})}{\partial y} + \frac{\partial\tau_{zy}}{\partial z} + S_{My} \tag{3.3}$$

$$\rho\frac{Dw}{Dt} = \frac{\partial\tau_{xz}}{\partial x} + \frac{\partial\tau_{yz}}{\partial y} + \frac{\partial(-p+\tau_{zz})}{\partial z} + S_{Mz} \tag{3.4}$$

La presión tiene signo negativo por convenio, al tratarse de una fuerza de compresión. En estas ecuaciones se tienen en cuenta las tensiones superficiales y, además, la presencia de términos fuente (S_{Mx}, S_{My} y S_{Mz}), que representan la acción debida a las fuerzas de volumen como podría considerarse la gravedad.

Considerando un fluido Newtoniano isótropo, reordenando los términos y modelando las tensiones viscosas como función de la tasa local de deformación (tres de elongación y seis de cizalladura), se obtienen las ecuaciones de Navier-Stokes (3.5-3.7).

$$\rho\frac{Du}{Dt} = -\frac{\partial p}{\partial x} + div(\mu\ grad\ u) + S_{Mx} \tag{3.5}$$

$$\rho\frac{Dv}{Dt} = -\frac{\partial p}{\partial y} + div(\mu\ grad\ v) + S_{My} \tag{3.6}$$

$$\rho\frac{Dw}{Dt} = -\frac{\partial p}{\partial z} + div(\mu\ grad\ w) + S_{Mz} \tag{3.7}$$

3.2.3. Energía

Según la Primera Ley de la Termodinámica, la tasa de variación de energía en una partícula fluida equivale a la suma de tasa de adición de calor y el trabajo realizado por la misma. Por un lado, la tasa de trabajo aportado a la partícula fluida ejercido por una fuerza de superficie es igual al producto de fuerza y velocidad. Además hay que tener en cuenta el fenómeno de conducción de calor hasta la partícula fluida considerada, así como la tasa de adición de calor que este mismo fenómeno produce. La ecuación de la energía queda como se muestra en 3.8, donde el término S_E representa una fuente de energía por unidad de volumen y tiempo.

$$\rho\frac{DE}{Dt} = -div(pu) + div(k\ grad(T)) + S_E + \frac{\partial(u\tau_{xx})}{\partial x} + \frac{\partial(u\tau_{yx})}{\partial y} + \frac{\partial(u\tau_{zx})}{\partial z} + \frac{\partial(v\tau_{xy})}{\partial x} + \frac{\partial(v\tau_{yy})}{\partial y} + \frac{\partial(v\tau_{zy})}{\partial z} + \frac{\partial(w\tau_{xz})}{\partial x} + \frac{\partial(w\tau_{yz})}{\partial y} + \frac{\partial(w\tau_{zz})}{\partial z} \quad (3.8)$$

3.2.4. Ecuaciones De Estado

En las condiciones que se dan en las simulaciones del presente trabajo, el fluido siempre permanece en equilibrio termodinámico, pues no se dan fenómenos tales como la formación de ondas de choque intensas. Las ecuaciones de estado (3.9 y 3.10) relacionan variables termodinámicas entre sí por medio de las ecuaciones del gas perfecto

$$p = \rho RT \quad (3.9)$$

$$i = C_v T \quad (3.10)$$

donde R es la constante de gases ideales, C_v la relación de calores específicos a volumen constante e i la energía interna específica.

En flujo compresible estas ecuaciones proporcionan una conexión entre la ecuación de la energía y las ecuaciones de la continuidad o conservación de masa y de conservación de la cantidad de movimiento. Así, se puede dar la posibilidad de que existan variaciones en la densidad como consecuencia de las variaciones en la presión y en la temperatura. Esta conexión se rompe en el flujo incompresible, donde no se considera variación alguna en la densidad, de modo que el campo fluido puede ser resuelto considerando únicamente la conservación de la cantidad de movimiento y de masa.

3.3. MODELOS DE TURBULENCIA

Es imprescindible abordar el modelado de la turbulencia que se da en toda clase de flujos convencionales. Para ello se emplea la descomposición de Reynolds, mediante la cual todas las propiedades del flujo se dividen en dos términos: un valor medio más una componente fluctuante estadística.

Estas fluctuaciones turbulentas presentan un carácter tridimensional, además de un amplio rango de escalas de longitud, incluyendo estructuras de flujo rotacionales conocidas como eddies o remolinos. Como consecuencia de estas estructuras, los flujos turbulentos favorecen un intercambio efectivo de masa, cantidad de movimiento y calor entre las partículas fluidas del dominio, acelerando de esta manera los procesos de atomización, mezclado y evaporación de las gotas líquidas de combustible. La energía

cinética se va transmitiendo progresivamente desde los remolinos más grandes hacia los más pequeños en la denominada cascada de energía.

La velocidad y longitud característica de los remolinos grandes son del mismo orden de magnitud que la escala de velocidad y longitud del flujo medio, estando su comportamiento dominado, por tanto, por los efectos inerciales. La estructura de estos remolinos es altamente anisótropa, presentando fluctuaciones diferentes en función de la dirección y afectadas por las condiciones de contorno del problema.

No obstante, los remolinos pequeños presentan unas escalas de longitud del orden de 0.1-0.01 mm y unas frecuencias en torno a los 10 kHz, predominando los efectos viscosos. En estas escalas la energía asociada a los remolinos más pequeños se disipa en forma de energía térmica interna, así que la estructura de los remolinos más pequeños solamente depende de la tasa de disipación de la energía cinética turbulenta ε y de la viscosidad cinemática del fluido ν.

Estos remolinos turbulentos con amplio rango de escalas de longitud y temporales que interactúan de manera dinámica y compleja deben poder capturarse con los modelos de turbulencia y métodos numéricos desarrollados (ver figura 3.2).

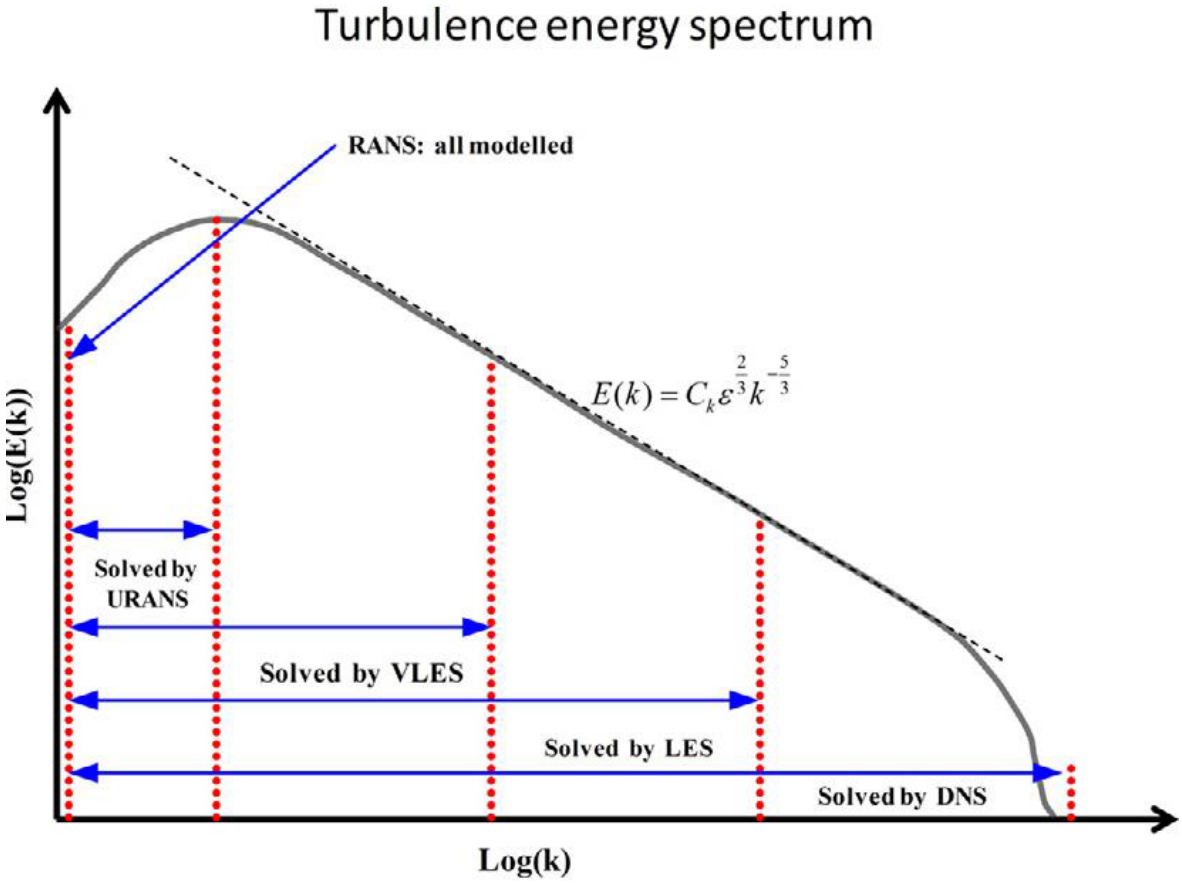


Figura 3.2: Espectro de energía turbulenta [47].

3.3.1. RANS

Los modelos de turbulencia RANS (Reynolds-Averaged Navier-Stokes) están centrados únicamente en capturar el flujo medio y los efectos que la turbulencia origina en las propiedades medias del flujo. Las ecuaciones de Navier-Stokes se promedian temporalmente antes de aplicar los métodos numéricos, descartando los detalles del flujo en fluctuaciones instantáneas y dando lugar a la aparición de términos extra como consecuencia de las interacciones entre fluctuaciones turbulentas, los esfuerzos de Reynolds: $-\rho\overline{u'^2}$, $-\rho\overline{v'^2}$ $-\rho\overline{w'^2}$, $-\rho\overline{u'v'}$, $-\rho\overline{u'w'}$ y $-\rho\overline{v'w'}$.

De esta manera, para ser capaz de calcular flujos turbulentos con las ecuaciones RANS es necesario desarrollar modelos de turbulencia capaces de predecir los esfuerzos de Reynolds y los términos de transporte escalar para cerrar el sistema. Los modelos de turbulencia más comunes se clasifican en base al número de ecuaciones de transporte adicionales que se necesitan resolver junto a las ecuaciones RANS y se enumeran en la tabla 3.1.

Tabla 3.1: Principales modelos de turbulencia empleados en simulaciones RANS

Nº ECUACIONES TRANSPORTE EXTRA	MODELO TURBULENCIA
0	Mixing length
1	Spalart-Allmaras
2	$k-\varepsilon$
2	$k-\omega$
2	Algebraic stress
7	Reynolds stress

Los recursos computacionales requeridos para llegar a la solución son moderados y asumibles por cualquier ordenador de la actualidad, razón por la cual ha sido la aproximación más empleada durante las últimas tres décadas.

Modelo K-epsilon. El modelo $k-\varepsilon$ permite tener en cuenta tanto los efectos del transporte de las propiedades turbulentas mediante convección y difusión. En él incorpora dos ecuaciones basadas en la energía cinética turbulenta k y la tasa de disipación ε de la misma. La formulación de las ecuaciones de transporte para estas variables dependen del modelo específico: standard, RNG, realizable.

La energía cinética instantánea $k(t)$ en un flujo turbulento es la suma de la energía cinética media y la energía cinética turbulenta (ecuación 3.11)

$$k(t) = \frac{1}{2}(U^2 + V^2 + W^2) + \frac{1}{2}(\overline{u'^2} + \overline{v'^2} + \overline{w'^2}) \tag{3.11}$$

Se puede obtener una ecuación para la energía cinética media K multiplicando cada una de las componentes de la ecuación de Reynolds por las correspondientes del vector de velocidad media [48], obteniendo la ecuación 3.12.

$$\underbrace{\frac{\partial(\rho K)}{\partial t}}_{a} + \underbrace{div(\rho K\vec{U})}_{b} = div\left(\underbrace{-P\vec{U}}_{c} + \underbrace{2\mu\vec{U}S_{ij}}_{d} - \underbrace{\rho\vec{U}\overline{u_i'u_j'}}_{e}\right) - \underbrace{2\mu S_{ij}\cdot S_{ij}}_{f} + \underbrace{\rho\overline{u_i'u_j'}\cdot S_{ij}}_{g} \tag{3.12}$$

a Tasa de evolución de la energía cinética media.

b Transporte de K por convección.

c Transporte de K por diferencias de presión.

d Transporte de K por esfuerzos viscosos.

e Transporte de K por esfuerzos de Reynolds.

f Tasa de disipación viscosa de K.

g Tasa de destrucción de K debido al trabajo de deformación realizado por los esfuerzos de Reynolds.

Resulta sencillo demostrar que en flujos con alto Reynolds los términos turbulentos (e) y (g) son siempre varios ordenes de magnitud mayores que sus correspondientes términos viscosos (d) y (f).

Operando de manera análoga con la ecuación instantánea de Navier-Stokes, y multiplicando cada componente fluctuante de la velocidad por su correspondiente ecuación, se obtiene una ecuación para la energía cinética turbulenta k [49] muy similar a la anterior:

$$\underbrace{\frac{\partial(\rho k)}{\partial t}}_{a} + \underbrace{div(\rho k\vec{U})}_{b} = div\left(\underbrace{-\overline{p'\vec{u}'}}_{c} + \underbrace{2\mu\overline{\vec{u}'s_{ij}'}}_{d} - \underbrace{\rho\overline{u_i'\cdot u_i'u_j'}}_{e}\right) - \underbrace{2\mu\overline{s_{ij}'\cdot s_{ij}'}}_{f} + \underbrace{\rho\overline{u_i'u_j'}\cdot S_{ij}}_{g} \tag{3.13}$$

La presencia de los términos fluctuantes en el lado derecho de la ecuación establece que las interacciones turbulentas son las principales causantes de los cambios que se producen en la energía cinética turbulenta. El término g es igual en ambas ecuaciones, pero de signo opuesto, de ahí la conversión de energía cinética media en energía cinética turbulenta. El término de disipación viscosa (f), origina una contribución negativa debido a la suma de los cuadrados de las tasas de deformación fluctuantes s_{ij}' (ecuación 3.14). Esta disipación de la energía cinética turbulenta ε ocurre porque los torbellinos más pequeños realizan trabajo contra los esfuerzos viscosos, y es un término totalmente necesario en el estudio de la turbulencia que no debe despreciarse.

$$-2\mu\overline{s'_{ij}\cdot s'_{ij}} = -2\mu\left(\overline{{s'_{11}}^2} + \overline{{s'_{11}}^2} + \overline{{s'_{11}}^2} + \overline{{s'_{11}}^2} + \overline{{s'_{11}}^2} + \overline{{s'_{11}}^2}\right) \tag{3.14}$$

En [50] se desarrolló una ecuación de transporte exacta similar para la tasa de disipación viscosa, pero debido a las innumerables incógnitas y términos difíciles de medir, el modelo $k-\varepsilon$ estándar emplea en su defecto las ecuaciones de transporte por [51] según las cuales la tasa de cambio de k o ε más el transporte de dicha magnitud por efecto de la convección es igual al transporte debido a difusión más la diferencia entre ratio de producción y ratio de destrucción (ecuaciones 3.15 y 3.16). La viscosidad turbulenta se define mediante análisis dimensional como $\mu_t = \rho C_\mu \frac{\varepsilon^2}{k}$.

$$\frac{\partial(\rho k)}{\partial t} + div(\rho k\vec{U}) = div\left(\frac{\mu_t}{\sigma_k} grad(k)\right) + 2\mu_t S_{ij}\cdot S_{ij} - \rho\varepsilon \tag{3.15}$$

$$\frac{\partial(\rho \varepsilon)}{\partial t} + div(\rho \varepsilon\vec{U}) = div\left(\frac{\mu_t}{\sigma_\varepsilon} grad(\varepsilon)\right) + C_{1\varepsilon}\frac{\varepsilon}{k}2\mu_t S_{ij}\cdot S_{ij} - C_{2\varepsilon}\rho\frac{\varepsilon^2}{k} \tag{3.16}$$

Las ecuaciones contienen cinco constantes C_μ, σ_k, σ_ε, $C_{1\varepsilon}$ y $C_{2\varepsilon}$ que deben ser ajustadas mediante experimentos para un amplio rango de flujos turbulentos. Los términos correspondientes al transporte turbulento se representan mediante la difusión del gradiente de la magnitud transportada, donde los números de Prandtl σ_k y σ_ε conectan las difusividades de k y ε con la viscosidad turbulenta μ_t. Además, los términos de producción y destrucción de energía cinética turbulenta se encuentran fuertemente acoplados, presentando un valor grande de tasa de disipación ε allí donde la producción de k es elevada.

Debido a la naturaleza elíptica de las ecuaciones de transporte, se deben imponer unas condiciones de contorno determinadas en la entrada, salida y paredes del dominio. Ante la ausencia de estudios de sensibilidad de las condiciones de contorno en los problemas ingenieriles específicos se pueden tomar aproximaciones para establecer distribuciones realistas a la entrada de k y ε en flujos internos a partir de la intensidad turbulenta I y longitud característica L, mediante las siguientes relaciones:

$$k = \frac{2}{3}(U_{ref}I)^2 \quad \varepsilon = C_\mu^{3/4}\frac{k^{3/2}}{l} \quad l = 0.07L \tag{3.17}$$

La principal ventaja de este modelo es que no hay necesidad de ajustar las constantes, y su éxito está demostrado y validado para multitud de flujos turbulentos. Ocurre incluso en aquellos con zonas de recirculación [52] como el del presente trabajo, donde la predicción del transporte turbulento es imprescindible para reproducir correctamente el proceso de combustión.

No obstante, al estar basado en la hipótesis de viscosidad turbulenta isótropa de Boussinesq, este modelo puede presentar algunas dificultades frente a flujos con alta

componente de swirl, o frente a aquellos que presenten una deformación rápida como sucede en capas límites con un elevado grado de curvatura o en secciones divergentes.

El caso estudiado reúne varias de estas características: capa límite curvada en el swirler, sección divergente en el venturi y algunas zonas de recirculación en la entrada a la cámara de combustión. En función del grado de impacto de cada zona, el modelo será más o menos preciso; sin embargo el uso del mismo en este trabajo se hará para obtener una solución inicial realista, es decir, como punto de partida para modelos más complejos.

Modelo De Esfuerzos De Reynolds. El RSTM (Reynolds Stress Turbulence Model) es un modelo clásico de turbulencia más complejo, que resuelve una ecuación extra de transporte para cada una de las seis componentes independientes del tensor de Reynolds. De esta manera, el modelo es capaz de predecir de forma más precisa los flujos con campos de deformaciones complejos en los que el modelo $k-\varepsilon$ obtiene peores resultados. Definiendo los esfuerzos de Reynolds $R_{ij} = -\tau_{ij}/\rho = \overline{u_i' u_j'}$, su tasa de cambio y su transporte debido a convección dependen de la tasa de producción, el transporte por difusión, la tasa de disipación, el transporte por presiones turbulentas y el debido a estructuras rotacionales (ecuación 3.18).

$$\frac{DR_{ij}}{Dt} = \frac{\partial R_{ij}}{\partial t} + C_{ij} = P_{ij} + D_{ij} - \varepsilon_{ij} + \Pi_{ij} + \Omega_{ij} \tag{3.18}$$

Los términos correspondientes a la convección, producción y rotación se definen como:

$$C_{ij} = \frac{\partial(\rho U_k \overline{u_i' u_j'})}{\partial x_k} = div(\rho \overline{u_i' u_j'} \vec{U}) \tag{3.19}$$

$$P_{ij} = -(R_{im}\frac{U_j}{x_m} + R_{jm}\frac{U_i}{x_m}) \tag{3.20}$$

$$\Omega_{ij} = -2\omega_k(\overline{u_j' u_m'} e_{ikm} + \overline{u_i' u_m'} e_{jkm}) \tag{3.21}$$

mientras que la difusión, tasa de disipación y correlación presión-deformación deben ser modelados. [53, 54]

El término difusivo se modela bajo la suposición de que la variación del transporte de los esfuerzos de Reynolds por difusión es proporcional a los propios esfuerzos (ecuación 3.22), siendo $\upsilon_t = C_\mu \frac{\varepsilon^2}{k}$, $C_\mu = 0.09$ y $\sigma_k = 1$.

$$D_{ij} = \frac{\partial}{\partial x_m}\left(\frac{\upsilon_t}{\sigma_k}\frac{\partial R_{ij}}{\partial x_m}\right) = div\left(\frac{\upsilon_t}{\sigma_k} grad(R_{ij})\right) \tag{3.22}$$

La tasa de disipación se modela asumiendo el comportamiento isótropo de los remolinos pequeños disipativos (ecuación 3.23), los cuales afectan únicamente a las componentes normales de los esfuerzos de Reynolds, teniendo en cuenta que la tasa de disipación de energía cinética turbulenta es $\varepsilon = 2\upsilon \overline{s_{ij}' s_{ij}'}$.

$$\varepsilon_{ij} = \frac{2}{3}\varepsilon\delta_{ij} \tag{3.23}$$

Finalmente, el término de interacciones presión-deformación es el más difícil de modelar con precisión, ya que el efecto que causan se debe a dos fenómenos físicos independientes: un proceso lento que reduce la anisotropía de los remolinos turbulentos por sus interacciones mutuas, y un proceso rápido como consecuencia de las interacciones entre las fluctuaciones turbulentas y la deformación del flujo medio. La manera más simple de representar el proceso lento considera que la transición hacia las condiciones isótropas es proporcional al grado de anisotropía a_{ij} de los esfuerzos de Reynolds dividido por un tiempo característico k/ε. Por el contrario, el efecto rápido únicamente tiene en cuenta el proceso de producción causante de la generación de anisotropía.

El efecto conjunto de ambos procesos es el de redistribuir la energía entre los esfuerzos de Reynolds normales (R_{11}, R_{22} y R_{33}) en busca de un comportamiento más isótropo, tratando de reducir los esfuerzos tangenciales (R_{12}, R_{13} y R_{23}). La fórmula que representa este término se muestra en la ecuación 3.24, siendo las constantes $C_1 = 1.8$, $C_2 = 0.6$ y $k = \frac{1}{2}(R_{11} + R_{22} + R_{33})$.

$$\Pi_{ij} = -C_1\frac{\varepsilon}{k}\left(R_{ij} - \frac{2}{3}k\delta_{ij}\right) - C_2\left(P_{ij} - \frac{2}{3}P\delta_{ij}\right) \tag{3.24}$$

Las seis ecuaciones para el transporte de los esfuerzos de Reynolds se resuelven de manera conjunta, modelando además la tasa de disipación de la energía cinética turbulenta de un modo prácticamente idéntico al modelo $k - \varepsilon$ (ecuación 3.25).

$$\frac{D\varepsilon}{Dt} = div\left(\frac{\upsilon_t}{\sigma_\varepsilon}grad(\varepsilon)\right) + C_{1\varepsilon}\frac{\varepsilon}{k}2\upsilon_t S_{ij}\cdot S_{ij} - C_{2\varepsilon}\frac{\varepsilon^2}{k} \tag{3.25}$$

De igual manera que en el método anterior, se requiere la definición de las condiciones de contorno de una manera específica para satisfacer la naturaleza elíptica del sistema de ecuaciones diferenciales. Por tanto, se debe establecer una distribución de R_{ij} y ε a la entrada del dominio, imponer las variaciones nulas $\partial R_{ij}/\partial n = 0$ y $\partial\varepsilon/\partial n = 0$ a la salida, y definir las funciones de pared declarando el comportamiento que toma R_{ij} en función de k en las zonas cercanas a éstas.

En la actualidad el modelo RSTM no se encuentra tan validado, y por a su elevado coste computacional la aplicación es escasa en el ámbito industrial. Además, puede sufrir problemas para alcanzar la convergencia debido al acoplamiento de la velocidad media y de los esfuerzos turbulentos mediante términos fuente. Sin embargo su gran ventaja frente al $k - \varepsilon$ es la alta precisión a la hora de calcular propiedades del flujo medio y de todos los esfuerzos de Reynolds en flujos complejos (incluyendo chorros contra pared, canales axisimétricos o flujos curvados) que de otra manera no podrían ser predichos con detalle.

3.3.2. URANS

La aproximación Unsteady Reynolds-Averaged Navier Stokes (URANS) se caracteriza por ser una versión más compleja de RANS. En ella se obtiene una solución temporal para problemas tridimensionales no estacionarios, con o sin tratamiento especial para inestabilidades del flujo [55].Las ecuaciones de URANS son similares a los modelos RANS pero incluyendo un término no estacionario adicional. El aspecto más importante es la selección correcta del modelo de turbulencia adecuado y el esquema de discretización (que no sea excesivamente disipativo). URANS actúa bien para calles de vórtices de von Kármán [56], cubos montados sobre superficies [57] y flujo alrededor de coches [58] entre otras aplicaciones.

3.3.3. LES

El Large Eddy Simulation (LES) se trata de un paso intermedio a la hora de obtener y calcular las interacciones de las fluctuaciones turbulentas. En este caso únicamente se modelan los remolinos más pequeños, que presentan un comportamiento isótropo e independiente del problema, mientras que el comportamiento de los remolinos grandes sí que es calculado directamente. Éstos son los encargados de interactuar y extraer energía del flujo medio, se caracterizan por una elevada anisotropía y una fuerte dependencia tanto de la geometría del dominio del problema como de las condiciones de contorno impuestas.

Esta separación entre los remolinos por sus tamaños se consigue definiendo una longitud de corte (cutoff width), que permite realizar un filtrado espacial de las ecuaciones de Navier-Stokes no estacionarias. Para modelar los efectos de los remolinos pequeños sobre el flujo resuelto se utilizan los modelos SGS (sub-grid scale).

Esta operación de filtrado espacial (ecuación 3.26) se lleva a cabo mediante una función de filtro $G(\vec{x}, \vec{x'}, \Delta)$, con longitud de corte Δ.

$$\overline{\phi}(\vec{x}, t) = \iiint_{-\infty}^{\infty} G(\vec{x}, \vec{x'}, \Delta)\ \phi(\vec{x'}, t) dx'_1 dx'_2 dx'_3 \tag{3.26}$$

Aunque se han investigado multitud de funciones de filtro, la más empleada en volúmenes finitos para simulaciones LES es el filtro de caja (ecuación 3.27). En principio se puede seleccionar cualquier valor de corte, pero a nivel práctico resulta inútil emplear longitudes inferiores al tamaño de celda de la malla utilizada (de ahí el nombre subgrid), ya que debido a que solamente se almacena un valor nodal por celda de cada variable del flujo, se perderían igualmente todos los detalles calculados. Por tanto, conviene tomar la longitud de corte del mismo orden de magnitud que tamaño de malla $\Delta = \sqrt[3]{\Delta x \Delta y \Delta z}$.

$$G(\vec{x},\vec{x'},\Delta)=\begin{cases}1/\Delta^3 & si \quad |\vec{x}-\vec{x'}|\leq \Delta/2\\ 0 & si \quad |\vec{x}-\vec{x'}|\geq \Delta/2\end{cases} \tag{3.27}$$

Tomando un filtro uniforme independiente de la posición a lo largo de todo el dominio computacional se obtiene la Ecuación de la Continuidad y las Ecuaciones de Cantidad de Movimiento para LES (ecuaciones 3.28-3.31).

$$\frac{\partial \rho}{\partial t}+div(\rho\overline{\vec{u}})=0 \tag{3.28}$$

$$\frac{\partial(\rho\overline{u})}{\partial t}+div(\rho\overline{u}\overline{\vec{u}})=\frac{\partial \overline{p}}{\partial x}+\mu\ div(grad(\overline{u}))-(div(\rho\overline{u\vec{u}})-div(\rho\overline{u}\overline{\vec{u}})) \tag{3.29}$$

$$\frac{\partial(\rho\overline{v})}{\partial t}+div(\rho\overline{v}\overline{\vec{u}})=\frac{\partial \overline{p}}{\partial y}+\mu\ div(grad(\overline{v}))-(div(\rho\overline{v\vec{u}})-div(\rho\overline{v}\overline{\vec{u}})) \tag{3.30}$$

$$\underbrace{\frac{\partial(\rho\overline{w})}{\partial t}}_{a}+\underbrace{div(\rho\overline{w}\overline{\vec{u}})}_{b}=\underbrace{\frac{\partial \overline{p}}{\partial z}}_{c}+\underbrace{\mu\ div(grad(\overline{w}))}_{d}-\underbrace{(div(\rho\overline{w\vec{u}})-div(\rho\overline{w}\overline{\vec{u}}))}_{e} \tag{3.31}$$

a Tasas de cambio de las componentes de la cantidad de movimiento filtradas.

b Flujos convectivos.

c Gradientes en las tres direcciones del campo de presión filtrado.

d Flujos difusivos.

e Aparecen a causa del filtrado.

Los términos (e) se pueden considerar como la divergencia de un conjunto de esfuerzos τ_{ij}, definidos como

$$div(\rho\overline{u_i\vec{u}}-\rho\overline{u_i}\overline{\vec{u}})=\frac{\partial(\overline{u_iu}-\overline{u_i}\ \overline{u})}{\partial x}+\frac{\partial(\overline{u_iv}-\overline{u_i}\ \overline{v})}{\partial y}+\frac{\partial(\overline{u_iw}-\overline{u_i}\ \overline{w})}{\partial z}=\frac{\partial \tau_{ij}}{\partial x_j}$$

donde $\tau_{ij}=\rho\overline{u_i\vec{u}}-\rho\overline{u_i}\overline{\vec{u}}=\rho\overline{u_iu_j}-\rho\overline{u_i}\overline{u_j}$. Se les llama esfuerzos SGS porque son originados por el transporte convectivo e interacciones entre los remolinos modelados, es decir, los que tienen una escala de longitud inferior al tamaño de malla. Además se ven afectados por diversas contribuciones, cuya naturaleza se determina descomponiendo las variables del flujo $\phi(\vec{x},t)$ como suma de la función filtrada $\overline{\phi}(\vec{x},t)$ y las variaciones espaciales pequeñas $\phi'(\vec{x},t)$.

Aplicando la descomposición mencionada, los esfuerzos SGS quedan definidos mediante:

$$\tau_{ij}=\underbrace{\rho\overline{\overline{u_i}\ \overline{u_j}}-\rho\overline{u_i}\ \overline{u_j}}_{a}+\underbrace{\rho\overline{\overline{u_i}u_j'}+\rho\overline{u_i'\overline{u_j}}}_{b}+\underbrace{\rho\overline{u_i'u_j'}}_{c} \tag{3.32}$$

El término (a) corresponde a los esfuerzos de Leonard, L_{ij}, debidos únicamente a fenómenos acontecidos en la escala resuelta. El término (b) corresponde a los esfuerzos cruzados C_{ij}, originados fruto de las interacciones entre los remolinos SGS modelados y el flujo resuelto, y cuyas expresiones se pueden encontrarse en [59]. Por último, se tienen

los esfuerzos de Reynolds de LES (c), causados como consecuencia de la transferencia de cantidad de movimiento convectiva provocada por las interacciones de los remolinos SGS. Igual que sucede con los esfuerzos de Reynolds en las ecuaciones RANS, estos esfuerzos SGS deben ser modelados, para lo cual existen actualmente multitud modelos en la bibliografía.

Smagorinsky-Lilly SGS Model. El modelo SGS [60] está basado en la suposición de la naturaleza isótropa de los remolinos turbulentos más pequeños siguiendo la hipótesis de Boussinesq [61]. Por lo tanto, el modelo considera que las tensiones SGS locales o las tensiones de Reynolds LES R_{ij} (ecuación 3.33) son proporcionales a la tasa de deformación local de los flujos resueltos.

$$R_{ij} = -2\mu_{SGS}\overline{S_{ij}} + \frac{1}{3}R_{ii}\delta_{ij} = -\mu_{SGS}\left(\frac{\partial\overline{u_i}}{\partial x_j} + \frac{\partial\overline{u_j}}{\partial x_i}\right) + \frac{1}{3}R_{ii}\delta_{ij} \tag{3.33}$$

El término μ_{SGS} (ecuación 3.34) representa la viscosidad dinámica SGS, mientras que $\frac{1}{3}R_{ii}\delta_{ij}$ asegura que la suma de los esfuerzos normales SGS modelados sea igual a la energía cinética de los remolinos. Este modelo asume que la viscosidad cinemática puede ser descrita por una escala de longitud (longitud Δ de corte) y una de velocidad ($\Delta * |\overline{S}|$).

$$\mu_{SGS} = \rho(C_{SGS}\Delta)^2|\overline{S}| \tag{3.34}$$

La tasa de deformación promediada se obtiene mediante

$$|\overline{S}| = \sqrt{2\overline{S}_{ij}\overline{S}_{ij}} \tag{3.35}$$

La diferencia principal entre el modelo Smagorinsky-Lilly y el Dynamic Smagorinsky reside en que la constante C_{SGS} pasa a ser variable y dependiente del tiempo y espacio. En la práctica requiere estabilización, que se suele obtener promediando en una dirección homogénea. En los casos donde no es posible, se utiliza un promediado local en su lugar. [62]

Finalmente, se deben imponer las condiciones iniciales y de contorno, similares a las empleadas en las simulaciones RANS, pero con ciertas peculiaridades. Al tratarse de simulaciones no estacionarias la solución final alcanzada es independiente de las condiciones iniciales, influyendo éstas únicamente en el camino hasta la convergencia y en el tiempo requerido.

Para saber cuánto se debe refinar una malla en la zona cercana a una pared se utiliza el parámetro adimensional y^+, que se calcula mediante la fórmula 3.36 donde u_τ es la velocidad cortante o de fricción, y la distancia a la pared y ν la viscosidad cinemática. La ley de pared de von Karman, dicta que dependiendo del valor de este

parámetro conviene utilizar una aproximación concreta para obtener la mayor precisión posible en el cálculo de la velocidad (figura 3.3): la capa viscosa, la capa buffer y la capa logarítmica.

$$y^+ = \frac{yu_\tau}{\nu} \tag{3.36}$$

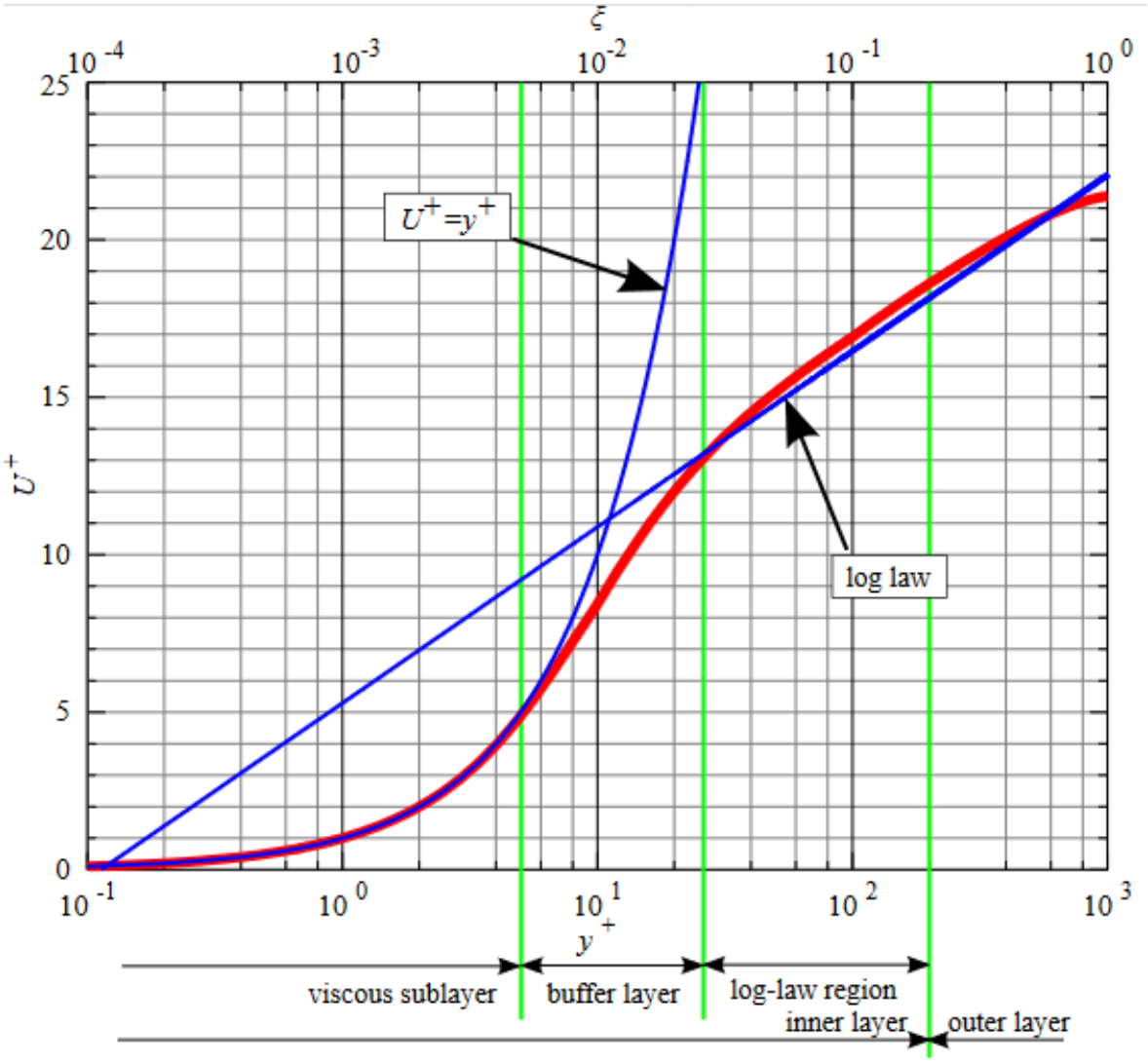


Figura 3.3: Ley de pared

En las zonas cercanas a la pared se aplica la condición de no deslizamiento, integrando las ecuaciones de Navier-Stokes filtradas y requiriendo, por tanto, un refinamiento que asegure el cumplimiento de $y^+ \leq 1$ para todas las superficies sólidas del dominio.

Las condiciones de contorno definidas a la entrada del dominio son determinantes sobre la solución final alcanzada. La manera más sencilla de proceder consiste en especificar distribuciones de velocidad media medidas o calculadas previamente con otro

modelo, y sobreponer perturbaciones gaussianas aleatorias con una intensidad turbulenta realista. No obstante, debido a los pequeños pasos temporales frecuentes en las simulaciones, las distorsiones de las propiedades turbulentas pueden requerir elevadas cantidades de tiempo para asentarse antes de que el flujo medio alcance el equilibrio. Como alternativa, se suele representar el flujo a la entrada mediante un modelo RANS, simulando un caso no estacionario con el modelo de esfuerzos de Reynolds para obtener una estimación de todos ellos en el plano de entrada.

Por su parte, las condiciones de contorno a la salida del dominio son menos problemáticas, empleando la conocida condición de gradiente nulo para el flujo medio, mientras que las propiedades fluctuantes son extrapoladas mediante:

$$\frac{\partial \phi}{\partial t} + \overline{u_n}\frac{\partial \phi}{\partial n} = 0 \tag{3.37}$$

La demanda de recursos computacionales en términos de almacenamiento y volumen de operaciones es enorme, ya que se deben resolver las ecuaciones del flujo no estacionarias. A pesar de ello, los avances en tecnología de computación están permitiendo progresivamente la aplicación de estas técnicas LES para el estudio de problemas con geometrías complejas. Se trata de una gran inversión de tiempo y recursos, pero tiene en cuenta los detalles estadísticos relacionados con las fluctuaciones turbulentas presentes en flujos con altas componentes de swirl.

3.3.4. DNS

En el Direct Numerical Simulation (DNS) se calcula el flujo medio y todas las fluctuaciones turbulentas de la velocidad. Para ello, se resuelven las ecuaciones de Navier-Stokes no estacionarias en mallas lo suficientemente finas (del orden de la escala de Kolmogorov, donde tiene lugar la disipación de la energía) y con pasos temporales extremadamente pequeños del orden del periodo de las fluctuaciones turbulentas más rápidas. Estos cálculos son altamente costosos en términos de recursos computacionales, motivo por el cual estos métodos todavía no se emplean en problemas industriales.

3.4. MÉTODOS DE DISCRETIZACIÓN

A pesar de ser ampliamente conocidas y utilizadas, las ecuaciones que se deben utilizar para resolver los problemas de CFD (detalladas en la sección 3.2) no tienen demostrada hasta el momento una solución analítica, por lo que es necesario utilizar métodos numéricos. Aplicando el método apropiado bajo las condiciones correctas se obtendrá una solución aproximada perfectamente válida, es decir, con un error asumible.

El primer paso para enfocar un problema desde el punto de vista numérico es la discretización, donde se intercambia el dominio contínuo por un dominio discreto. Existen tres métodos más conocidos [48]:

Método de diferencias finitas. Un problema en derivadas parciales puede aproximarse mediante este método en formulación fuerte:

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0 \Rightarrow \frac{du}{dt} + \frac{u_i - u_{i-1}}{\Delta x} = 0 \tag{3.38}$$

donde u es la variable dependiente y x es la posición temporal. La ecuación 3.38 ha sido convertida a diferencias finitas tras discretizar en múltiples nodos de cálculo. Cada nodo i se encuentra en una posición x_i, con las posiciones separadas una distancia Δx. La derivada ha sido sustituida por diferencias finitas adelantadas de primer orden (método upwind de primer orden).

La resolución de los sistemas en derivadas parciales mediante diferencias finitas tiene ciertos problemas al intentar adaptarse a geometrías complejas. Además, la solución ha de ser tan suave como exija el sistema de ecuaciones. Utilizando la formulación débil (con forma integral), es posible adaptarse con más facilidad a geometrías complejas y reducir las exigencias de suavidad de la solución:

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0 \Rightarrow \int_0^L \frac{\partial u}{\partial t} \cdot w(x)dx + \int_0^L \frac{\partial u}{\partial x} \cdot w(x)dx = 0 \tag{3.39}$$

donde w(x) es la función prueba, función test o función de peso.

Método de elementos finitos. Si se aproxima la solución anterior como:

$$u(x,t) = \sum_i U_i(t) \cdot N_i(x) \tag{3.40}$$

donde las funciones $N_i(x)$ son la base del desarrollo en serie de u, y se usa $w(x) = N_i(x)$, se obtiene el método de los elementos finitos. En él se usan funciones base contínuas. En general, suelen ser polinomios (o polinomios definidos a trozos). Cada N_i es distinto de 0 sólo para cierta zona del dominio.

Métodos espectrales. Si el dominio en el que $Ni(x)$ es distinto de 0 (soporte de la base) es todo el intervalo, se tiene un método espectral. Éstos tienen convergencia espectral: convergen de forma exponencial si la solución es suave, pero lamentablemente, no capturan bien las discontinuidades.

Método de volúmenes finitos. Si se discretiza el dominio en n volúmenes de control y usamos n funciones para la base tales que N_i vale 1 sólo en el volumen de control i y 0 fuera de él, se obtiene el método de volúmenes finitos. Este método es extremadamente interesante para resolver las ecuaciones de Navier-Stokes (o una aproximación de las mismas) y puede permitir capturar bien discontinuidades

como ondas de choque. Por ello es el más utilizado en CFD y en el presente trabajo.

Para la ecuación de continuidad se aplica el teorema de la divergencia de Gauss, transformando la integral de volumen a una integral de área:

$$\frac{\partial \rho}{\partial t} = -\vec{\nabla} \cdot (\rho \vec{u}) \Rightarrow \frac{d\bar{\rho}_i}{dt} \cdot V_i = -\int_{V_i} \vec{\nabla} \cdot (\rho \vec{u}) dV = -\int_{A_i} (\rho \vec{u}) \cdot \vec{n} \; dA \qquad (3.41)$$

donde V_i es el volumen finito i del volumen de control, A es la sección y $\bar{\rho}_i$ es el valor medio de la densidad de la celda i.

Con volúmenes finitos, el problema reduce el orden de sus derivadas parciales. Gran parte de la complejidad estará en obtener los flujos a través de las superficies de las celdas. La solución es necesariamente conservativa: el flujo que sale de una celda entra en otra.

3.5. ETAPAS DE LA SIMULACIÓN

En este apartado se tratarán por orden las tres etapas existentes y bien diferenciadas que tienen lugar en la simulación CFD.

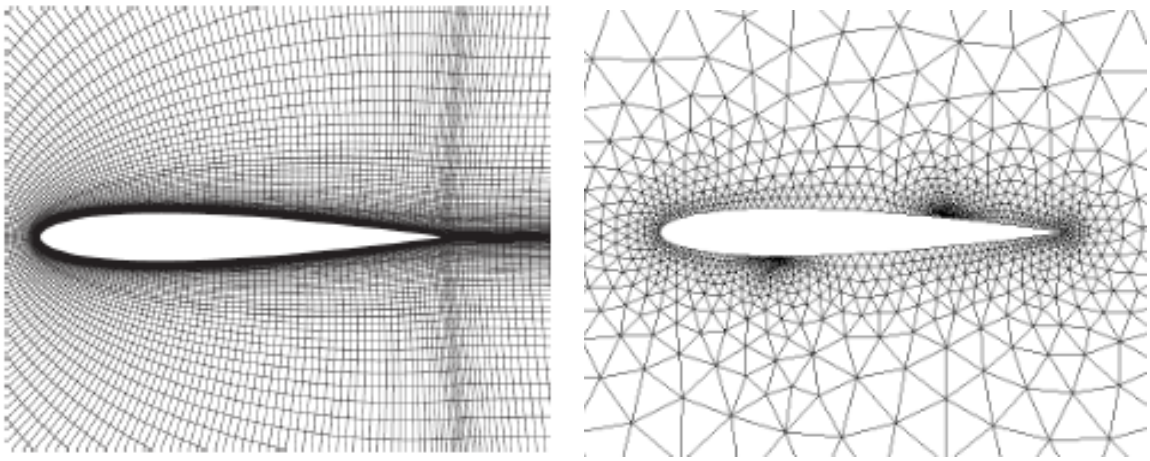
3.5.1. Pre-procesado

Para que las ecuaciones puedan ser aplicadas en el dominio computacional, debe generarse un mallado, cuyas celdas o elementos representarán los volúmenes de control que componen la geometría del problema. Es necesario poner especial atención en este punto, pues de la calidad de la malla dependerá en gran medida la precisión de la simulación. La solución del problema se define en los nodos dentro de cada celda, con lo cual la cantidad de éstas influye en la validez de resultados, pero también en el tiempo necesario para la resolución. A mayor número de celdas, tanto la precisión como el coste computacional aumentan.

Con carácter general, existen dos maneras de realizar el mallado:

- Malla estructurada (figura 3.4(a)). Consiste en emplear elementos hexaédricos a lo largo del dominio computacional de modo que se consiga un diseño más simple adaptado a la geometría del problema. No siempre es posible utilizar un mallado estructurado, puesto que en geometrías de gran complejidad resulta prácticamente imposible. El número de elementos adyacentes a un nodo o a una línea ha de conservarse, por la forma en que se localiza y representa gráficamente la distribución de la geometría. Por su parte, según [48] la gran ventaja radica en el hecho de que al disponerse los elementos de forma alineada con la dirección del fluido se consigue acelerar la ruta hasta la convergencia del solver.

- Malla no estructurada (figura 3.4(b)). Cuando resulta complicado realizar un mallado estructurado por la complejidad de la geometría, se opta por esta otra opción. Se disponen los elementos en forma de prisma rectangular, tetraédricos o poliédricos (entre otros), de manera que puedan ajustarse de una manera más eficiente a la geometría. No existen restricciones en cuanto al número de elementos adyacentes a un nodo o a una línea, como sí ocurría en el caso estructurado, y el tiempo necesario para realizar este mallado es inferior. En cambio, al no estar alineados los elementos con la dirección del fluido suele ser más complicado el camino hacia la convergencia del solver.
- Malla híbrida. En ocasiones, cuando se quiere estudiar fenómenos relacionados con la capa límite se disponen elementos estructurados en las proximidades de la pared y no estructurados en las zonas alejadas, dando lugar a lo que se conoce como una malla híbrida. De esta manera se combina lo mejor de cada tipo de malla, para disminuir el tiempo de mallado y a su vez ayudar a la convergencia del solver.



(a) Malla estructurada (b) Malla no estructurada

Figura 3.4: Mallado bidimensional de un perfil alar.

En la teoría, una malla enorme con elementos infinitamente pequeños sería ideal para unos resultados totalmente precisos, pero sería inabordable por su coste computacional. Por ello, independientemente del tipo de malla escogido, se debe asegurar que la cantidad de celdas (así como el tamaño de las mismas) representa de forma suficientemente fidedigna la geometría del problema, a la vez que no supone un tiempo de cálculo exagerado. En este punto se pone de manifiesto la importancia del análisis de independencia de malla, cuyo objetivo es garantizar ambas cosas.

Este proceso es simple teóricamente: se estudia el mismo caso varias veces, modificando únicamente el refinamiento de la malla hasta que los resultados no sufren una

alteración exagerada. A partir de ese punto en el que el aumento de tiempo de cálculo no compensa la diferencia de precisión que se obtiene, se dice que la convergencia de malla ha sido alcanzada. La malla puede modificarse mediante el tamaño base de las celdas, los refinamientos locales en zonas de interés o incluso por algoritmos especializados como el AMR (que se tratará más adelante) llevados a cabo automáticamente por el software, bajo unos criterios definidos por el usuario.

A continuación se requiere la definición física del problema, un proceso que abarca desde la elección de las propiedades del fluido (presión, velocidad, temperatura, etc.), pasando por los fenómenos físicos que han de modelarse (turbulencia, transferencia de calor, reacciones químicas, etc.), hasta la definición de las condiciones de contorno del problema.

3.5.2. Solver

En el solver se resuelve la matriz o sistema de ecuaciones algebraicas que se ha obtenido tras la discretización de las ecuaciones integrales y de la geometría y físicas del problema. Esto se hace a través de un proceso iterativo, hasta la convergencia, o sea, cuando la variación de los resultados entre dos iteraciones consecutivas queda dentro de los límites en un criterio establecido por el usuario.

En el mercado hay infinidad de software, tanto de libre uso como de licencia comercial, dedicado al cálculo CFD. En la sección 4.2 se exponen sendos programas utilizados durante este trabajo para las simulaciones.

Algoritmo SIMPLE. A la hora de resolver simulaciones estacionarias, el método más utilizado es el Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) [63] mostrado en la figura 3.5. No está limitado por el número de Courant (CFL), ya que es estable en $CFL > 1$ al contrario que ocurre con el algoritmo PISO. El número de Courant es una condición para la convergencia de algoritmos numéricos como el descrito aquí, y se define como dicta la ecuación 3.42 donde u es la velocidad, Δt es el intervalo temporal y Δx es el intervalo de espacio.

$$C = \frac{u\Delta t}{\Delta x} \tag{3.42}$$

Partiendo de la matriz con las ecuaciones de momento, se obtiene la presión al resolver la ecuación de momento (solve momentum). Entonces se aplica una corrección en la presión (pressure corrector). Esta presión obtenida sirve para corregir las ecuaciones de momento y flujo (momentum and flux correctors), y se añaden a la matriz los nuevos resultados. Este bucle (SIMPLE loop) puede ser repetido las veces que sea necesario hasta alcanzar la convergencia. Una vez terminado se podrán resolver el resto de ecuaciones de transporte en serie.

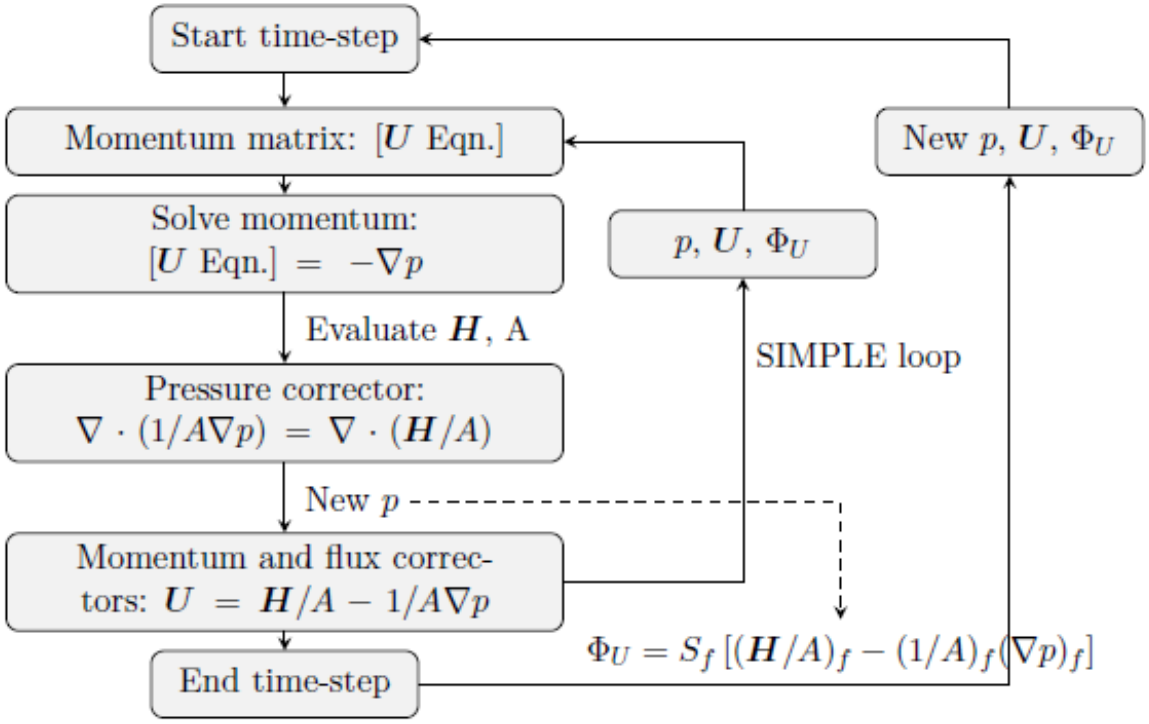


Figura 3.5: Esquema del algoritmo SIMPLE para casos incompresibles. [64]

Algoritmo PISO. Para resolver el acoplamiento entre presión y velocidad en simulaciones transitorias se utiliza el método Pressure Implicit with Splitting of Operators (PISO) [65] mostrado en la figura 3.6. Este será el utilizado en el presente trabajo por la naturaleza transitoria del problema a resolver, además de ser más rápido porque evita el amortiguamiento (under-relaxation) en la convergencia que provoca reconstruir la matriz del sistema repetidas veces en cada iteración.

Este algoritmo comienza resolviendo la ecuación de momento implícita, como paso de predicción. A continuación se deriva y se resuelve la corrección de la presión. Con la nueva presión se corrigen las ecuaciones de momento y flujo, pudiendo volver a repetir este bucle de resolución (PISO loop) tantas veces como sea necesario para obtener la precisión deseada por el criterio de convergencia. A diferencia del algoritmo SIMPLE, el bucle no vuelve al momento de construcción de la matriz del sistema, sino a la corrección de la presión. Finalmente, se resuelven el resto de ecuaciones de transporte en serie.

Para problemas de flujo compresible, el resto de las ecuaciones (continuidad, energía...) se pueden añadir tras el corrector de presión [66], sin embargo la mayoría lo hace justo antes de ensamblar la matriz de momento. [67]

En algunos solvers el usuario puede limitar las correcciones PISO indicando un número mínimo y máximo de las mismas. Si no ha alcanzado la convergencia tras

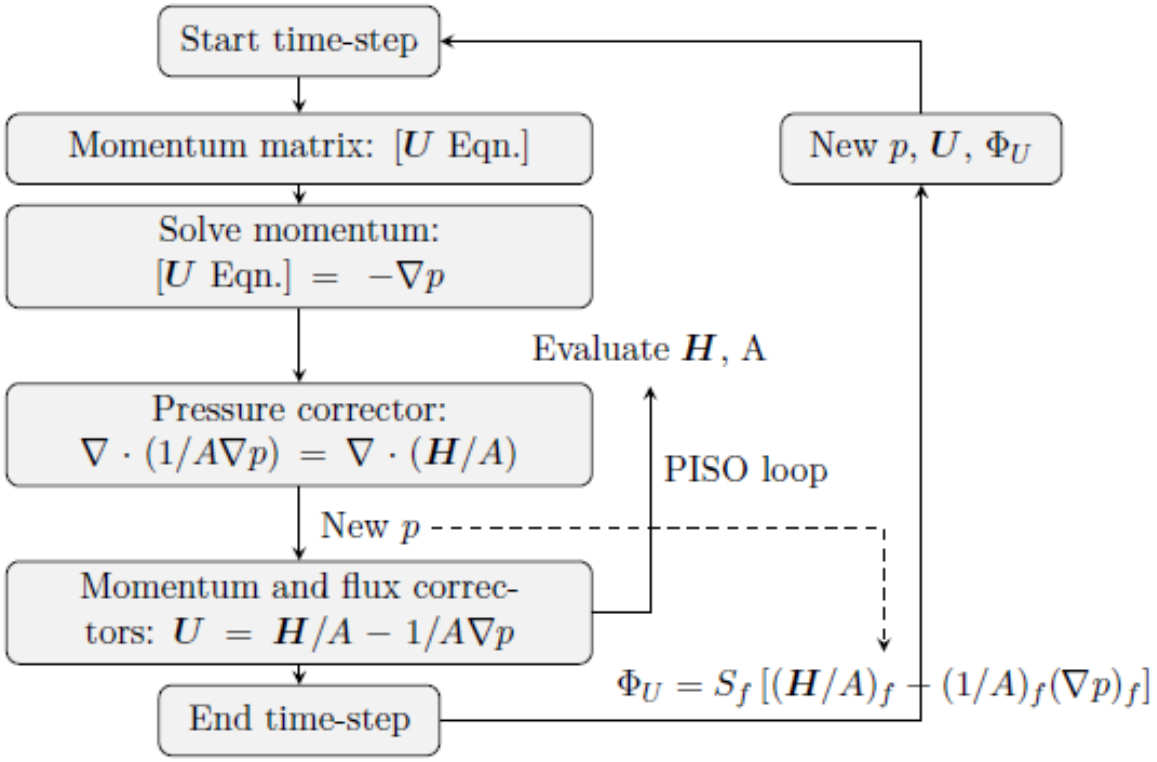


Figura 3.6: Esquema del algoritmo PISO para casos incompresibles. [64]

llegar a dicho número, se reducirá el siguiente paso temporal.

3.5.3. Post-procesado

Tras la simulación llega el último paso, donde se deben analizar los resultados. Para ello es imprescindible conocer las magnitudes representativas del problema en cuestión, y realizar gráficas que permitan obtener conclusiones sobre el estudio que se ha llevado a cabo. Existe una gran variedad de posibilidades para el tratamiento de los resultados, tanto desde el punto de vista gráfico (posibilidad de obtener resultados en secciones o líneas, realización de animaciones, etc.) como a la hora de gestionar los resultados (estos pueden ser exportados en múltiples formatos para su posterior tratamiento, se pueden definir nuevas variables físicas a partir de las que vienen por defecto, guardado de las trazas de las partículas de una fase líquida dentro de otra continua, etc.). En definitiva, el ingeniero dispone de una amplia gama de utilidades dentro del post-procesado para exprimir al máximo el valor de su simulación.

En este sector conviven múltiples programas de post-procesado. Algunos han sido concebidos por los mismos creadores de un solver concreto, incluidos en el paquete de software y adaptados para facilitar su uso junto a él, mientras que otros son más genéricos e incluyen opciones para importar proyectos de diferentes softwares CFD.

Otra opción pasa por crear rutinas propias en algún lenguaje de programación para tener mayor control sobre las magnitudes deseadas y la organización o almacenaje de las mismas. Esto también permite obtener en gráficas con mayor grado de personalización y automatización los datos generados por el solver.

Como en el actual trabajo se ha utilizado dos programas de cálculo diferentes, uno con licencia comercial y el otro gratuito, para el post-procesado se ha combinado el uso de todos los tipos de programas mencionados, incluyendo rutinas propias programadas en Matlab. Se incluyen fragmentos del código como apéndices al final de la memoria (Anexo B).

Capítulo 4

Metodología

En este capítulo se describe la configuración experimental del problema, cuyos resultados servirán para validar las simulaciones realizadas. También se expone el software utilizado para el cálculo CFD, la manera de proceder para comparar resultados y la configuración numérica. Finalmente se expone un estudio de independencia de malla, necesario para calibrar las simulaciones tratando de obtener los mejores resultados en el menor tiempo posible.

4.1. CONFIGURACIÓN EXPERIMENTAL

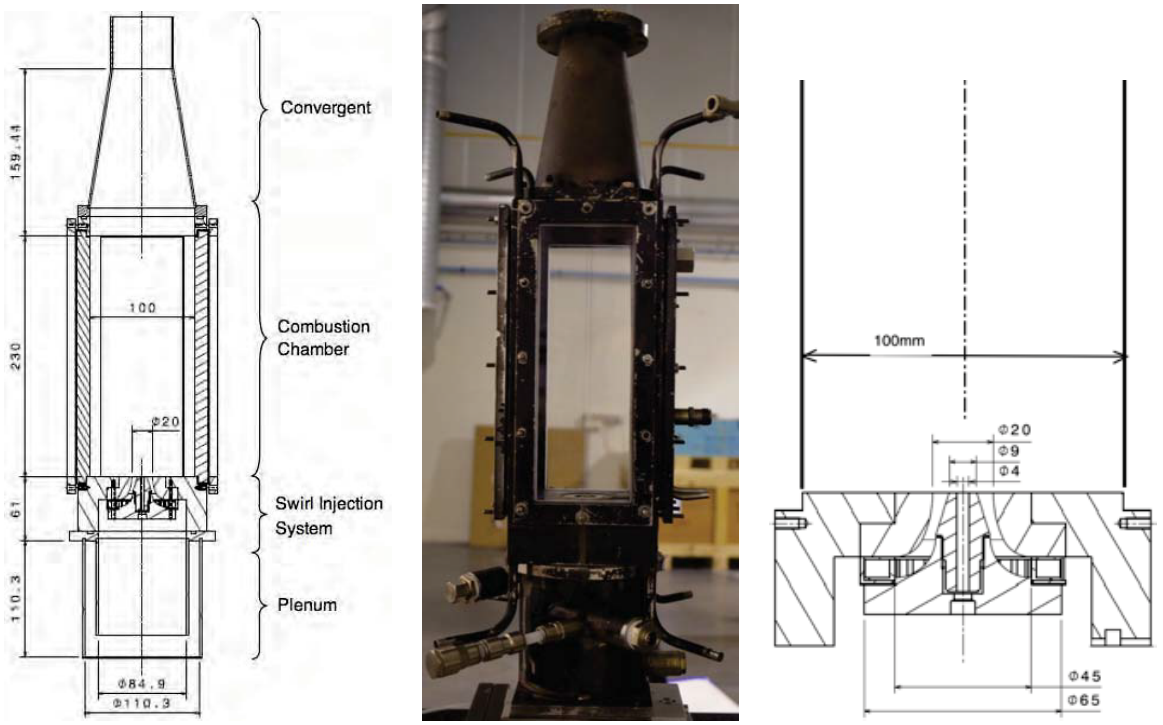
Para validar el código CFD se emplea la geometría del quemador KIAI monoinyector instalado en CORIA (COmplexe de Recherche Interprofessionel en Aérothermochimie) para estudios dedicados a la comprensión del fenómeno de ignición [68, 69]. Es una unidad mixta de investigación del CNRS, la Universidad de Rouen y el INSA de Rouen, y forma parte del proyecto de investigación europeo KIAI (Knowledge for Ignition, Acoustics and Instabilities) que aborda soluciones innovadoras para el desarrollo de *lean combustors* fiables en motores de aviación. Los estudios presentados han sido llevados a cabo con configuraciones académicas simplificadas, pero siendo suficientemente representativas para proporcionar resultados relevantes para aplicaciones industriales.

4.1.1. Geometría Del Quemador KIAI

La configuración del quemador consta de cuatro componentes como se puede observar en el esquema de la figura 4.1(a):

- El plenum, que decelera el flujo antes de entrar en el sistema de inyección. Una red colocada en la parte inferior del plenum destruye las estructuras grandes del flujo formadas en la línea de admisión de aire.

- El sistema de inyección (figura 4.2(c)) está inspirado en los de la industria aeronáutica con dos admisiones. En el centro del swirler, un tubo de diámetro $d = 4mm$ actúa como inyector de combustible (figura 4.2(a)) y el swirler (figura 4.2(b)) es radial con un diámetro de salida externo $D = 20mm$ y está compuesto por 18 canales inclinados 45°. Además, como unión entre swirler y cámara de combustión, existe un conducto de Venturi convergente formando 45º con la línea transversal. Este se encarga de dirigir al flujo hacia el interior de la cámara en la dirección adecuada.
- La cámara de combustión tiene una sección transversal cuadrada con una longitud de borde $100mm$, asegurando un campo simétrico de flujo. La cámara tiene una longitud total de $260mm$. Se utiliza cuarzo sintético para proporcionar acceso óptico de $228 * 78mm^2$ en al menos tres lados. Estas ventanas permiten diagnósticos como PIV (Particle Image Velocimetry) y PLIF (Planar Laser Induced Fluorescence), que son técnicas ópticas para obtener el campo de velocidades en el interior de la cámara.
- Finalmente se dispone de una tobera convergente para prevenir la entrada parásita de aire, potencialmente inducida por la depresión generada con el flujo recirculatorio del swirler.

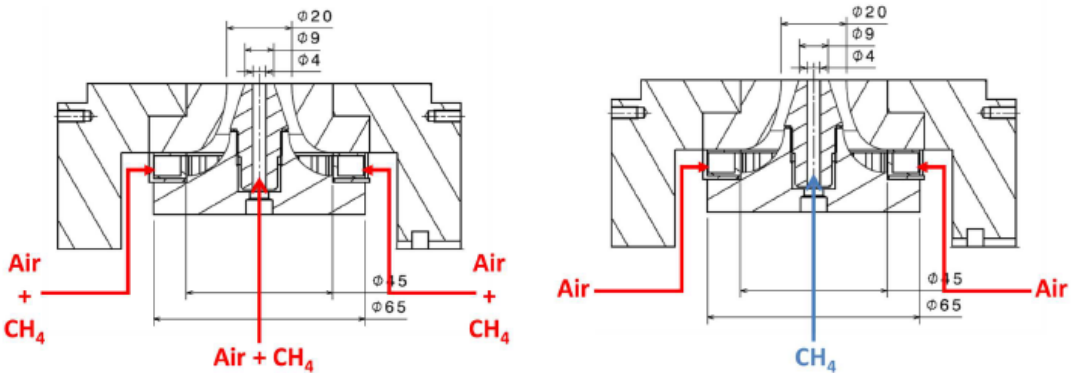


(a) Esquema descriptivo (b) Fotografía (c) Zoom del esquema descriptivo

Figura 4.1: Quemador KIAI.

(a) Inyector de combustible (b) Swirler radial (c) Sistema de inyección

Figura 4.2: El sistema de inyección del quemador KIAI.



(a) Premezclada (b) No premezclada

Figura 4.3: Estrategias de inyección gaseosa.

4.1.2. Condiciones Iniciales Y De Contorno

Las condiciones del flujo inicial y estacionario son de temperatura $T = 298K$ y presión ambiente $P = 101325Pa$, es decir, en condiciones no reactivas. El modo de operación escogido para validar el problema es, por cuestiones de simplicidad, una inyección gaseosa premezclada. Como muestra la figura 4.3(a) a través del plenum y de la línea de combustible se introduce una mezcla pobre de aire y metano. Esta validación es necesaria para confirmar las predicciones realizadas por los códigos OpenFOAM y CONVERGE de las principales características y estructuras transitorias generadas en el interior de la cámara. Como se puede observar en la figura 4.4, existen tres zonas diferenciadas en un flujo de estas características: el chorro torbellinado (Swirled Jet, SWJ), la zona de recirculación central (Inner Recirculation Zone, IRZ) y la zona de recirculación en las esquinas (Corner Recirculation Zone, CRZ).

Las condiciones de contorno a las entradas del plenum y de la línea de combustible se definen replicando el caso experimental premezclado [70], imponiendo un gasto másico de mezcla aire/metano con dosado relativo $\Phi = 0.75$, siendo el gasto a través del swirler $\dot{m}_{sw} = 5.612g/s$ y el gasto a través de la línea de combustible $\dot{m}_{ln} = 0.236g/s$. A la salida de la cámara de combustión se establece el valor de presión estática $P =$

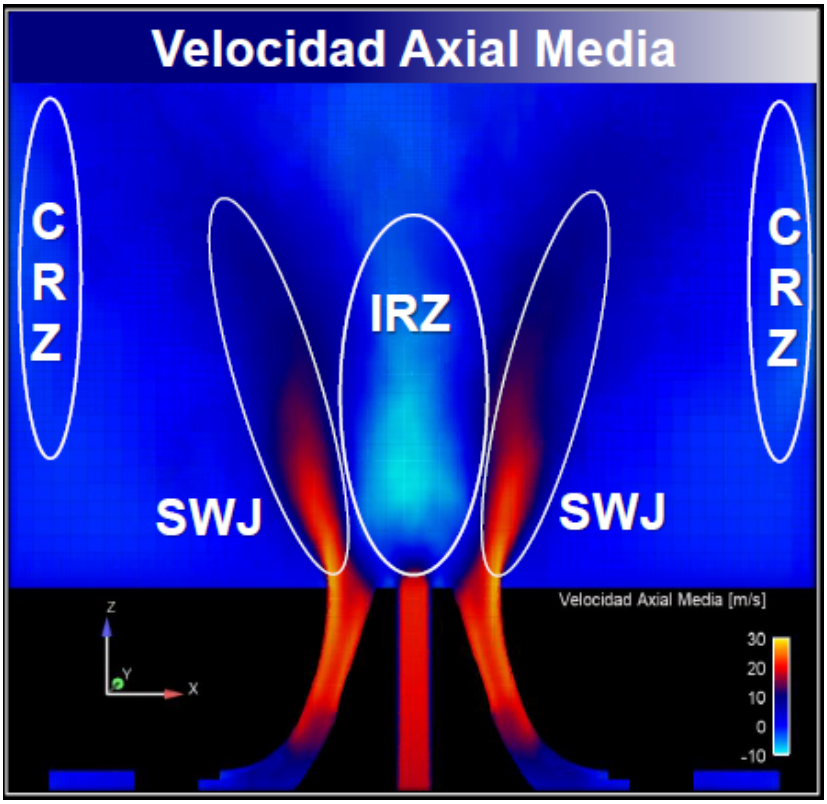


Figura 4.4: Estructura de un flujo torbellinado en el interior de la cámara de combustión del proyecto KIAI.

$101325Pa$. En el caso de las aproximaciones URANS y LES se ha empleado una solución RANS estacionaria como punto de partida realista.

Tras la validación del código mediante la inyección gaseosa premezclada, el siguiente paso es considerar el caso no premezclado, obviamente más cercano al esquema de combustión real, que se observa en la figura 4.3(b). En este nuevo estudio se añaden algunas variables que pasan a ser de gran importancia, como la concentración de especies en las estaciones cercanas a la entrada de la cámara de combustión.

4.2. SOFTWARE EMPLEADO

En este apartado se describen los principales programas de cálculo CFD que han sido empleados en la realización del presente trabajo. También se ha utilizado una máquina virtual debido a que cada uno necesitaba un sistema operativo distinto (Linux Ubuntu 16.04 y Microsoft Windows 10).

4.2.1. OpenFOAM

El software de CFD gratuito y de código abierto OpenFOAM se puede resumir como un conjunto de módulos y bibliotecas en lenguaje C++ que permiten la resolución de

problemas que involucran complejos fenómenos como la turbulencia, las reacciones químicas o la transferencia de calor (figura 4.5). Entre las principales ventajas del código se encuentra el amplio rango de solvers, librerías y herramientas (utilities) de pre- y post-procesado disponibles para la manipulación de datos. Asimismo, al tratarse de un software libre no se requieren licencias y dispone de una gran comunidad de usuarios desarrolladores de funcionalidades.

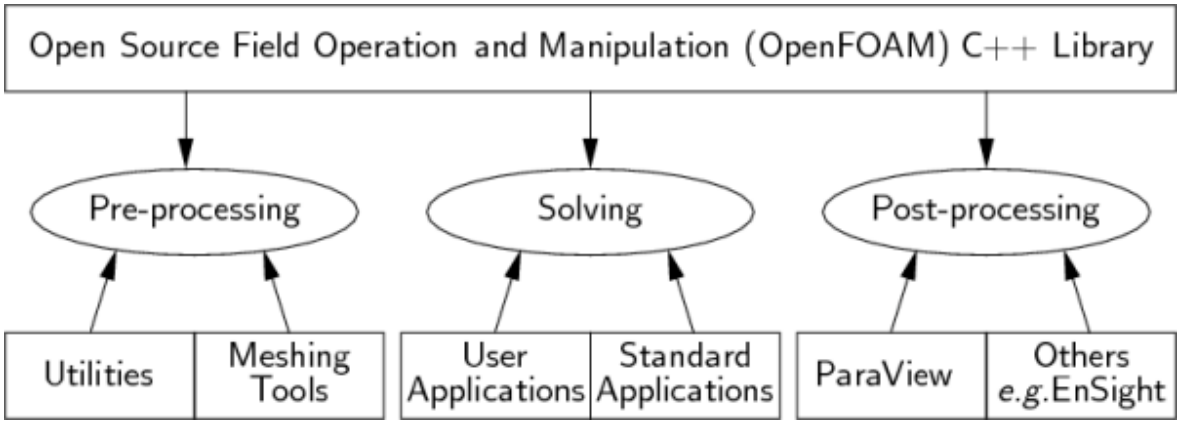


Figura 4.5: Estructura general de OpenFOAM©. [71]

No obstante, esta comunidad desarrolladora sufre una gran fragmentación con numerosos proyectos dando lugar a bifurcaciones del software que avanzan en paralelo. La ausencia de interfaz gráfica y la pronunciada curva de aprendizaje necesaria para agregar funcionalidades (User Applications) son los mayores inconvenientes y limitantes del uso del software por parte del usuario medio. Además, algunas herramientas innovadoras como el mallado adaptativo (AMR) todavía se encuentran en una fase temprana con limitaciones en su uso.

En la figura 4.6 se muestra la estructura del directorio de un caso básico con el conjunto de archivos mínimos para lanzar una simulación. Se compone de un directorio /0 que contiene las condiciones iniciales y de contorno de la presión y velocidad, así como de aquellas que requieran los modelos empleados. En el directorio /constant se almacena la toda la información concerniente a la malla (polyMesh) y a las propiedades físicas de los fenómenos acontecidos en la aplicación de estudio (modelos de turbulencia, propiedades del fluido, etc.). Por último, en el directorio /system se encuentran los parámetros de configuración relacionados con los esquemas de discretización empleados para cada función de las ecuaciones (fvSchemes), los algoritmos numéricos con sus respectivas tolerancias (fvSolution) y los parámetros temporales que controlan la simulación (controlDict).

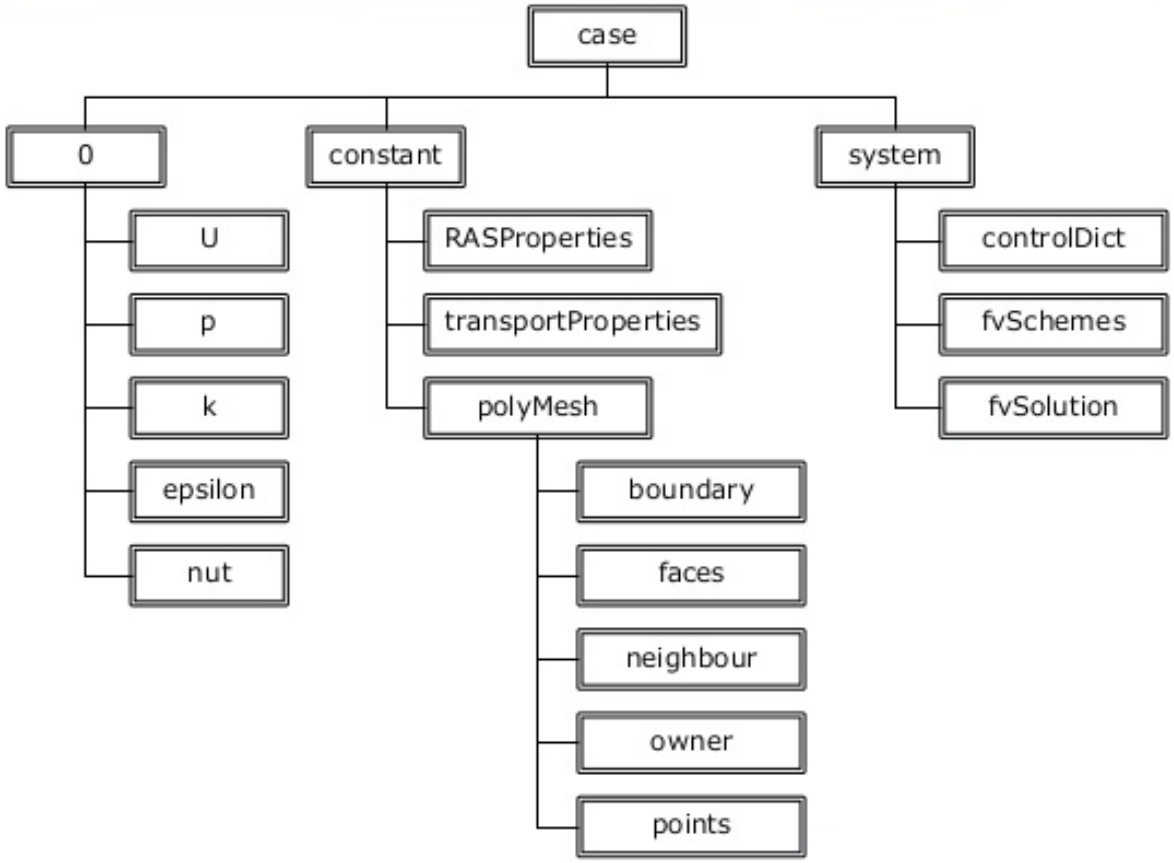


Figura 4.6: Estructura del directorio de un caso en OpenFOAM. [71]

4.2.2. CONVERGE

CONVERGE es un software comercial de CFD basado en el método de volúmenes finitos que elimina el cuello de botella que supone la generación de la malla en las etapas de pre-proceso de las simulaciones. Fue desarrollado inicialmente para aplicaciones concernientes a los motores de combustión interna alternativos, pero durante los últimos años su campo de aplicación se está ampliando e impulsando a otros ámbitos ingenieriles, especialmente al estudio de turbinas de gas. A diferencia del resto de programas CFD, CONVERGE genera de manera automática una malla ortogonal, perfectamente estructurada durante la propia simulación mediante unos sencillos parámetros de control definidos por el usuario [72].

Para crear una geometría desde el propio software o importar un modelo 3D existente se puede utilizar CONVERGE Studio, herramienta complementaria y adaptada a este solver (ya que ha sido desarrollado por la misma compañía) que permite el pre-proceso y post-proceso del caso. A la hora de reconocer la geometría, el programa necesita representar a través de triángulos las superficies, acción que lleva a cabo automáticamente tras la importación del archivo. Sin embargo, el usuario tiene capacidad de modificar dichos triángulos para reparar errores que se puedan encontrar, o incluso

disponerlos de manera que mejoren el resultado obtenido por el algoritmo automático que aplica el software.

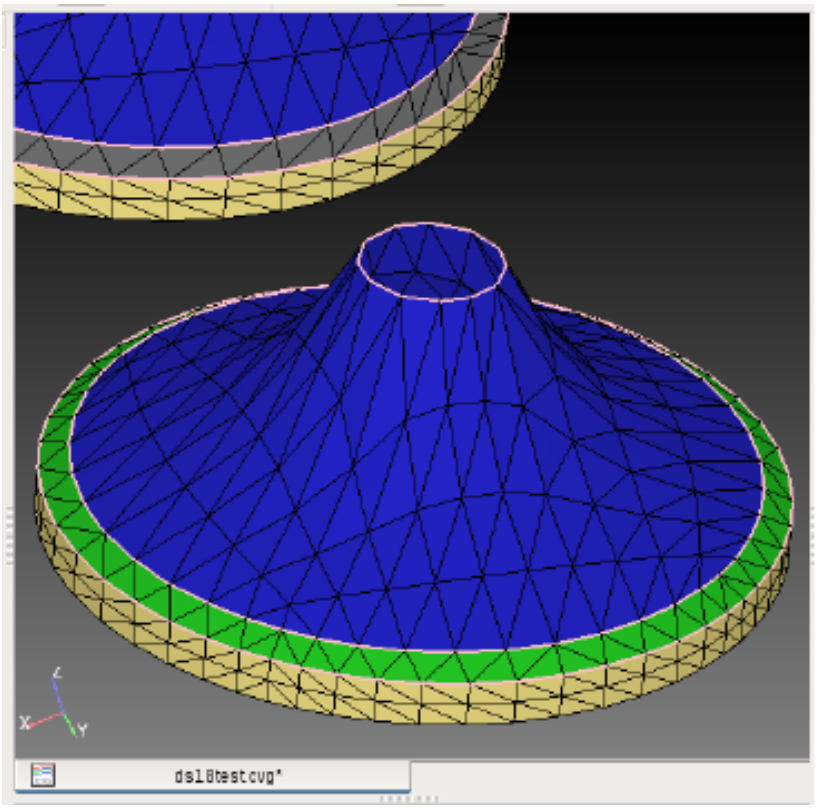


Figura 4.7: Ejemplo de modelo 3D triangulado en CONVERGE Studio [73].

Una vez terminado este paso, ya no es necesario repetirlo más, siempre y cuando se vaya a usar la misma geometría. Será CONVERGE el que, interpretando la disposición de los elementos, cree la malla hexaédrica estructurada (que presenta mayor eficiencia computacional y simplifica los métodos numéricos del solver respecto a las no estructuradas, como se ha mencionado en el apartado 3.5) siguiendo las indicaciones del usuario sobre las características de la misma. Por estos motivos presenta enormes ventajas en términos de eficiencia.

CONVERGE incluye diversas herramientas para controlar el tamaño de la malla antes y durante la simulación, refinando o ensanchando el tamaño base en potencias de 2. Algunas de estas herramientas se ven en las figuras 4.8-4.9 y se explican a continuación.

- Grid Scaling, que refina o ensancha de manera global el dominio completo. Resulta especialmente útil durante las primeras iteraciones de la simulación hasta que el flujo se establece por completo.

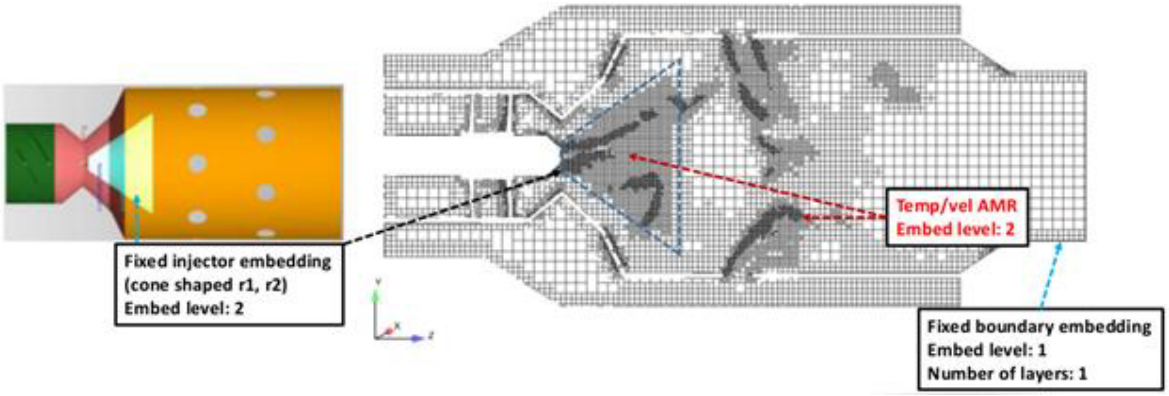


Figura 4.8: Aplicación de las herramientas de control sobre el mallado Fixed Embedding y AMR en un quemador LDI [74].

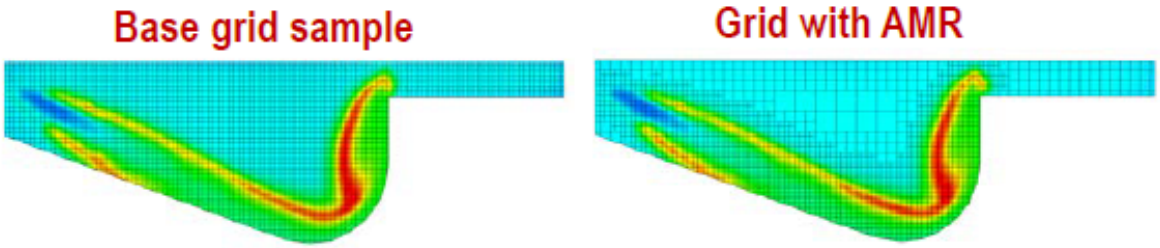


Figura 4.9: Diferencia entre mallado sin y con aplicación de AMR.

- Fixed Embedding, encargado de refinar la malla en regiones específicas del dominio donde se requiera de una mayor resolución para obtener mayor precisión en la solución. En el trabajo actual resulta interesante su uso en la región del swirler y en la zona de recirculación, cercana a la inyección del combustible.
- Adaptive Mesh Refinement (AMR). Es la herramienta más distintiva del programa, que permite refinar la malla de manera automática durante la propia simulación añadiendo resolución en las zonas donde el campo fluido no se encuentre resuelto satisfactoriamente. Para ello se basa en la detección de los altos gradientes de las variables de interés. Esta opción resulta de gran ayuda para capturar de manera precisa fenómenos como la propagación de la llama o flujos con estructuras complejas, como los presentes aguas abajo del swirler en los quemadores LDI con un coste computacional razonable. Cabe destacar que cuanto menor sea el umbral de AMR definido por el usuario, mayor sensibilidad tendrá el código para refinar. Además, se puede liberar el refinamiento automático de las regiones adyacentes a las paredes, evitando así modificar el valor de y^+ de tal manera que caiga fuera del rango deseable por el modelo de pared aplicado y produciendo resultados físicamente no realistas.

4.3. CÁLCULO DE ERRORES Y DESVIACIÓN

Cuando se trata de comparar gráficas, la manera más intuitiva consiste en hacer una primera inspección visual. Sin embargo, este método es insuficiente para extraer toda la información necesaria, sirve únicamente para obtener una primera aproximación.

La forma habitual de calcular errores (en este caso se entiende error como la diferencia entre el valor de la simulación y el experimental) es mediante la fórmula del error relativo (ecuación 4.1). Con este método, ampliamente utilizado en todos los campos de la ciencia, se consigue un valor porcentual de la desviación.

$$error_{relativo} = \frac{|magnitud_{real} - magnitud_{medida}|}{magnitud_{real}} * 100 \tag{4.1}$$

Sin embargo, este método tiene algunas limitaciones, y es que para magnitudes reales verdaderamente pequeñas, el error se dispara hasta límites excesivamente altos. Esto es debido a que, al dividir entre números cada vez más pequeños, la ecuación tiende a infinito. En el presente estudio, este problema ocurre a la hora de comparar velocidades medias o desviaciones típicas, ya que conviven magnitudes de alto valor absoluto (positivas y negativas) con magnitudes cercanas al cero.

La solución optada consiste en utilizar otro método, la Diferencia Porcentual Relativa o Relative Percent Difference (ecuación 4.2) [75]. Un ejemplo de aplicación del mismo es en el campo de la química, ya que en ocasiones las concentraciones son tan pequeñas que no puede medirse la desviación con el error relativo.

$$RPD = \frac{|magnitud_{real} - magnitud_{medida}|}{(magnitud_{real} + magnitud_{medida})/2} * 100 \tag{4.2}$$

En la figura 4.10 se muestra la relación existente entre el error relativo y la RPD. En un intervalo de errores relativos cercano al 50 % (tanto positivo como negativo) los porcentajes son muy similares a éste, aunque al salirse de este intervalo se sobreestiman los errores negativos y se subestiman los positivos.

Entre los resultados de los casos experimentales que se pretende replicar en las simulaciones se tomaron medidas de determinadas variables en ciertas localizaciones. Sin embargo, cuando se trata de comparar de una manera eficiente unos casos con otros, es necesario obtener un valor único de precisión representativo del caso, o al menos por cada variable o conjunto de variables que se quiera estudiar de forma separada. Lo más sencillo es realizar una media aritmética entre los porcentajes RPD de todas las medidas de dicha variable, por lo que se ha elegido este método como manera de proceder. Por ejemplo, si se quiere medir la velocidad axial en diferentes estaciones o posiciones de la geometría, se calcula el RPD en cada punto en relación al resultado experimental, se

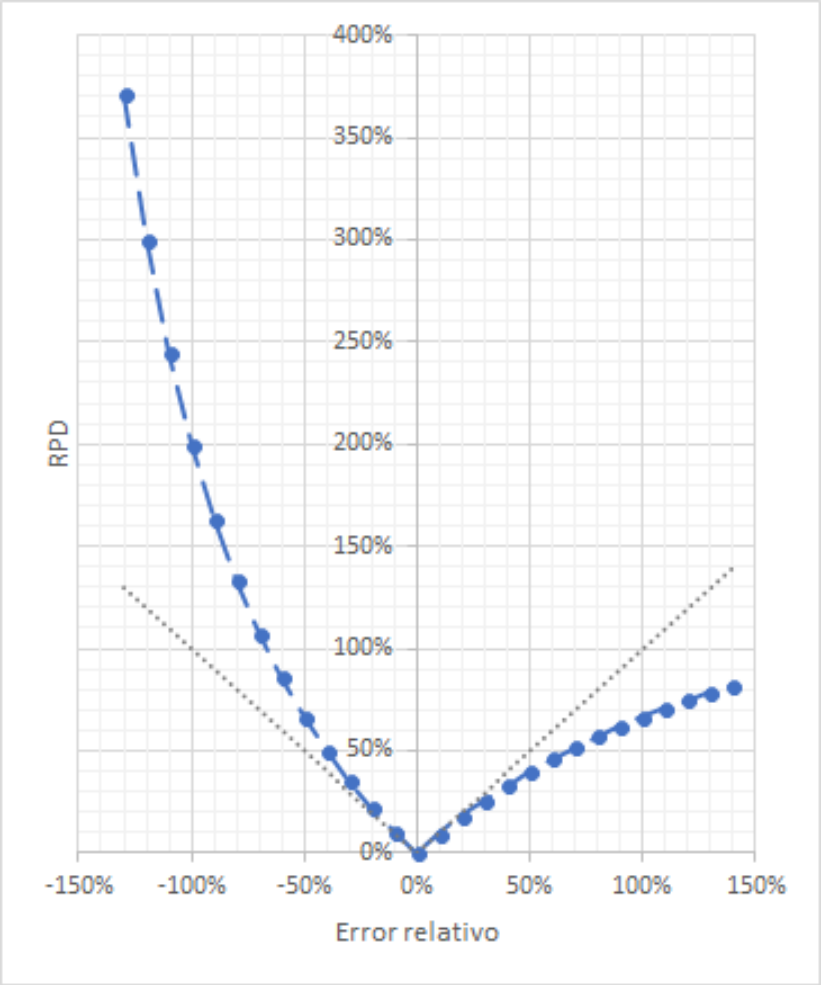


Figura 4.10: Comparación entre RPD y error relativo.

promedia entre todos los puntos para cada estación, se promedian todas las estaciones y se obtiene un escalar que, aunque no sea intuitivo, es comparable a otro obtenido de igual manera.

Es importante destacar que este valor no es absoluto, es una magnitud relativa que sirve para comparar en cierta variable unos casos con otros. Sin embargo no es correcto comparar para un mismo caso distintas variables: si en un caso no premezclado, para la velocidad media en las estaciones se obtiene un 60 % de error pero para la concentración de especies se obtiene un 30 %, esto no significa que se haya conseguido un resultado más fiel en la segunda que en la primera. Puede parecer que para una determinada magnitud un RPD de (por ejemplo) 80 % sea algo desorbitado, pero este es resultado de muchos promedios y no tiene por qué ser un error excesivo, cosa que se demuestra al realizar gráficas de los perfiles y contornos de esas variables. También conviene recordar que los resultados experimentales no están exentos de error, aunque en este trabajo se tomen como referencia y se les suponga una precisión adecuada pueden alejarse de la realidad.

4.4. CONFIGURACIÓN NUMÉRICA

Una vez explicada la configuración experimental en el apartado 4.1, el siguiente paso consiste en preparar la geometría para introducirla en cada uno de los softwares de cálculo CFD de los que se dispone, así como describir las condiciones iniciales y de contorno que repliquen el ensayo experimental. Además se debe introducir la turbulencia correctamente según el tipo de simulación.

Existen multitud de parámetros a la hora del mallado así como esquemas de discretización y de turbulencia disponibles para el estudio del problema, por lo que al final del capítulo se propone un estudio de independencia de malla como primer paso cuyo objetivo es calibrar la precisión y duración de las simulaciones.

4.4.1. Geometría En OpenFOAM

El proceso de creación de la geometría y su posterior mallado se ha realizado mediante ANSYS Workbench. Con la información de la que se disponía en los estudios experimentales se ha reproducido el quemador KIAI para, a continuación, crear múltiples mallas que permitan estudiar los fenómenos deseados.

Las mallas se han creado con elementos tetraédricos, partiendo de un tamaño base en la región del plenum y en la cámara, y refinando la región del swirler. Además, tratando de capturar con mayor precisión la zona de recirculación, se ha refinado la zona central de la cámara de combustión con esferas a diferentes distancias medidas desde la entrada de la misma. Se puede ver un ejemplo de malla utilizada en la figura 4.11, con la estrategia mencionada.

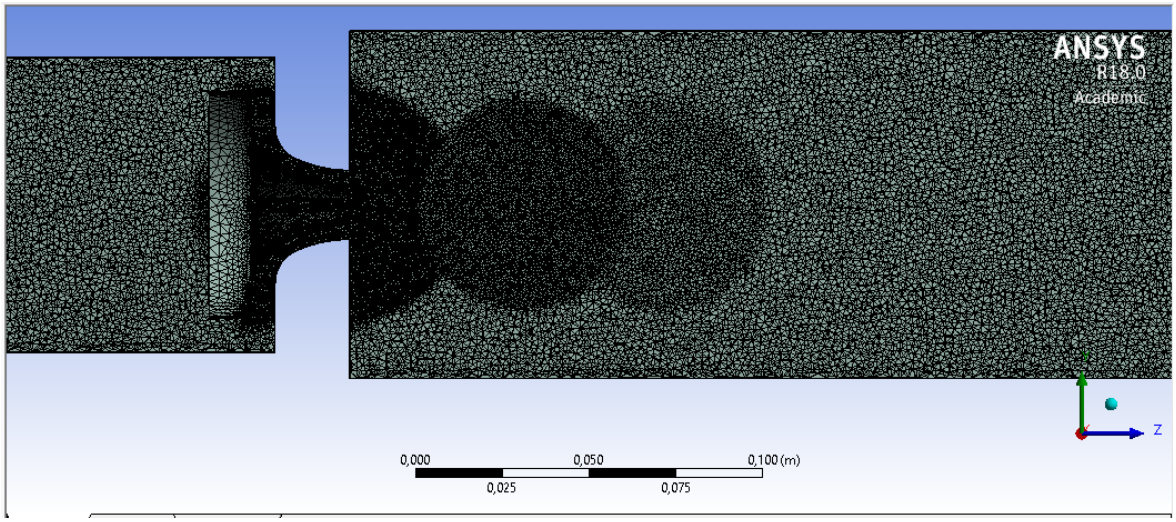


Figura 4.11: Ejemplo de estructura de mallado del quemador KIAI con 6 millones de celdas.

Celdas (M)	0,5		0,8		1,2		3		6		17	
/	Tam (mm)	Radio (mm)	Tam (mm)	Radio (mm)	Tam (mm)	Radio (mm)	Tam (mm)	Radio (mm)	Tam (mm)	Radio (mm)	Tam (mm)	Radio (mm)
Base (mm)	6,5	/	5,0	/	5,0	/	5,0	/	3,0	/	3,0	/
0 (mm)	2,0	33,0	1,5	33,0	1,2	33,0	0,8	33,0	0,8	33,0	0,4	33,0
10 (mm)	2,0	33,0	1,5	33,0	1,2	33,0	0,8	33,0	0,8	33,0		
swirler	2,0	/	1,5	/	1,2	/	0,8	/	0,8	/	0,4	/
50 (mm)	3,0	30,0	2,0	30,0	1,5	30,0	1,5	30,0	1,1	30,0	1,1	33,0
90 (mm)	4,0	30,0	3,0	30,0	2,0	30,0	2,0	30,0	1,5	30,0	1,5	30,0

Figura 4.12: Tabla con los parámetros de las mallas utilizadas en OpenFOAM.

Además, en la tabla de la figura 4.12 están descritas todas las mallas utilizadas en este apartado. La primera columna indica las distancias entre el eje de coordenadas y el centro de cada esfera (excepto el tamaño base y el mallado concreto del swirler), la primera fila es el número de celdas en millones de cada malla y la segunda fila indica por columnas el tamaño de los elementos y el radio de las esferas de cada malla.

4.4.2. Geometría En CONVERGE

La geometría y triangulación para el caso de CONVERGE se ha realizado mediante CONVERGE Studio, como se muestra en la figura 4.13. A la hora de controlar el mallado en este software, se deben definir correctamente los parámetros mencionados en el apartado 4.2.2.

Para generar la malla estructurada por defecto, en primer lugar el programa subdivide el dominio en hexaedros teniendo en cuenta el tamaño base definido y el *grid scaling*, que es utilizado para acelerar el establecimiento del flujo en el dominio, y así también la convergencia del código. A continuación se obtiene la intersección entre estas celdas y la triangulación de la geometría, consiguiendo una gran cantidad de celdas cúbicas y un número menor de celdas irregulares (figura 4.14) correspondientes a los bordes y superficies más complejos de la geometría. Después se aplica el parámetro de *fixed embedding* en las zonas indicadas para refinar las celdas. Finalmente, el *AMR* se tiene en cuenta durante la simulación, comparando los gradientes de las variables elegidas con los límites establecidos para refinar en caso de sobrepasarlos. De esta manera se consigue aumentar la resolución de la malla de manera local en secciones críticas manteniendo elementos más gruesos en el resto del dominio.

Los niveles de *AMR* y *fixed embedding* se han variado entre 0 y 3, pero en el segundo además se ha modificado el tamaño de la región de influencia. Por su parte, se ha analizado el efecto de adelantar o atrasar el instante en el que deja de actuar el *gridscale*. Se ha llegando a un compromiso ya que, si deja de actuar antes de lo

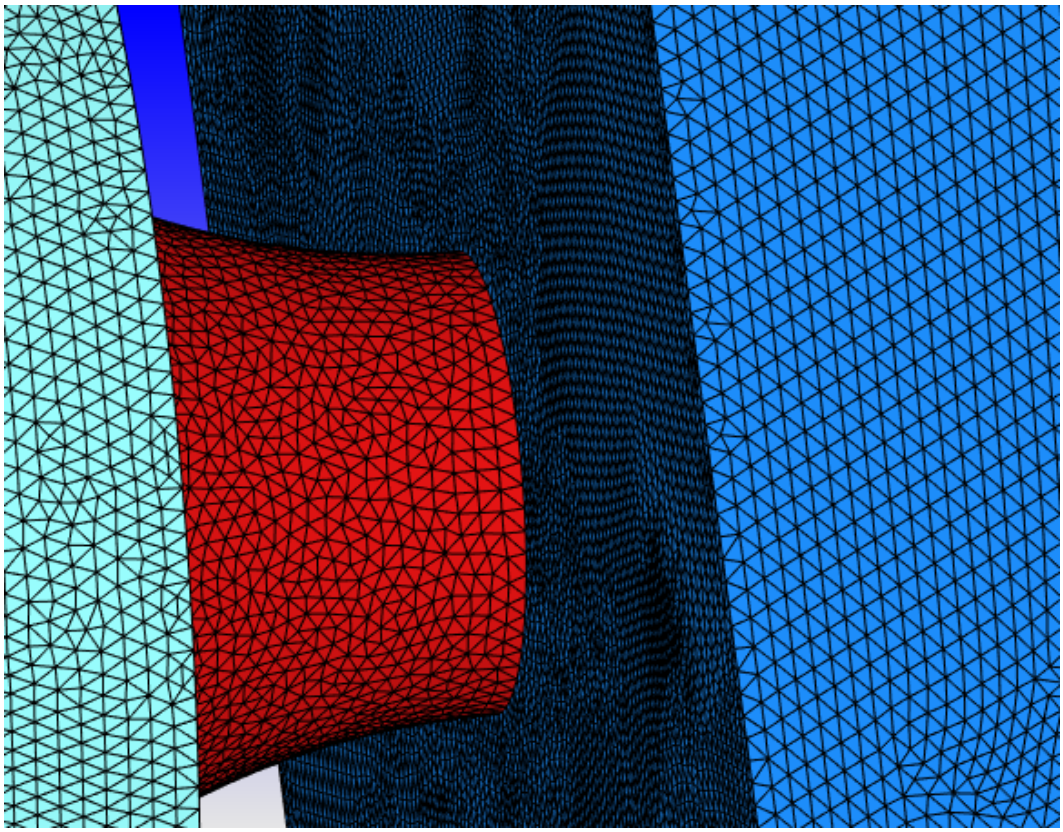


Figura 4.13: Triangulación del quemador KIAI en CONVERGE Studio, zoom en el swirler y entrada a la cámara de combustión.

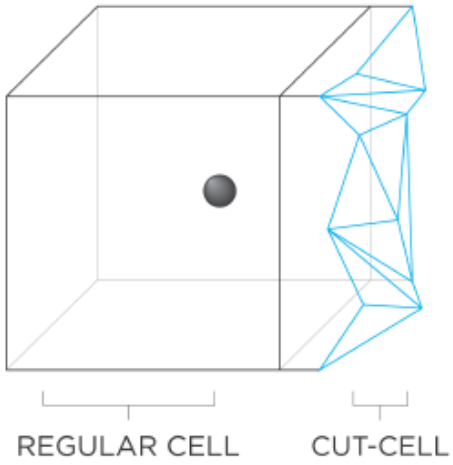


Figura 4.14: Tipos de celda tras el mallado automático [72].

debido la simulación se vuelve más pesada y el flujo tardaría un tiempo mayor en estabilizarse, pero si deja de actuar mucho después la solución se vería frenada por la gruesa resolución de la malla sin avanzar hacia el resultado final.

4.4.3. Modelado De La Turbulencia

En las simulaciones RANS la turbulencia se especifica mediante una intensidad turbulenta (ecuaciones 3.17).

Aunque se han llevado a cabo una gran cantidad de simulaciones variando los valores de intensidad turbulenta entre el 2 % y el 15 % en ambas entradas, y la escala de longitud turbulenta entre un 5 % y un 12 % del diámetro, los mejores resultados se han obtenido con la configuración de intensidad turbulenta del 2 % a la entrada del plenum y 10 % en la línea de combustible.

Otro factor importante es el tratamiento de pared considerado y los valores de y^+ presentes en la zona de mayor velocidad: la sección convergente que sigue al swirler. Mediante los niveles de refinamiento se ha tenido la cautela de, independientemente de la configuración de los parámetros de las herramientas de mallado, encontrarse con valores de y^+ entre 30 y 100, asegurando por tanto que la primera celda se encuentra en la capa logarítmica donde los modelos de la ley de pared actúan con una mejor aproximación.

Los promediados temporales y fluctuaciones de las variables de interés en las simulaciones URANS se efectúan una vez el gasto másico a la salida del quemador se ha establecido. Este cálculo se lleva a cabo durante $15ms$, es decir, unas 20 veces el tiempo de residencia del flujo en el dominio que se define según la ecuación 4.3, donde R_i es el radio medio de la entrada convergente y u_θ es la componente azimutal de la velocidad media en el plano de la entrada [76].

$$t_{res} = \frac{\pi R_i}{u_\theta} = 0.81 \tag{4.3}$$

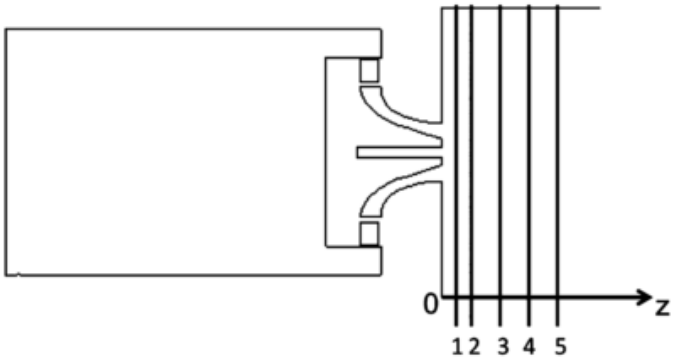
En las simulaciones LES se debe refinar mucho más la malla, de modo que se consiga obtener unos y^+ lo suficientemente pequeños (menores de 4) como para que la ley de pared se corresponda con la capa viscosa. Esto también permite capturar y calcular con mayor precisión las estructuras del flujo más pequeñas, por eso a medida que se refina la malla, mejores son los resultados.

4.4.4. Variables De Estudio

Los datos experimentales con los que se comparan las simulaciones del presente trabajo forman parte de las tesis y artículos [70, 77, 78, 79]. En ellos se presentan

los resultados e incluso se comparan con simulaciones realizadas mediante diferentes códigos CFD como AVBP y YALES2. Como los datos han sido tomados de 3 maneras diferentes, es necesario que el procesado y presentación de los resultados de las simulaciones se realice según esos mismos formatos, que se describen a continuación.

- Centerline o línea central. Las variables se toman a lo largo de la recta que se dibuja desde el punto central a la entrada de la cámara de combustión (donde se ubica el eje de coordenadas de la geometría) en dirección axial hasta el final de la misma.
- Stations o estaciones. Las variables se toman en planos perpendiculares a la línea central del quemador, a las distancias de $z = 5, 10, 20, 30$ y $40mm$ ($z/D = 0.25, 0.5, 1, 1.5, 2$). En estos planos, se miden las variables a lo largo de un eje radial (ver figura 4.15).



Estaciones	Coordenadas adimensionales
1	$z/D = 0.25$
2	$z/D = 0.5$
3	$z/D = 1$
4	$z/D = 1.5$
5	$z/D = 2$

Figura 4.15: Posición de las estaciones donde tomar las medidas en el quemador.

- Contours o contornos. Las variables se toman en un plano que divide la cámara en dos partes iguales. El valor de las variables se plasma mediante una escala de colores que figura en la misma leyenda de la figura.

Los parámetros más importantes que se han medido en los experimentos y simulaciones son la velocidad media (o mean velocity) y las medias cuadráticas o RMS (root mean squared) en todas sus componentes: axial, radial y tangencial. En el caso no premezclado también se ha medido la concentración del combustible.

Los valores RMS son una medida estadística de la magnitud de una cantidad variable y representan la incertidumbre o el grado de dispersión entre unos datos y su media. Se calculan como se indica en la ecuación 4.4 para la velocidad, mediante la raíz cuadrada del promedio de todos los datos elevados al cuadrado (para eliminar los signos).

$$v_{RMS} = \sqrt{\frac{v_{ax}^2 + v_{rad}^2 + v_{tan}^2}{3}} \tag{4.4}$$

4.5. RUTINAS DE POSPROCESADO

La cantidad de simulaciones que se han realizado en el trabajo actual requieren de un tiempo muy grande para posprocesar los resultados, realizar gráficas, comparar unos casos con otros y sacar conclusiones. Por ese motivo ha sido necesario realizar unas rutinas en Matlab capaces de automatizar esta tarea, mostrando y guardando las gráficas de los casos y sus comparaciones, calculando desviaciones entre los datos experimentales y las simulaciones, dibujando contornos de variables sin tener que cargar programas muy pesados de posprocesado y mantener un entorno de carpetas ordenado donde almacenar toda la información.

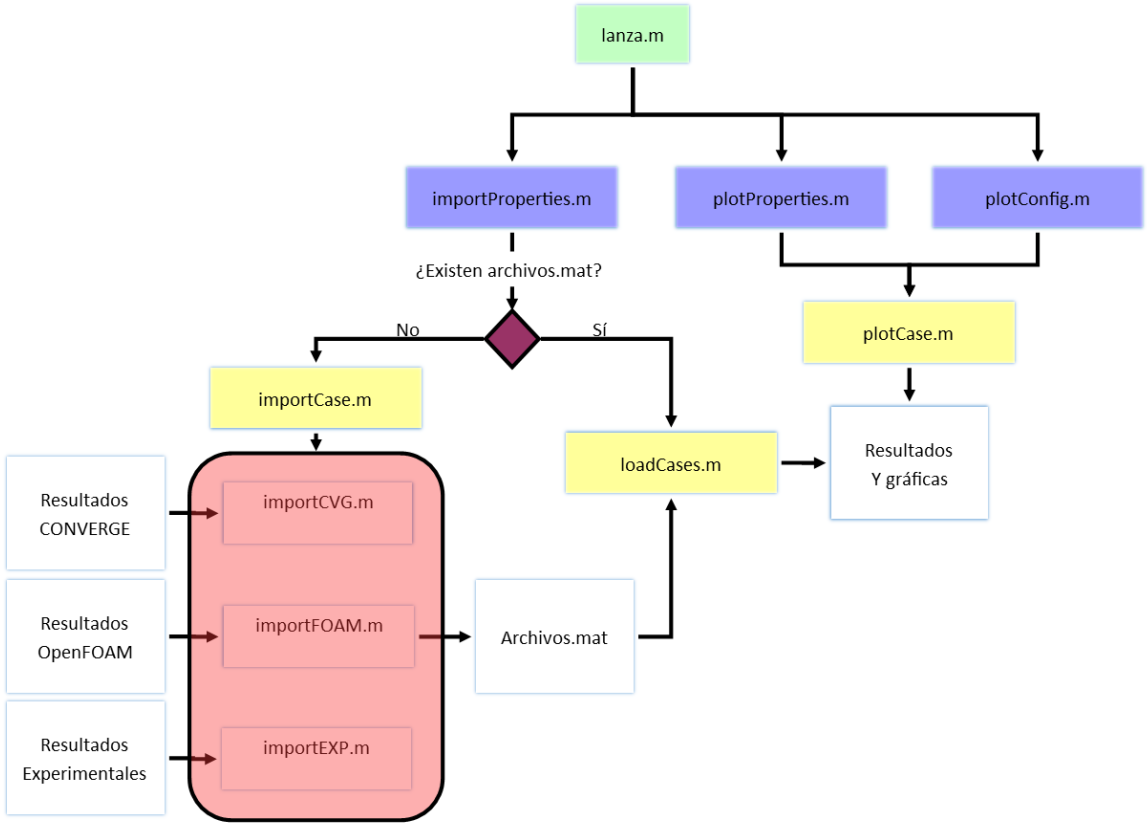


Figura 4.16: Diagrama de funcionamiento de las rutinas de posprocesado.

Las rutinas de posprocesado que se adjuntan en el anexo B funcionan de forma resumida como el diagrama que se muestra en la figura 4.16. Previamente el usuario debe definir correctamente las variables del lanzador (en verde) y de los archivos de configuración (en azul). A continuación se ejecuta el lanzador que comprueba si existen los archivos *mat* requeridos donde se guarda la información ya en los formatos especificados anteriormente. Si no es así llama a las funciones que importan los datos (en rojo) y los transforman en archivos *mat*, pero si ya existen simplemente los importa. A continuación, dependiendo de las preferencias indicadas, los resultados pueden verse en la consola y en gráficas que realiza la función *plotCase.m*, que además pueden

guardarse en ficheros pdf, tiff, u otras extensiones. Es importante mantener los casos en un entorno de carpetas ordenado como se indica en los comentarios del código, y asegurarse.

4.6. ESTUDIO DE INDEPENDENCIA DE MALLA

En el presente trabajo es necesario realizar un estudio de independencia de malla a partir de las numerosas simulaciones del caso premezclado llevadas a cabo en CONVERGE y OpenFOAM con el objetivo de obtener la configuración óptima en cuanto a precisión y tiempo de cálculo.

4.6.1. Malla OpenFOAM

En la tabla de la figura 4.17 están todos los casos realizados en OpenFOAM con premezcla, donde se refleja detalladamente las variaciones en los parámetros que se han ido sucediendo y que se explicaban del mismo modo en el apartado 4.4.

Caso	Total Celdas	Modelo Turbulencia	RPD Línea Central	RPD Media Estaciones
RANS 1	500.000	k-epsilon Std	116,86%	111,82%
RANS 3	500.000	RSM - LRR	112,32%	115,08%
RANS 4	800.000	k-epsilon Std	90,72%	102,62%
RANS 6	800.000	RSM - LRR	102,56%	105,78%
RANS 7	1.200.000	k-epsilon Std	76,38%	88,54%
RANS 9	1.200.000	RSM - LRR	84,78%	91,86%
RANS 10	3.000.000	k-epsilon Std	69,84%	87,04%
RANS 12	3.000.000	RSM - LRR	60,12%	89,66%
RANS 13	6.000.000	k-epsilon Std	64,62%	89,52%
RANS 15	6.000.000	RSM - LRR	50,94%	89,14%
RANS 17	17.000.000	k-epsilon Std	67,36%	85,34%
RANS 18	17.000.000	RSM - LRR	35,76%	64,28%

Figura 4.17: Configuración de los casos RANS y URANS con premezcla en OpenFOAM.

En las gráficas de las figuras 4.18 y 4.19 se muestran los porcentajes de RPD en función del número de celdas de cada caso, tanto para la velocidad axial en la línea

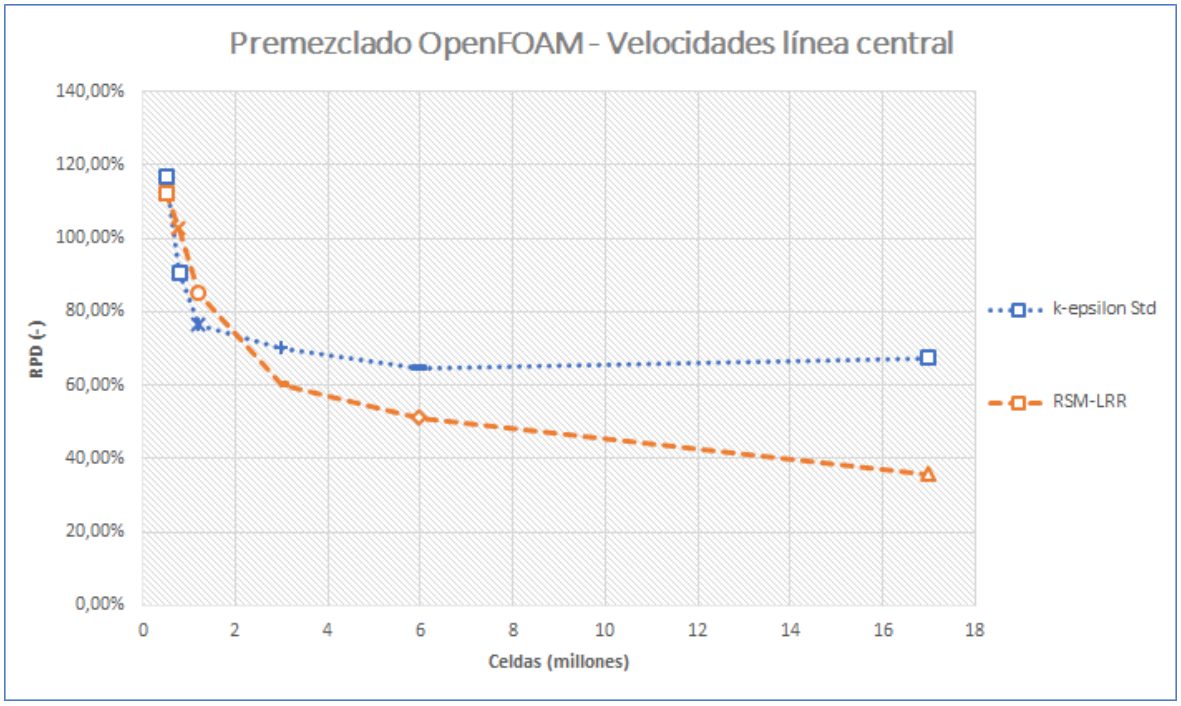


Figura 4.18: RPD de la velocidad en la línea central en función del número de celdas en la malla de OpenFOAM.

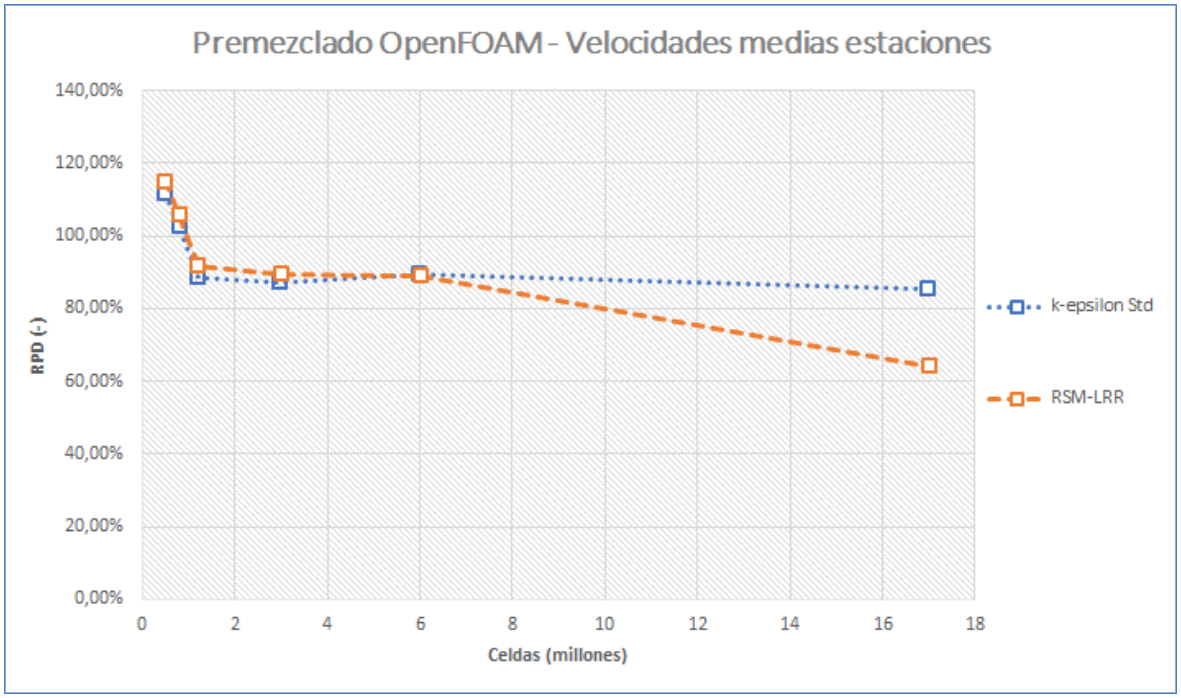


Figura 4.19: RPD de la velocidad en las estaciones de la entrada a la cámara de combustión en función del número de celdas en la malla de OpenFOAM.

central como para la media de los perfiles de velocidades (axial, radial y tangencial) en las estaciones.

Como se puede observar en ambas figuras, a medida que aumenta el número de celdas disminuye la desviación de manera potencial. Conviene destacar que las simulaciones con el modelo $k-\varepsilon$ sí que se estabilizan y llegan a una convergencia de malla, por ser un caso RANS. No obstante, la curva inferior correspondiente a RSM-LRR es URANS y por ello siempre mejora al refinar la malla, ya que con menor tamaño (y mayor cantidad) de celdas logra resolver más escalas.

Hay diferenciadas dos zonas:

- La primera comprende hasta los 2 millones de celdas, y tiene una pendiente significativa, aumentando el número de celdas se nota una diferencia grande en el error.
- La segunda zona (más de 2 millones de celdas) sigue teniendo pendiente pero es mucho menor, con lo que la variación de celdas es demasiado grande respecto de la diferencia en el error. No merece la pena aumentar tanto el peso de la malla y así el tiempo de cálculo ya que el aumento de precisión no será tan esencial.

Con estos datos, la malla más conveniente para realizar los estudios en OpenFOAM del modelo $k-\varepsilon$ es la que se compone de entre 3 a 6 millones de celdas, si las limitaciones de capacidad de cálculo son restrictivas. En caso contrario, se puede utilizar mallas de 17 millones de celdas o superior para el modelo RSM-LRR.

4.6.2. Malla CONVERGE

En la tabla de la figura 4.20 están todos los casos realizados en CONVERGE con premezcla, donde se refleja detalladamente las variaciones en los parámetros que se han ido sucediendo y que se explicaban en el apartado 4.4. Aunque algunos casos tienen el mismo número de celdas pero niveles diferentes de *fixed embedding* o *AMR* se debe a que se ha modificado la zona donde actúan.

En las gráficas de las figuras 4.21 y 4.22 se muestran los porcentajes de RPD en función del número de celdas de cada caso, igual que la sección anterior, tanto para la velocidad axial en la línea central como para la media de los perfiles de velocidades medias (axial, radial y tangencial) en las estaciones.

Como se puede observar en ellas, a medida que aumenta el número de celdas disminuye la desviación de manera potencial. Además hay unas tendencias observables según las características de la malla:

Caso	Tam Base (mm)	Fixed Embedding	AMR Levels	AMR Threshold	Total Celdas	Modelo Turbulencia	Intensidad Turbulenta	RPD Línea Central	RPD Media Estaciones
RANS 9	3	0	3	0,1	850.000	k-epsilon Std	0,02	38%	56%
RANS 10	3	3	3	0,1	2.700.000	k-epsilon Std	0,02	35%	51%
RANS 12	3	3	3	0,1	3.800.000	k-epsilon Std	0,1	34%	50%
RANS 14	3	3	3	0,1	3.900.000	k-epsilon Std	0,02	35%	51%
RANS 15	3	3	3	0,1	2.700.000	k-epsilon Std	0,02	39%	65%
RANS 16	3	2	2	0,1	1.200.000	k-epsilon Std	0,02	36%	61%
RANS 17	3	3	3	0,1	1.200.000	k-epsilon Std	0,02	36%	53%
RANS 18	3	3	3	0,1	2.800.000	K-epsilon RNG	0,02	32%	67%
RANS 19	3	3	3	0,1	4.000.000	k-epsilon Std	0,02	38%	57%
RANS 20	3	3	3	0,1	1.200.000	k-epsilon Std	0,02	37%	57%
RANS 21	3	2	3	0,1	900.000	k-epsilon Std	0,02	46%	53%
RANS 22	3	3	2	0,1	1.200.000	k-epsilon Std	0,02	37%	53%
RANS 23	3	2	2	0,1	600.000	k-epsilon Std	0,02	48%	59%
RANS 25	3	3	3	0,1	2.500.000	RSM - LRR	0,02	27%	73%
RANS 26	3	3	3	0,1	1.150.000	k-epsilon Std	0,02	38%	53%
RANS 27	3	3	3	0,1	1.150.000	k-epsilon RNG	0,02	50%	59%
RANS 28	3	3	3	0,1	1.200.000	k-epsilon Realizable	0,02	50%	60%
RANS 29	3	3	3	0,1	1.200.000	k-epsilon Std	0,02	39%	51%
RANS 30	3	2	1	0,1	450.000	k-epsilon Std	0,02	52%	63%
RANS 31	4	3	3	0,1	620.000	k-epsilon Std	0,02	41%	56%
RANS 32	4	2	2	0,1	286.000	k-epsilon Std	0,02	59%	64%
RANS 33	5	3	3	0,1	370.000	k-epsilon Std	0,02	52%	63%
RANS 34	5	3	2	0,1	225.000	k-epsilon Std	0,02	69%	116%
RANS 35	6	3	3	0,1	275.000	k-epsilon Std	0,02	49%	54%
RANS 36	3	3	3	0,25	1.050.000	k-epsilon Std	0,02	36%	51%
RANS 37	3	3	3	0,5	1.030.000	k-epsilon Std	0,02	33%	60%
RANS 38	3	3	3	0,1	1.850.000	k-epsilon Std	0,02	41%	59%
RANS 39	3	0	0	N/A	158.000	k-epsilon Std	0,02	190%	93%
RANS 41	4	0	0	N/A	69.000	k-epsilon Std	0,02	189%	124%
RANS 42	2	0	0	N/A	524.000	k-epsilon Std	0,02	81%	137%
RANS 43	4	0	4	0,1	4.000.000	k-epsilon Std	0,02	47%	58%
RANS 44	3	0	3	0,1	2.200.000	RSM - LRR	0,02	32%	84%
RANS 45	5	0	4	0,1	2.200.000	k-epsilon Std	0,02	52%	68%
RANS 46	6	0	4	0,1	1.600.000	k-epsilon Std	0,02	41%	64%
RANS 47	3	1	3	0,1	2.000.000	k-epsilon Std	0,02	39%	59%
RANS 48	3	1	2	0,1	410.000	k-epsilon Std	0,02	49%	53%
RANS 49	3	0	2	0,1	420.000	k-epsilon Std	0,02	48%	52%

Figura 4.20: Configuración de los casos RANS con premezcla en CONVERGE.

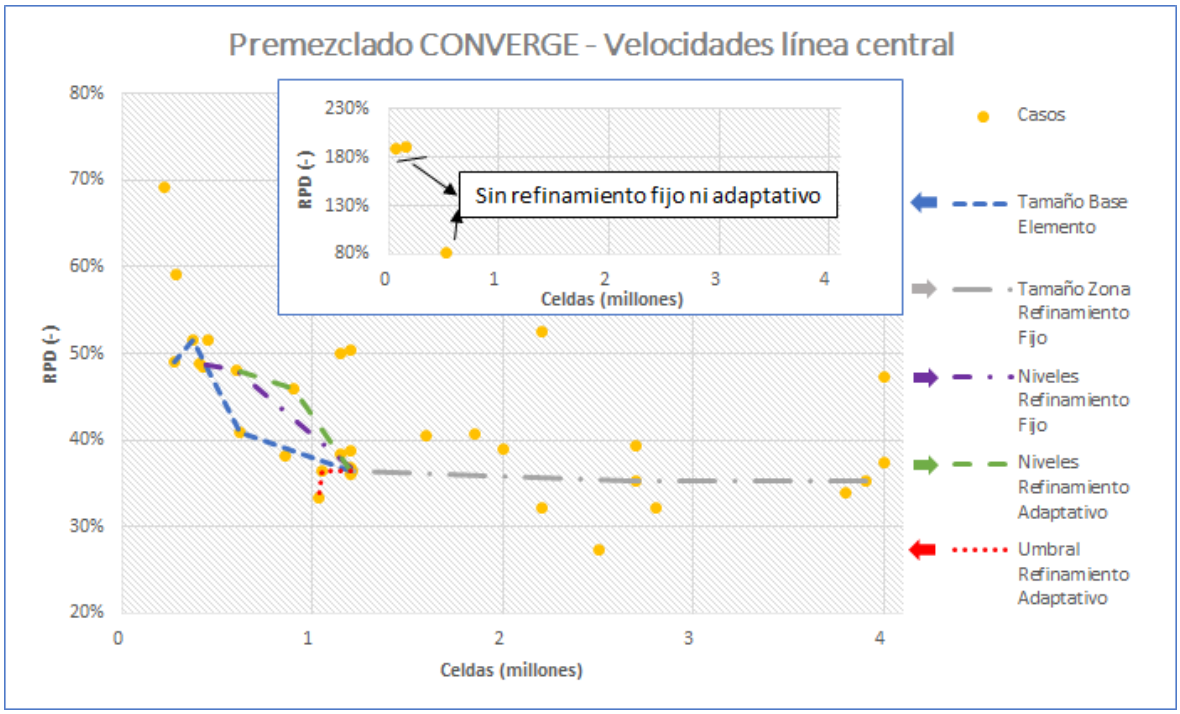


Figura 4.21: RPD de la velocidad en la línea central en función del número de celdas en la malla de CONVERGE.

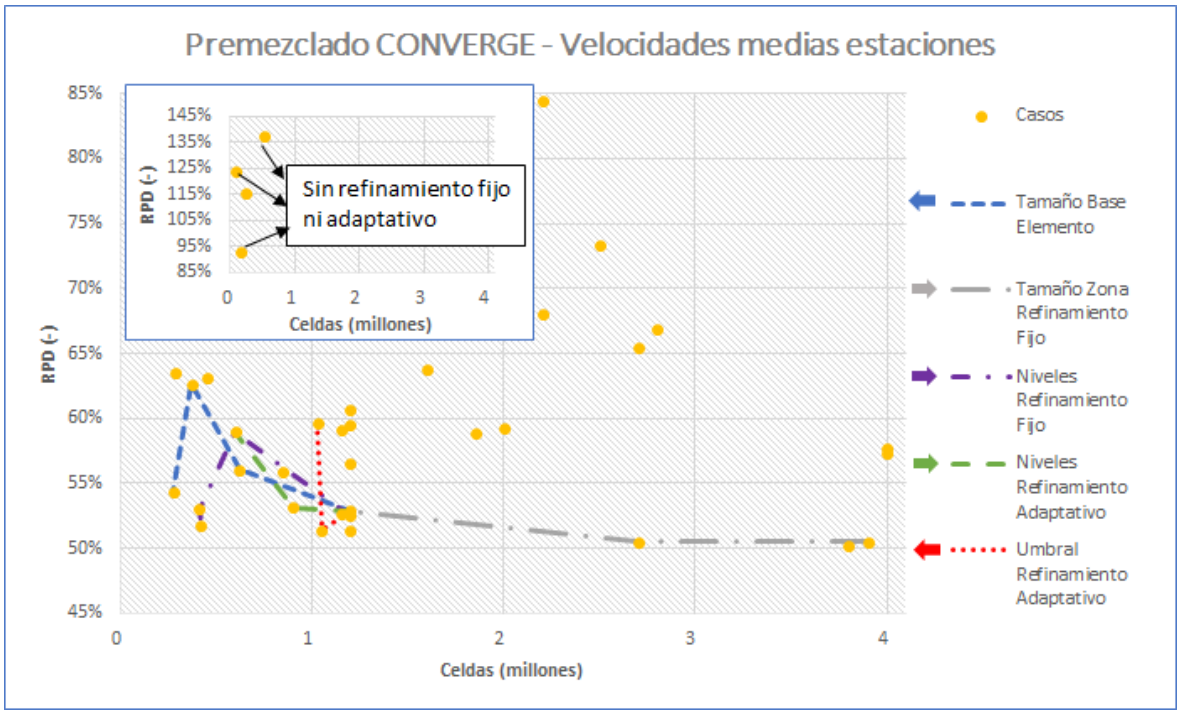


Figura 4.22: RPD de la velocidad en las estaciones de la entrada a la cámara de combustión en función del número de celdas en la malla de CONVERGE.

- El tamaño base de elemento es inversamente proporcional al número de celdas. Por tanto, a menor tamaño más millones de celdas. En ambas gráficas la tendencia es la esperable, aumenta la precisión con la cantidad de celdas.
- El tamaño de la zona de refinamiento fijo es proporcional al número de celdas, si la zona a refinar es más grande el aumento de celdas es mayor. Sin embargo, la precisión se ve muy poco afectada por este parámetro en relación al número de celdas que aumenta, y por tanto, al tiempo de simulación.
- Los niveles de refinamiento fijo y adaptativo siguen la misma tendencia, cuanto mayor es el nivel mayor es el número de celdas. Sin embargo no crece tanto como en el caso anterior, y el aumento de precisión es más notable.
- El umbral para el refinamiento adaptativo es inversamente proporcional al número de celdas. Si es más grande, se refina menos y se obtienen menos millones de celdas. La diferencia en la desviación es prácticamente inapreciable en la gráfica.

Ambas gráficas muestran una zona plana en general salvo pequeñas excepciones con una desviación excesiva. Como la pendiente no es demasiado grande (excepto en las mallas menos pesadas) la mejor malla en CONVERGE se obtiene entre 0.6-1.2 millones de celdas.

Capítulo 5

Resultados

En este capítulo se interpretarán los resultados de las simulaciones más significativas del caso premezclado para ambos programas de CFD, tanto CONVERGE como OpenFOAM. Tras esta comparación, se estudiará el caso no premezclado únicamente con resultados de CONVERGE debido a que se disponía de tiempo y recursos limitados.

5.1. CASO PREMEZCLADO

Las condiciones del caso premezclado han sido expuestas en el apartado 4.6 para el estudio de independencia de malla, pero deben ser analizadas detenidamente y no sólo mediante aproximación RANS, sino también mediante LES. La información que se extraiga de este caso será de gran ayuda para analizar el caso siguiente.

5.1.1. Simulaciones En OpenFOAM

A continuación se dispone una comparativa entre el error que se obtiene de los modelos de turbulencia RANS y LES utilizados en el software OpenFOAM. Las figuras 5.1-5.3 muestran respectivamente las velocidades en la línea central, las velocidades medias en las estaciones y las RMS en las mismas.

La figura 5.1 indica que, para mallas de peso reducido, de todos los modelos de turbulencia se obtienen resultados similares. En este caso destaca el modelo $k-\varepsilon$, que tiene menor error que el resto. Sin embargo, a partir de los 3 millones de celdas se estabiliza y pasa a ser el peor modelo con bastante diferencia sobre los demás. Los otros modelos, a medida que aumenta el número de celdas, continúan disminuyendo el error de forma significativa; no se alcanza una convergencia de malla porque son modelos URANS/LES.

En la figura 5.2 se observa que todos los modelos tienen resultados similares. De nuevo, como en la figura anterior, en mallas reducidas el modelo $k-\varepsilon$ obtiene resultados ligeramente mejores, pero a partir de los 6 millones de celdas se diferencia del resto,

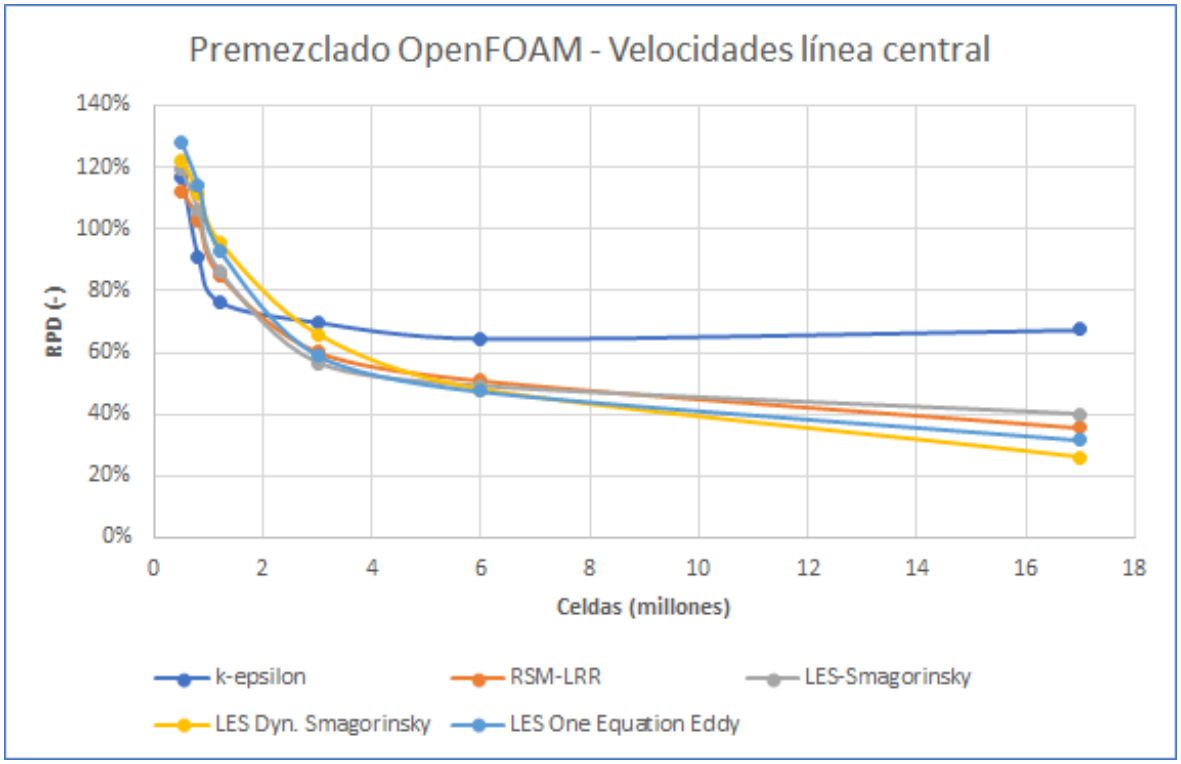


Figura 5.1: Caso premezclado en OpenFOAM, comparación modelos de turbulencia RANS y LES mediante las velocidades en la línea central.

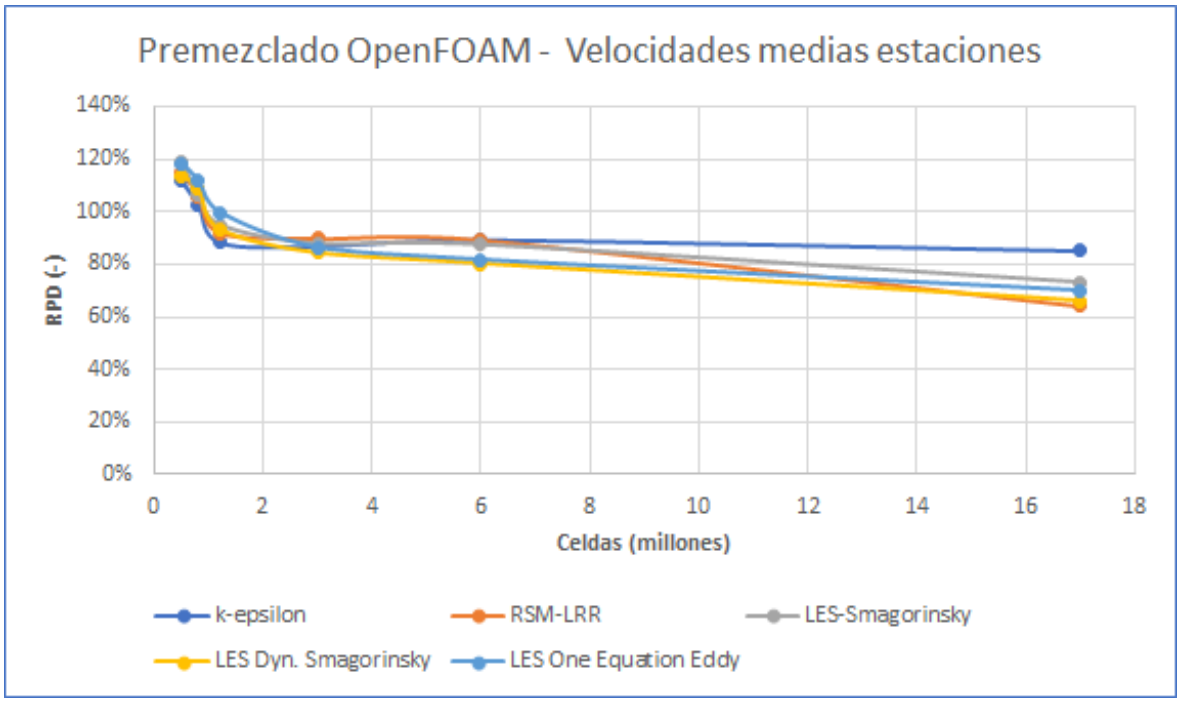


Figura 5.2: Caso premezclado en OpenFOAM, comparación modelos de turbulencia RANS y LES mediante las velocidades medias en las estaciones.

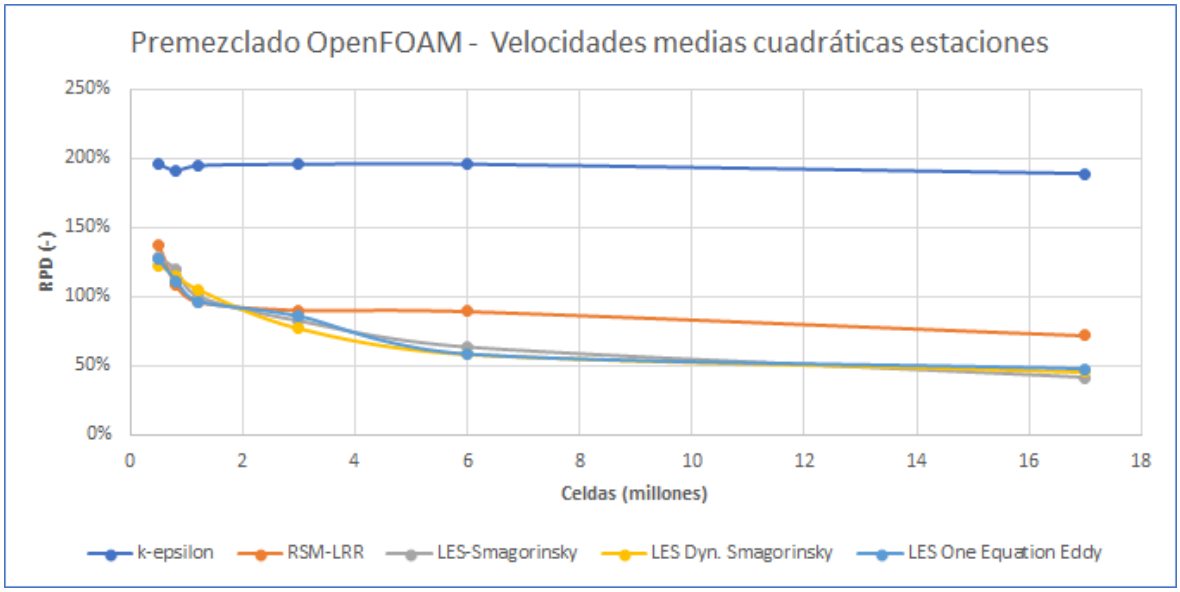


Figura 5.3: Caso premezclado en OpenFOAM, comparación modelos de turbulencia RANS y LES mediante las RMS en las estaciones.

habiendo alcanzado convergencia de malla. Los demás modelos, conforme aumenta el número de celdas, en esta ocasión también disminuyen el error pero con menor impacto. En función de si prima la precisión o la rapidez de simulaciones se escogerá el modelo Dynamic Smagorinsky con 17 millones de celdas o $k-\varepsilon$ con entre 1 y 2 millones.

Al contrario de las anteriores, la figura 5.3 pone de manifiesto que el modelo $k-\varepsilon$ se aleja mucho al referirse a las RMS del resultado experimental para todas las mallas, algo esperable al ser una aproximación RANS. No obstante, el Reynolds Stress Model (que es URANS) da un resultado muy cercano a los LES con mallas de hasta 3 millones de celdas. A partir de los 6 millones, los tres modelos LES son muy similares, y no existen grandes diferencias en el error con mallas más pesadas como la de 17 millones. En caso de encontrarse con limitaciones de capacidad de cálculo, lo ideal sería utilizar 6 millones de celdas y el modelo LES más sencillo: Smagorinsky. Esta sería la opción a escoger en caso de querer investigar otros fenómenos que dependan de la precisión en las RMS.

Aunque las futuras simulaciones deban realizarse con las indicaciones que se han ido dando en este apartado (por la capacidad computacional y tiempo de cálculo limitados), para dibujar los perfiles y contornos se han escogido los casos más precisos de los ya simulados, tratando de analizar las estructuras fluctuantes:

- Entre las aproximaciones RANS, el número 18: con 17 millones de celdas y modelo de turbulencia RSM-LRR (URANS).
- Entre las aproximaciones LES, el número 17: con 17 millones de celdas y modelo de turbulencia Dynamic Smagorinsky.

5.1.2. Simulaciones En CONVERGE

Una de las disyuntivas que plantean las simulaciones es la inicialización del dominio: puede suponerse que todo es aire antes de empezar a introducir la mezcla de aire y combustible, o partir de que el dominio entero es mezcla. Según la figura 5.4 todos los resultados son muy parecidos, tanto en la línea central (cuya diferencia del 1 % es inapreciable) como en las estaciones (medias y RMS), y demuestra que es mejor con la mezcla. Además, en cuanto a los tiempos de simulación, se llegaba más rápido a una solución de este modo.

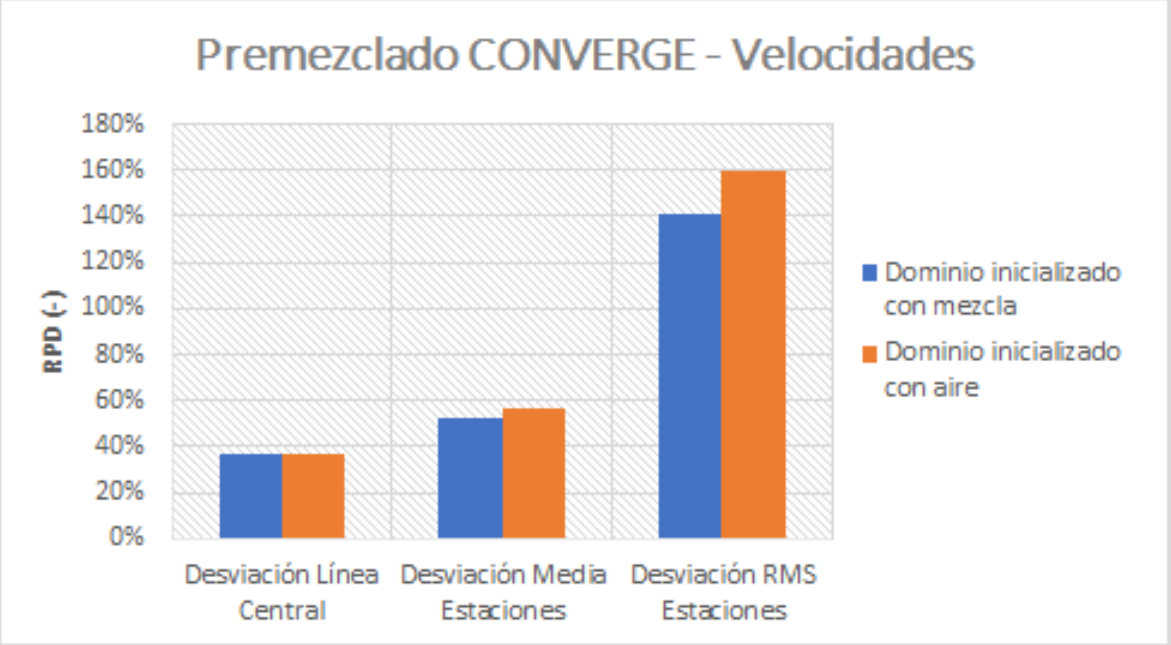


Figura 5.4: Caso premezclado en CONVERGE, inicialización del dominio.

El siguiente aspecto importante es averiguar el modelo de turbulencia para cerrar las ecuaciones RANS que mejores resultados da. El modelo $k-\omega$ fue descartado en simulaciones previas, por lo que en la figura 5.5 se ve una comparación entre RSM-LRR y los tres modelos $k-\varepsilon$: standard, RNG y realizable. Dependiendo de la variable a la que se le deba dar mayor importancia según el problema, conviene elegir un modelo concreto. Si importa el cálculo de velocidades medias en las estaciones (como ocurre en el caso premezclado), el que mejor resultado da es $k-\varepsilon$ standard; pero en lo que a velocidades de la línea central y las RMS en las estaciones se refiere, desempeña mejor papel el modelo RSM-LRR (importante en el caso no premezclado).

Entre las herramientas del mallado adaptativo explicadas en el apartado 4.2.2 se encuentra el *fixed embedding*, que permite refinar la malla en un lugar específico del dominio. Además de utilizarla en el propio swirler, se ha propuesto aplicarla a tres geometrías de cilindros distintas cuyo origen es la entrada de la cámara de combustión,

y penetran en la misma con una longitud y diámetro determinados (figura 5.6). Con ello se pretende capturar mejor las variables en esa zona.

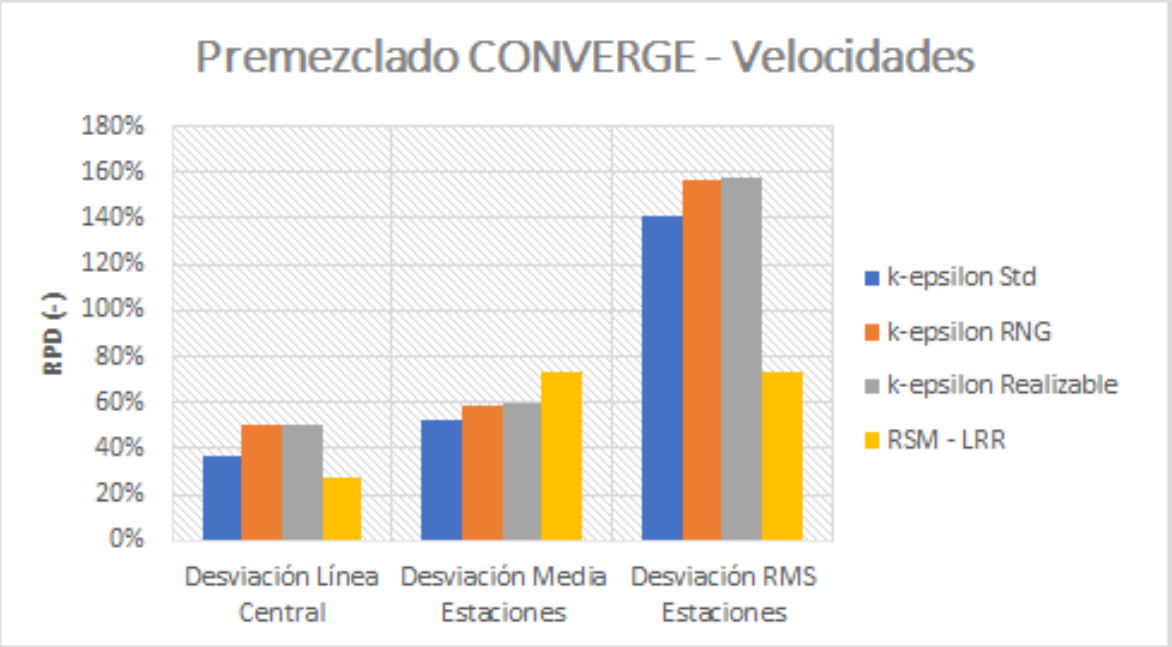


Figura 5.5: Caso premezclado en CONVERGE, modelo de turbulencia RANS.

- El primer tipo de cilindro mide $1cm$ de largo y tiene un diámetro de $6.8cm$. En la leyenda se muestra como C1_6.8 y con él la malla que se obtiene es de 2.7 millones de celdas.
- El segundo tipo de cilindro mide $2cm$ de largo y tiene un diámetro de $6.8cm$. En la leyenda se muestra como C2_6.8 y con él la malla que se obtiene es de 3.9 millones de celdas.
- El tercer tipo de cilindro mide $1cm$ de largo y tiene un diámetro de $3cm$. En la leyenda se muestra como C1_3 y con él la malla que se obtiene es de 1.2 millones de celdas.

La figura 5.7 refleja la comparativa entre los tres tipos. En cuanto a los dos primeros no hay diferencia apreciable de desviación en línea central o en media de estaciones, por lo que merece la pena un cilindro de longitud $1cm$, refinando así la malla en un lugar reducido y evitando que aumente el tiempo de simulación por la diferencia en la cantidad de celdas. Una vez descartado el segundo, la comparativa entre el resto de cilindros muestra muy poca diferencia, aunque favorable para el de mayor diámetro. Sin embargo, no merece la pena duplicar el número de celdas y, por tanto, disparar el tiempo de simulación por un cambio mínimo de la desviación, menor al 4 %. Además,

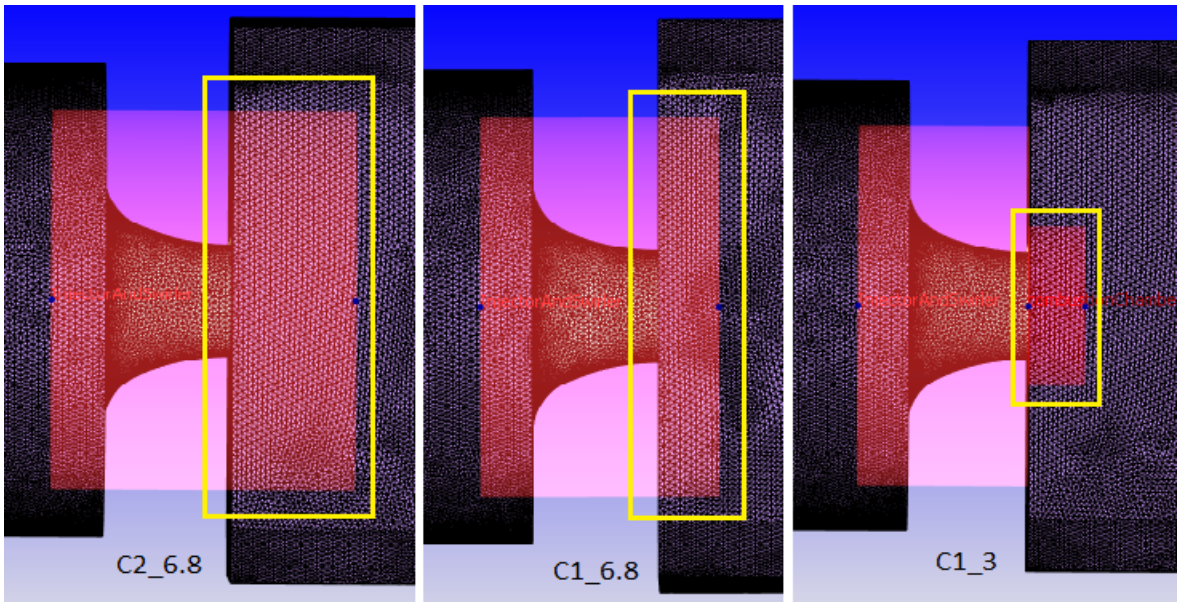


Figura 5.6: Cilindros donde se emplea *fixed embedding*.

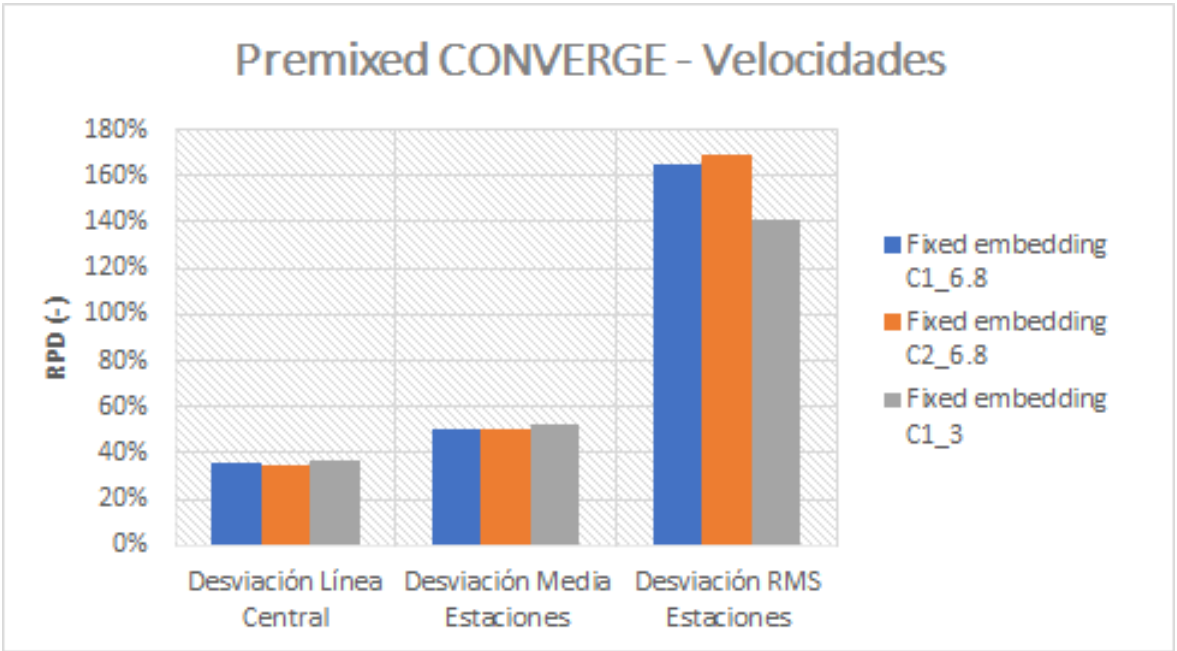


Figura 5.7: Caso premezclado en CONVERGE, *fixed embedding*.

en lo referente a las RMS sí que se reduce el error notablemente con el cilindro C1_3, aunque sigue siendo alto.

A continuación, para el caso estudiado mediante LES también es necesario ver qué modelo de turbulencia se ajusta más a los resultados experimentales. En la figura 5.8 se ve una comparación entre los tres modelos: Dynamic Smagorinsky (DSGS), Dynamic Structural (DSEM) y Smagorinsky-Lilly (SGS). En las velocidades de la línea central y las estaciones, el modelo Dynamic Structural se desvía demasiado respecto de los otros dos que dan resultados muy similares, por lo elegir uno u otro se debe decidir desde el punto de vista del tiempo de simulación. Sin embargo, en cuanto a las RMS, todos los modelos tienen unos resultados muy similares entre ellos, siendo ligeramente mejor el modelo Dynamic Smagorinsky.

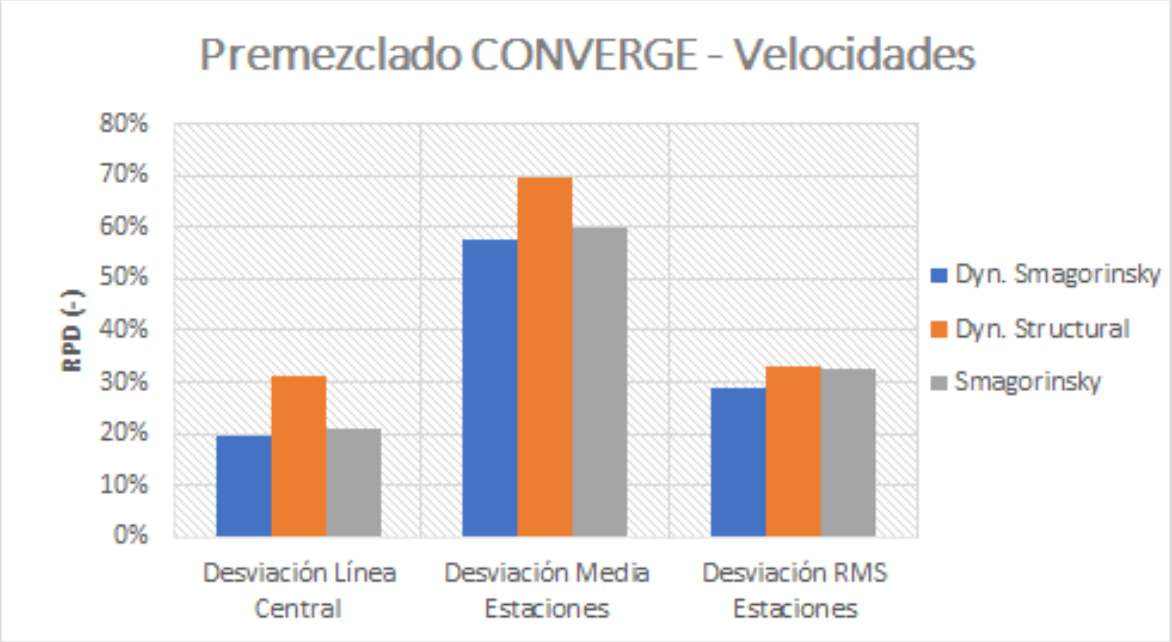


Figura 5.8: Caso premezclado en CONVERGE, modelo de turbulencia LES.

En esta ocasión, para las gráficas de perfiles y contornos se han seleccionado de nuevo dos casos, pero el primero sigue las indicaciones del apartado en cuanto a minimizar el tiempo de simulación (con los modelos turbulentos y tamaños de malla adecuados) mientras que el segundo se ha elegido por ser el más preciso para captar las estructuras transitorias:

- Entre los modelos RANS, el número 17: tamaño base de $3mm$, 3 niveles de *fixed embedding* y *AMR*, 1.2 millones de celdas y modelo de turbulencia $k-\varepsilon$ std.
- Entre los modelos LES, el número 3: tamaño base de $2mm$, 3 niveles de *fixed embedding* y *AMR*, 17 millones de celdas y modelo de turbulencia Dynamic Smagorinsky.

5.1.3. Perfiles De Velocidades

En este apartado se muestran los perfiles de velocidades de los casos de simulación elegidos durante este mismo capítulo. En primer lugar se sitúan las gráficas que comparan directamente los perfiles de las velocidades en todas sus componentes con los datos tomados experimentalmente, para después elegir el más preciso y plasmar la cámara de combustión con sus contornos de velocidades.

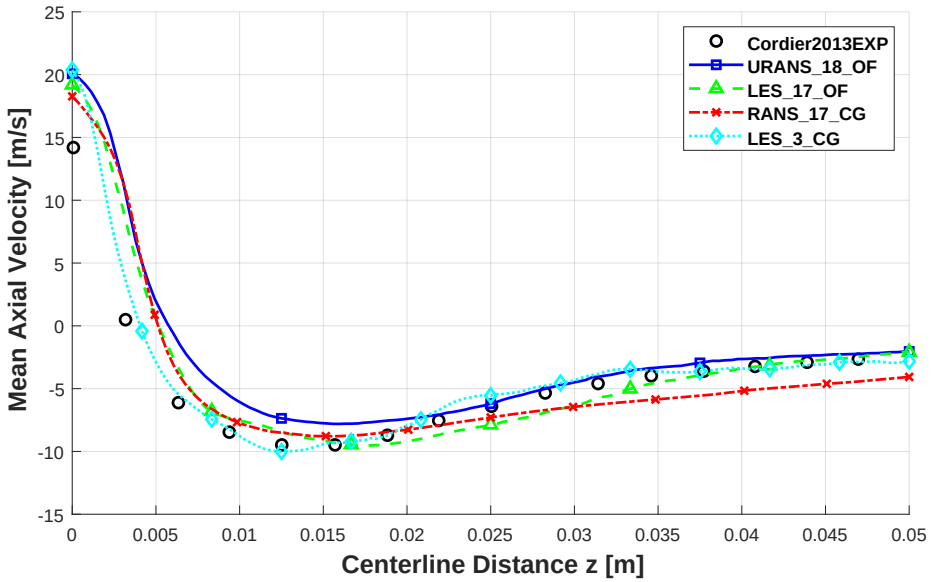


Figura 5.9: Caso premezclado, comparación de velocidades axiales en la línea central entre los mejores modelos de turbulencia y los diferentes códigos CFD con los datos experimentales.

La figura 5.9 muestra los resultados de velocidad media axial en la línea central. De todos los casos, el más cercano a los datos experimentales es el LES de CONVERGE, sin embargo hay una pequeña particularidad: todos ellos sobreestiman la velocidad en las primeras distancias, especialmente en el origen. Esto puede deberse a que el modelo de pared en el inyector no actúa correctamente (los y^+ se encuentran en la capa buffer, zona de mayor error al modelar), dando como resultado una mayor velocidad axial (en torno a $20m/s$) a la salida del mismo (en el centro de la entrada a la cámara). En cualquier caso, los autores de las medidas experimentales reportaron dificultades para realizar las mediciones en esa zona [70, 77, 78, 79].

En las figuras 5.10-5.12 se ven las tres componentes axial, radial y tangencial respectivamente de las velocidades medias (a la izquierda) y las RMS (a la derecha) en función de la distancia radial tomada desde la línea central de la cámara en cada estación.

En primer lugar, la velocidad media axial es sobreestimada por todos los modelos en las dos primeras estaciones, mientras que en el resto ocurre solamente con los modelos

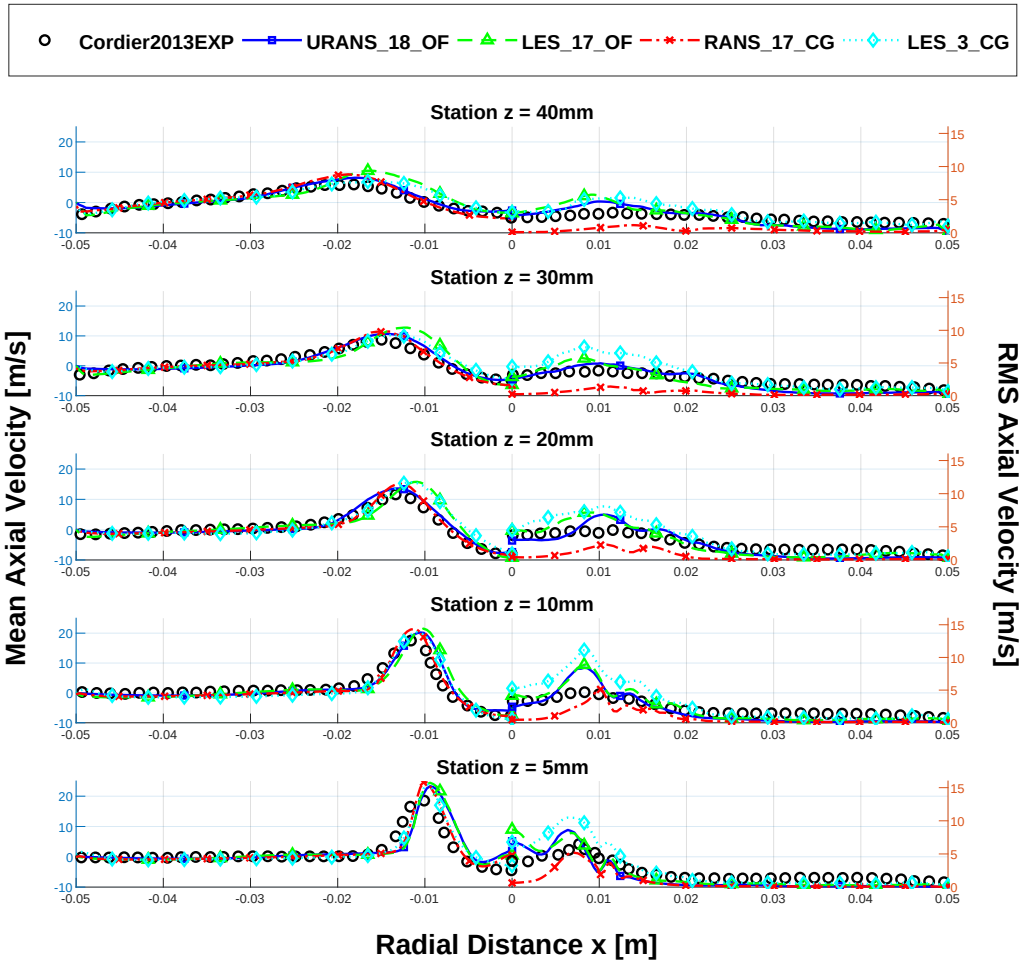


Figura 5.10: Caso premezclado, comparación de velocidades axiales en las estaciones entre los mejores modelos de turbulencia y los diferentes códigos CFD con los datos experimentales.

LES y cerca de la zona central. Esto indica que la zona de recirculación que se mostrará en los contornos de simulación es más estrecha que la experimental, y ocurre acorde a lo explicado en la figura anterior: existe una velocidad axial mayor de la esperada en la entrada. En cuanto a las RMS, como ya se ha adelantado previamente en este capítulo, el modelo RANS_17_CG da los peores resultados subestimando los valores. En el resto de modelos llama la atención que alrededor de la zona $0.01m$ de distancia radial exista un pico que los datos experimentales no reflejan.

En segundo lugar, la velocidad media radial se ajusta casi perfectamente en todas las estaciones y modelos. Llama la atención que en la única discrepancia que existe (en la primera estación) el modelo que mejor se adapta es el RANS_17_CG. Sin embargo, este mismo modelo vuelve a dar resultados lejanos de los experimentales en las RMS

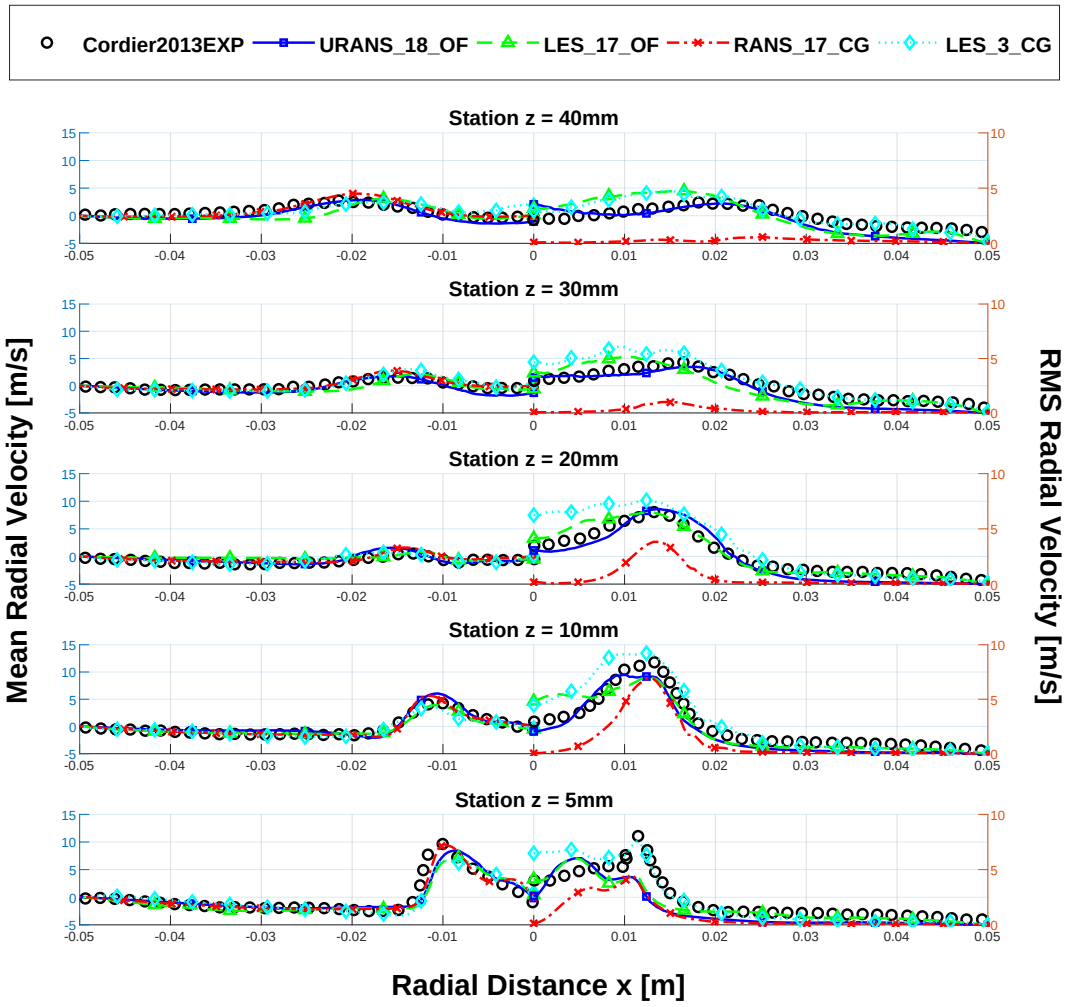


Figura 5.11: Caso premezclado, comparación de velocidades radiales en las estaciones entre los mejores modelos de turbulencia y los diferentes códigos CFD con los datos experimentales.

en comparación al resto, aunque los resultados obtenidos por ellos tampoco logran capturar correctamente las estructuras turbulentas cerca del centro.

Finalmente, la velocidad media tangencial se ajusta mejor según los modelos RANS_17_CG y URANS_18_OF. Igual que ocurría con la velocidad media axial, ésta se ve sobreestimada (mayor valor absoluto, la velocidad es negativa). Esto ocurre también en las primeras estaciones, en la zona cercana al centro de la entrada a la cámara de combustión. Para las RMS existen ciertas diferencias en la tendencia de todos los modelos, de nuevo no se han logrado capturar correctamente las estructuras turbulentas, aunque aparentemente el LES_3_CG es el que mejor se ajusta.

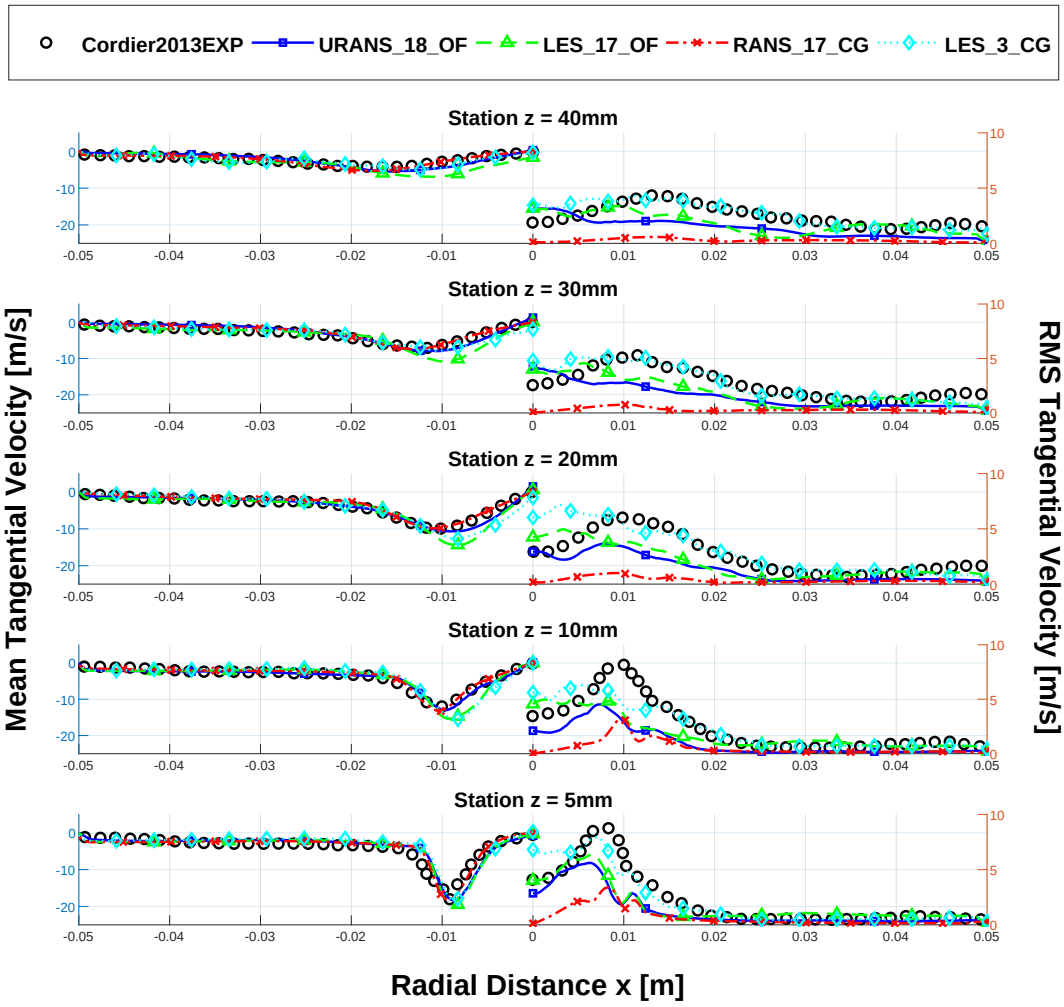


Figura 5.12: Caso premezclado, comparación de velocidades tangenciales en las estaciones entre los mejores modelos de turbulencia y los diferentes códigos CFD con los datos experimentales.

5.1.4. Contornos De Velocidad Y Energía Cinética Turbulenta

A continuación se representan los contornos de velocidades y de energía cinética turbulenta (tke) en uno de los 4 casos seleccionados. La diferencia entre ellos no es muy significativa, sin embargo se adjuntan el resto de contornos en el anexo A.

En la figura 5.13, la columna izquierda muestra las velocidades medias, mientras que la derecha muestra las RMS, y cada componente (axial, radial y tangencial) se corresponde con una fila. No es casualidad que las velocidades radial y tangencial sean prácticamente simétricas, ya que el chorro proveniente del swirler tiene forma de torbellino o espiral como resultado de las fuerzas centrífugas inducidas por la rotación

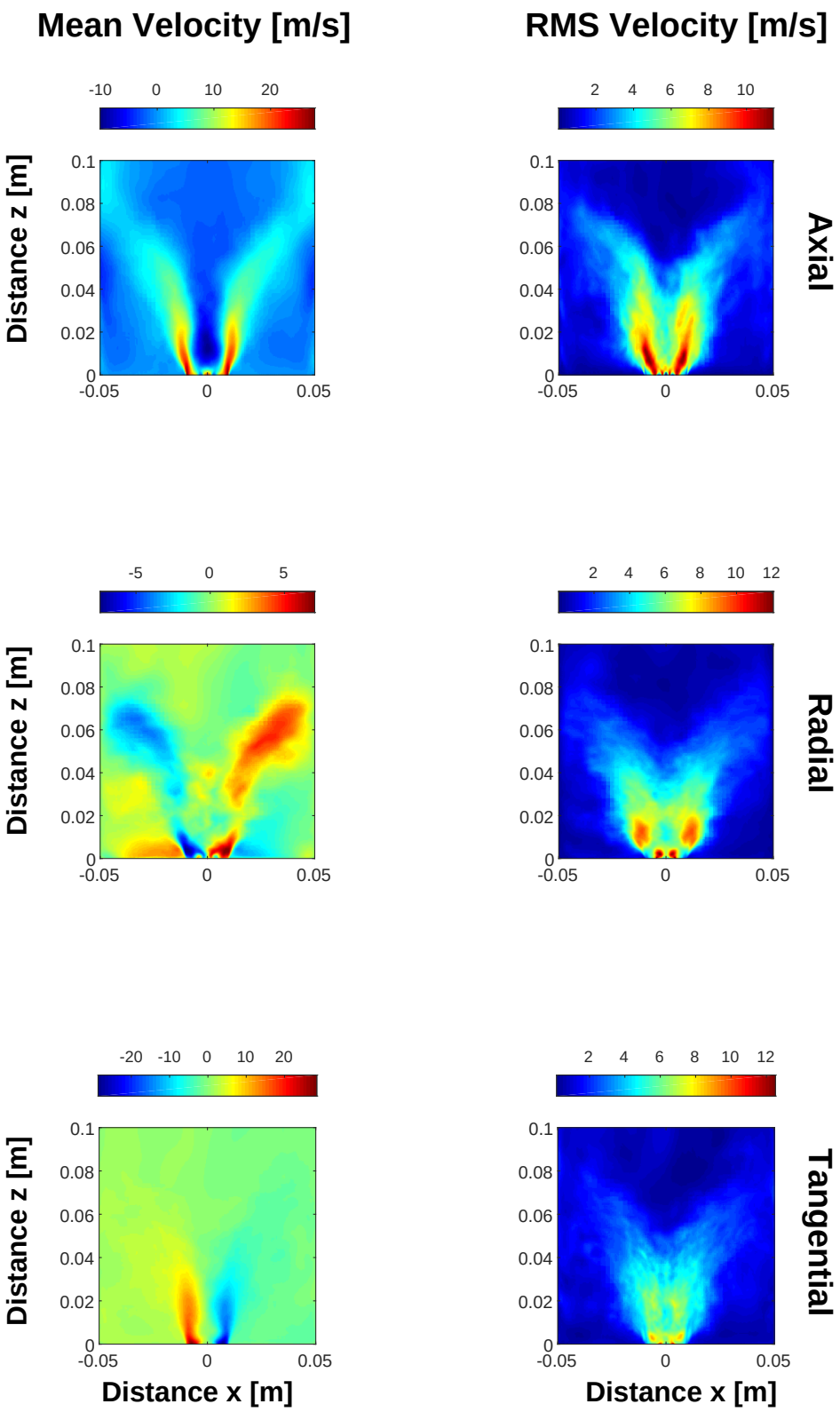


Figura 5.13: Contornos de velocidades del caso premezclado LES_3_CG.

del flujo. Además, esta estructura (descrita en 4.1) que se muestra en la componente axial de velocidades medias, es de gran importancia por su función estabilizadora de llama gracias al bloqueo aerodinámico formado. Su presencia dota al flujo del tiempo de residencia, temperatura y turbulencia adecuadas para una atomización, evaporación, mezclado y combustión eficientes.

Tras mostrar los contornos de velocidades llega el turno de la turbulencia. La figura 5.14 representa el contorno de la energía cinética turbulenta obtenido con las rutinas de posprocesado en Matlab. En estos términos, la zona cercana al punto de remanso presenta una alta turbulencia debido las inestabilidades locales del flujo que provocan un movimiento del punto alrededor de su localización media [83].

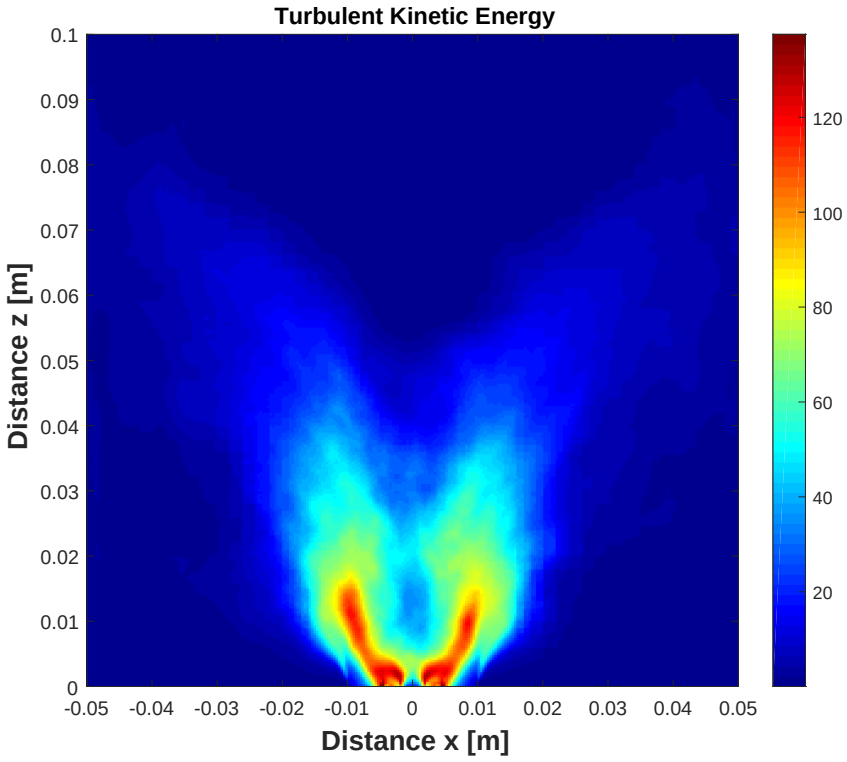


Figura 5.14: Contorno de energía cinética turbulenta del caso premezclado LES_3_CG.

Otro método para identificar los vórtices es el Q-criterion [84], o segundo invariante del tensor gradiente de velocidades. Separando las partes simétrica S y antisimétrica Ω del tensor, el Q-Criterion se establece de la siguiente manera:

$$\nabla v = S + \Omega \quad \text{donde} \quad S = \frac{1}{2}(\nabla v + \nabla v^T), \quad \Omega = \frac{1}{2}(\nabla v - \nabla v^T) \tag{5.1}$$

Para visualizar los vórtices se obtiene una isosuperficie de valores $Q > 0$ obtenidos de la ecuación 5.2. Esto se observa en la figura 5.15 obtenida con el programa de posprocesado EnSight, y ocurre porque las isosuperficies con Q positivas aislan áreas donde

la intensidad de rotación de las estructuras se sobrepone a la deformación, hallando así las envolturas de los vórtices.

$$Q = \frac{1}{2}(|\Omega|^2 - |S|^2) \tag{5.2}$$

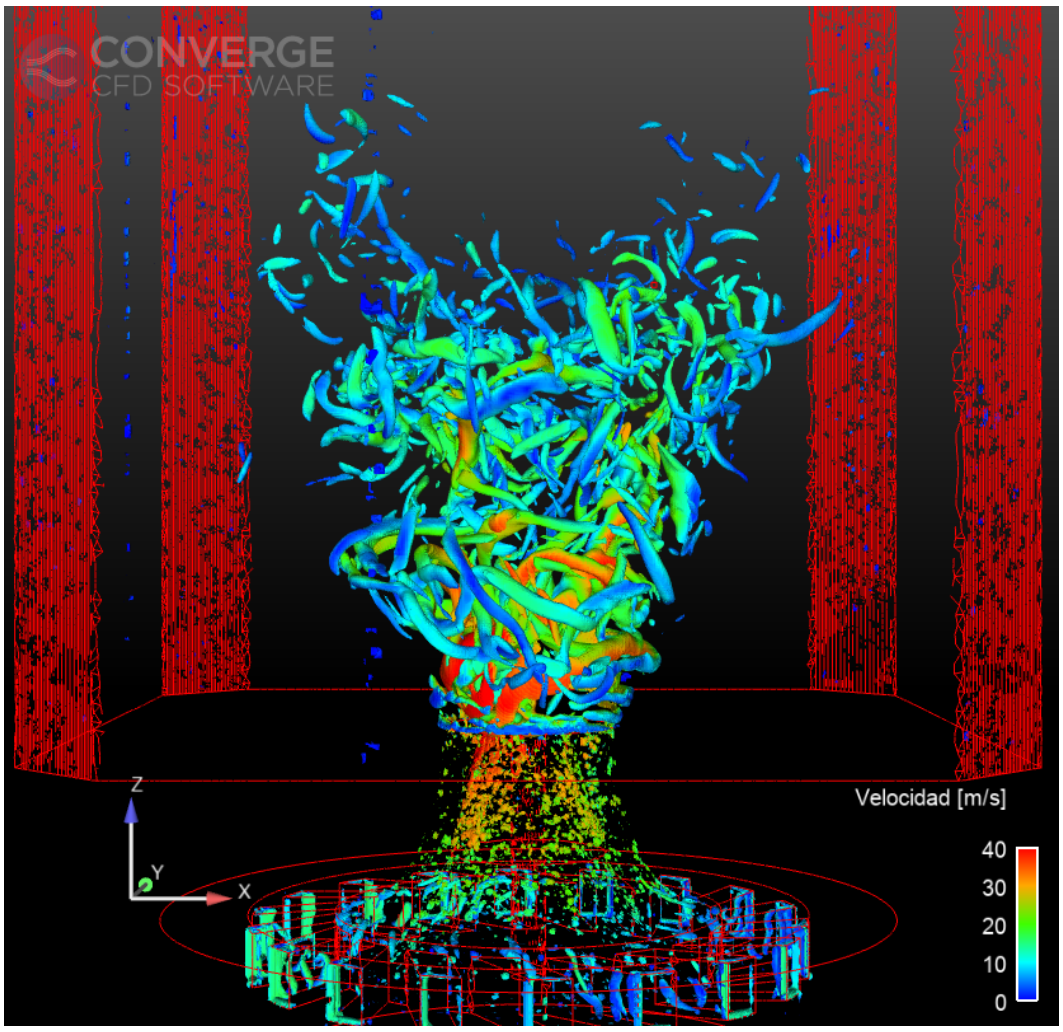


Figura 5.15: Visualización de vórtices turbulentos mediante Q-Criterion obtenida en EnSight.

5.2. CASO NO PREMEZCLADO

En el caso anterior (sección 5.1) se ha extraído información útil con la cual encarar el resto de simulaciones del siguiente, como el tamaño de la malla debido a los niveles de *fixed embedding* y *AMR*. De esta manera se puede continuar con el estudio del flujo no premezclado en la cámara de combustión LDI.

5.2.1. Simulaciones En CONVERGE

Anteriormente se concluyó que en CONVERGE era beneficioso inicializar el dominio con la mezcla de aire y combustible porque mejoraba la precisión de resultados y

ayudaba a una mayor rapidez en la convergencia de los cálculos. No obstante, en el caso no premezclado parece más legítimo que en el plenum y el swirler solamente haya aire, por lo que se ha realizado una comparación entre esas dos formas de inicializar el dominio.

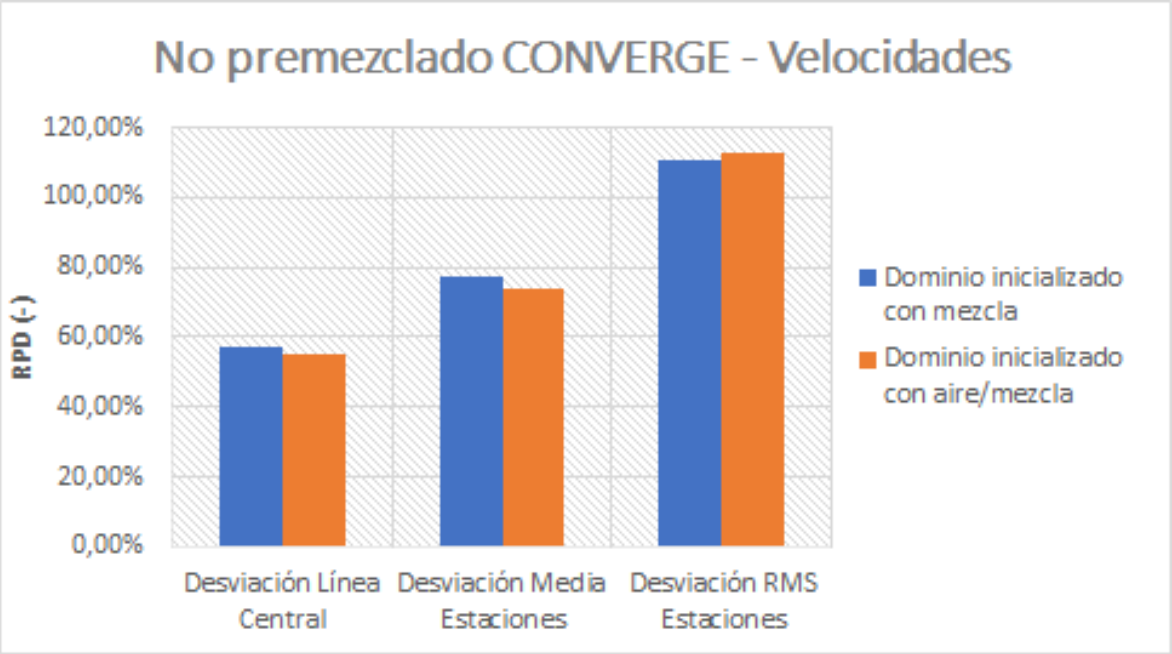


Figura 5.16: Caso no premezclado en CONVERGE, velocidades según inicialización del dominio.

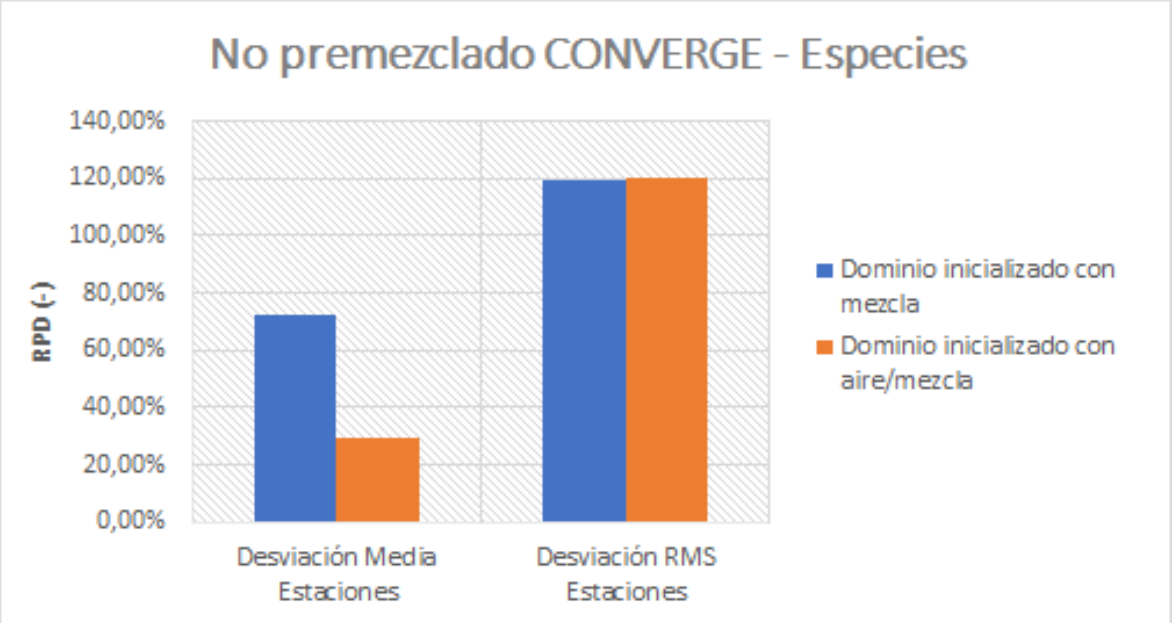


Figura 5.17: Caso no premezclado en CONVERGE, especies según inicialización del dominio.

En la gráfica de las velocidades (figura 5.16) se advierte que no hay grandes cambios entre un modelo u otro, y es favorable para el dominio inicializado con aire y mezcla

excepto para las RMS. Por otra parte, la gráfica de especies (figura 5.17) refleja unas RMS similares y una diferencia muy importante en la media. Esta última razón hace que sea imperioso elegir el dominio inicializado con aire en plenum y swirler, ya que además no supone un perjuicio apreciable en otros aspectos como el tiempo de simulación.

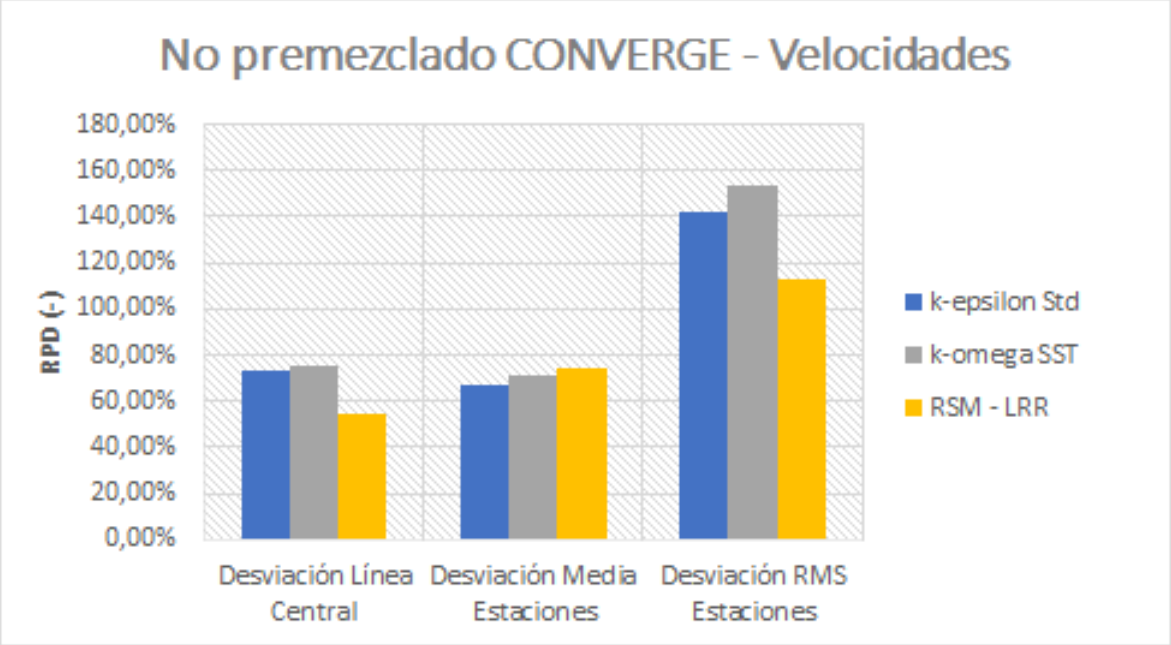


Figura 5.18: Caso no premezclado en CONVERGE, velocidades según modelo de turbulencia RANS.

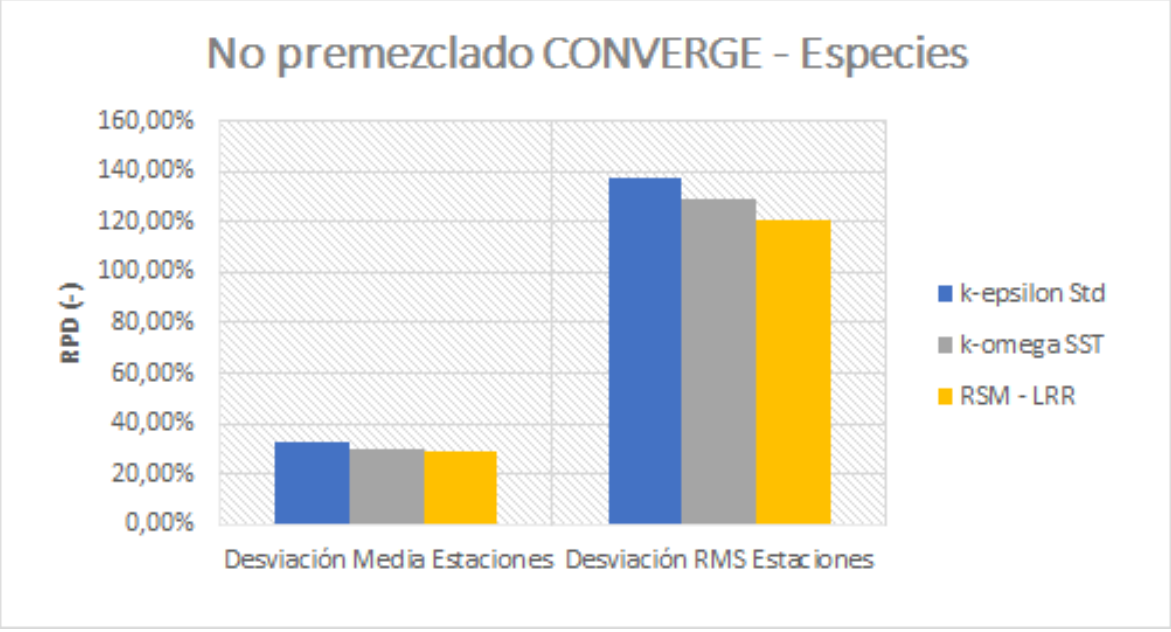


Figura 5.19: Caso no premezclado en CONVERGE, especies según modelo de turbulencia RANS.

De nuevo se ha realizado una comparación entre modelos de turbulencia para la aproximación RANS como se observa en las figuras 5.18-5.19. El modelo $k-\omega$ fue

descartado porque se obtenían resultados similares pero generaba el doble de elementos en la malla, con el consiguiente retardo en el tiempo de simulación. Entre los otros dos modelos, la gráfica de velocidades muestra que las medias tienen un resultado muy similar, ligeramente mejor para $k - \varepsilon$ standard, que podría ser útil por su corto tiempo de simulación. Sin embargo, para simulaciones en las que es esencial conocer las RMS con precisión (futuros estudios de inyección de combustible líquido, atomización y combustión) el modelo RSM-LRR es la mejor opción, dando además una mayor precisión en la línea central. En cuanto a la gráfica de concentración de especies, no aporta nada nuevo: las desviaciones son muy parecidas, y el modelo RSM-LRR tiene más precisión.

5.2.2. Perfiles De Velocidades Y Especies

En este apartado se muestran los perfiles de velocidades y especies del caso de simulación elegido en el apartado anterior, plasmando las gráficas que comparan directamente los perfiles en todas sus componentes con los datos tomados experimentalmente. Además, se incluyen resultados de las simulaciones llevadas a cabo en [76] mediante su código propio AVBP, para comparar con los resultados obtenidos en el presente trabajo. Se sirven de aproximaciones LES, las mallas utilizadas son de hasta 31 millones de elementos y, a diferencia de este estudio, aplican condiciones de no deslizamiento en pared.

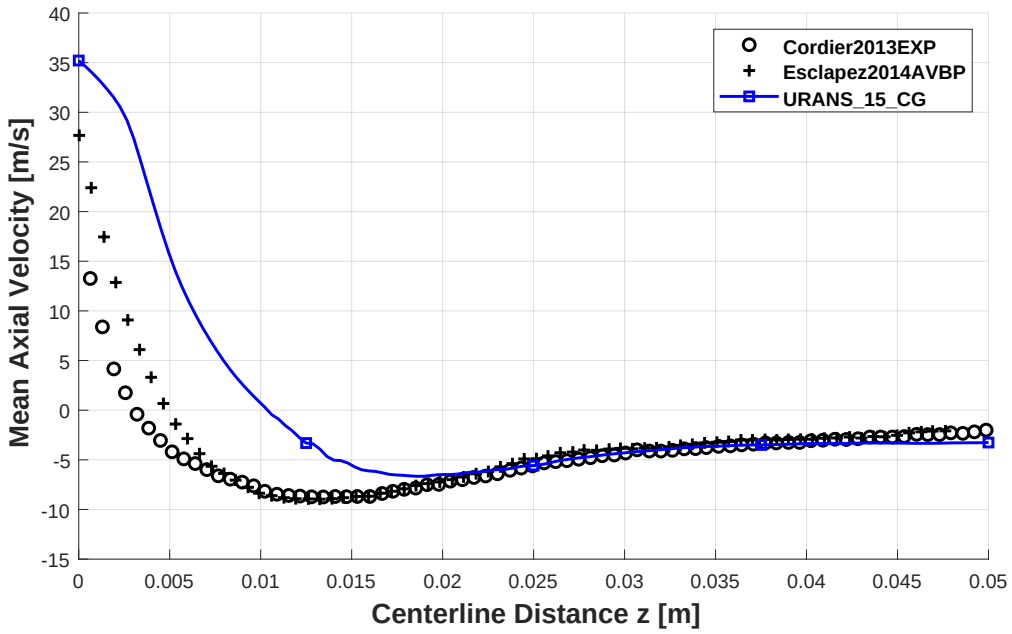


Figura 5.20: Caso no premezclado, comparación de velocidades axiales en la línea central entre el mejor modelo RANS y los datos experimentales.

La figura 5.20 muestra los resultados de velocidad media axial en la línea central. A partir de los $15mm$ se asemejan mucho los datos de la simulación con los de la medición, pero en la zona más cercana existe una diferencia bastante grande. Esto se debe, en parte, a los problemas que ocasionó la medición experimental en estos puntos, que ya ocurrían en el caso premezclado pero se ven magnificados por la densidad del combustible. En este sentido, el código Esclapez2014AVBP da unos resultados más acordes con la velocidad teórica, y se parece más al caso experimental desde los $5mm$.

En las figuras 5.21-5.23 se ven las tres componentes axial, radial y tangencial respectivamente de las velocidades medias (a la izquierda) y las RMS (a la derecha) en función de la distancia radial tomada desde la línea central de la cámara en cada estación.

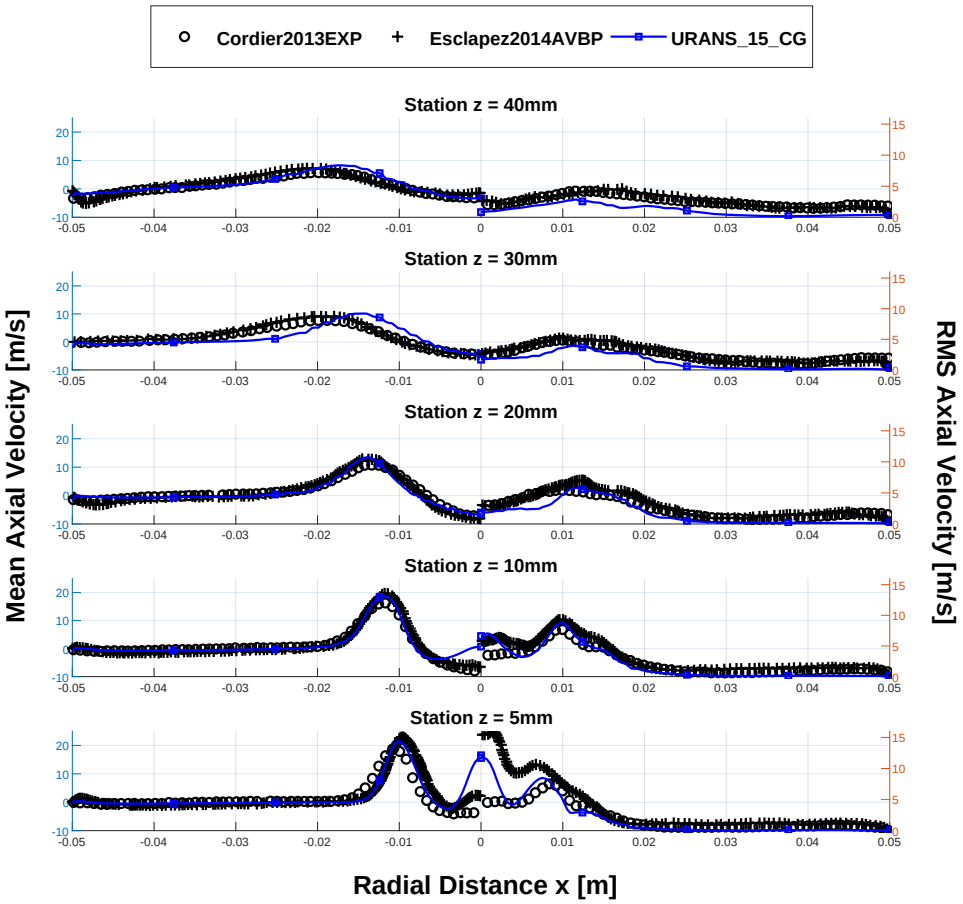


Figura 5.21: Caso no premezclado, comparación de velocidades axiales en las estaciones entre el mejor modelo RANS y los datos experimentales.

En primer lugar, la velocidad media axial se ve sobreestimada en el centro de la geometría, generando un pico que se corresponde con el chorro central de combustible

debido a su mayor densidad. A partir de $20mm$ ya se asemejan más los resultados, pero la zona de recirculación es más estrecha de lo que debería ser. Ocurre lo mismo con los resultados de Esclapez2014AVBP, pero con un pico menos pronunciado y adaptándose en una estación anterior ($10mm$). Las RMS del caso URANS_15_CG consiguen acercarse de manera aceptable a la tendencia de los resultados experimentales, exceptuando la misma zona ya nombrada, y subestimando las medidas en las últimas estaciones, mientras que el AVBP los sobreestima siempre. Aunque los resultados de AVBP son algo más ajustados, los conseguidos en este proyecto no tienen nada que envidiar teniendo en cuenta que hay una gran diferencia en el coste computacional.

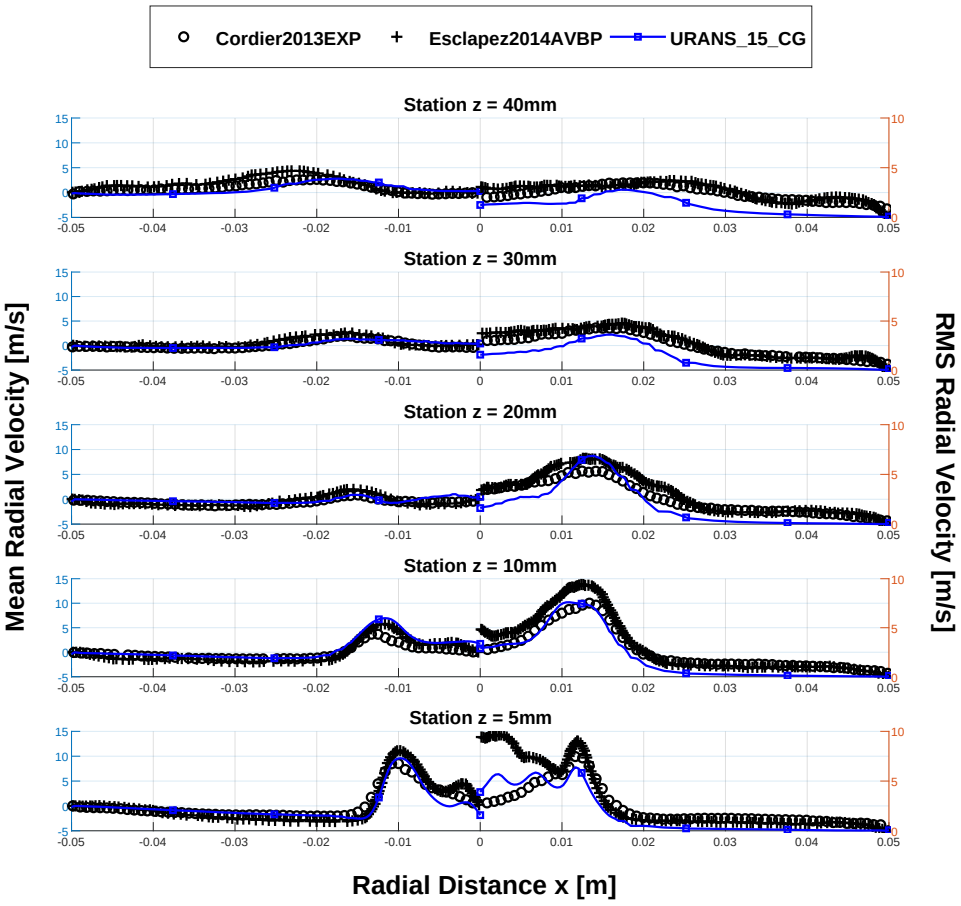


Figura 5.22: Caso no premezclado, comparación de velocidades radiales en las estaciones entre el mejor modelo RANS y los datos experimentales.

En segundo lugar, la velocidad media radial de ambos códigos CFD se ajusta casi perfectamente en todas las estaciones. En cuanto a las RMS, de nuevo los resultados del código AVBP son más precisos que los obtenidos en CONVERGE, los primeros sobreestimando y los segundos subestimando los valores en relación a los resultados

experimentales. Cabe destacar que la tendencia de la gráfica en la primera estación es capturada mejor por el caso URANS_15_CG.

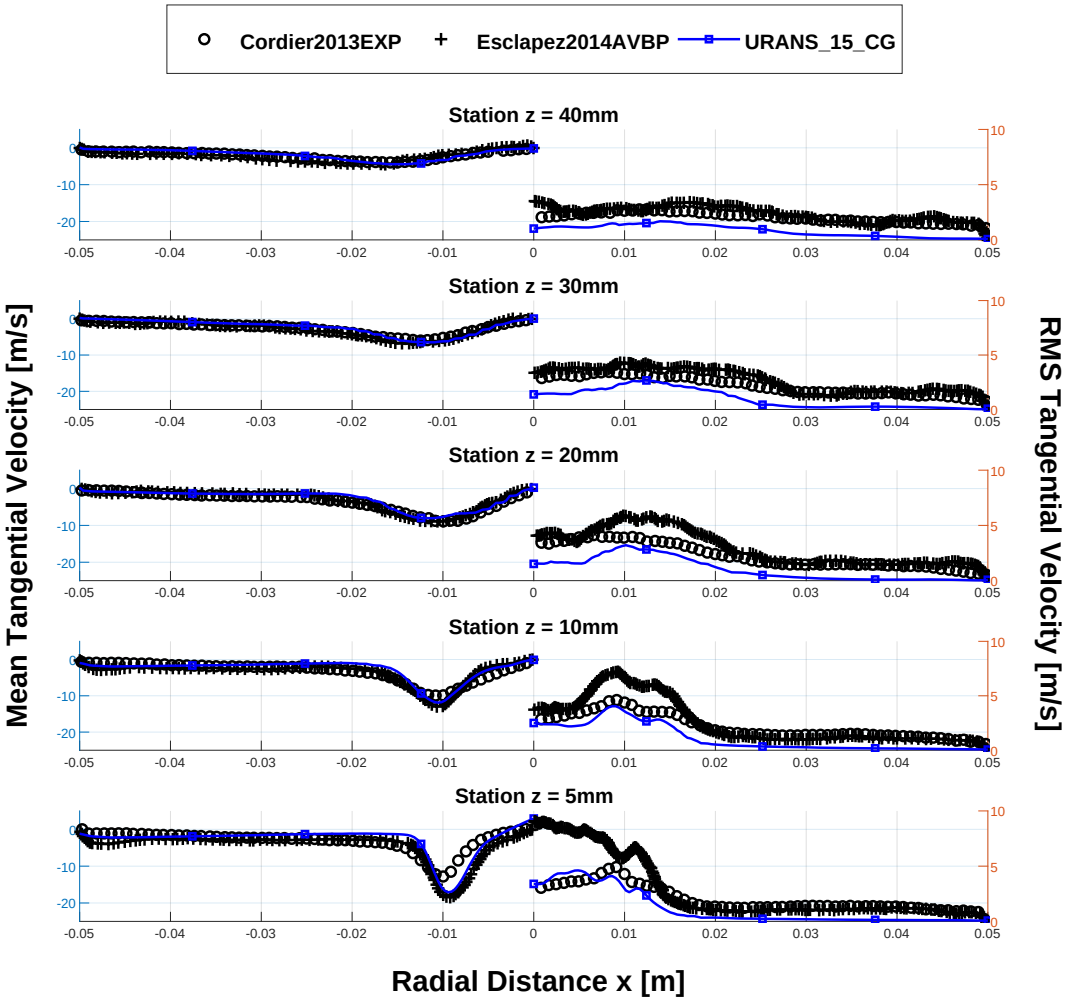


Figura 5.23: Caso no premezclado, comparación de velocidades tangenciales en las estaciones entre el mejor modelo RANS y los datos experimentales.

A continuación, la velocidad media tangencial de ambos códigos CFD también se ajusta casi perfectamente en todas las estaciones. Incluso en la zona de $5mm$, donde hay una pequeña discrepancia respecto de los resultados experimentales, coinciden los resultados de ambas simulaciones. Para las RMS, los resultados de las primeras estaciones del caso URANS_15_CG se corresponden mucho mejor con los experimentales, pero a partir de los $20mm$ terminan subestimándolas. momento en el cual el caso Esclapez2014AVBP es más fiel a la tendencia de la gráfica experimental, aunque sobreestime ligeramente los valores.

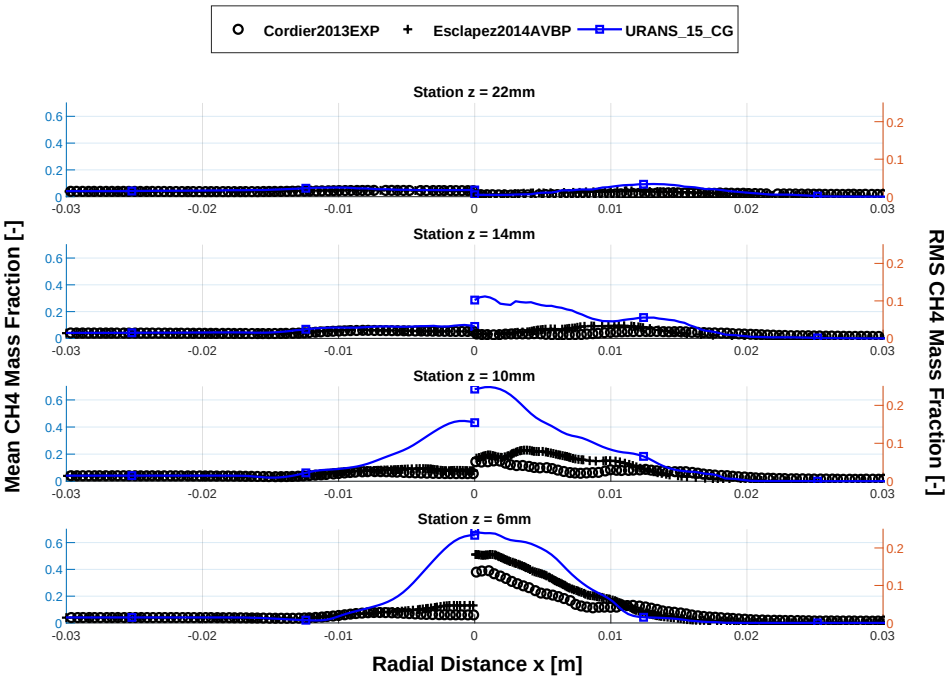


Figura 5.24: Caso no premezclado, comparación de concentraciones en las estaciones entre el mejor modelo RANS y los datos experimentales.

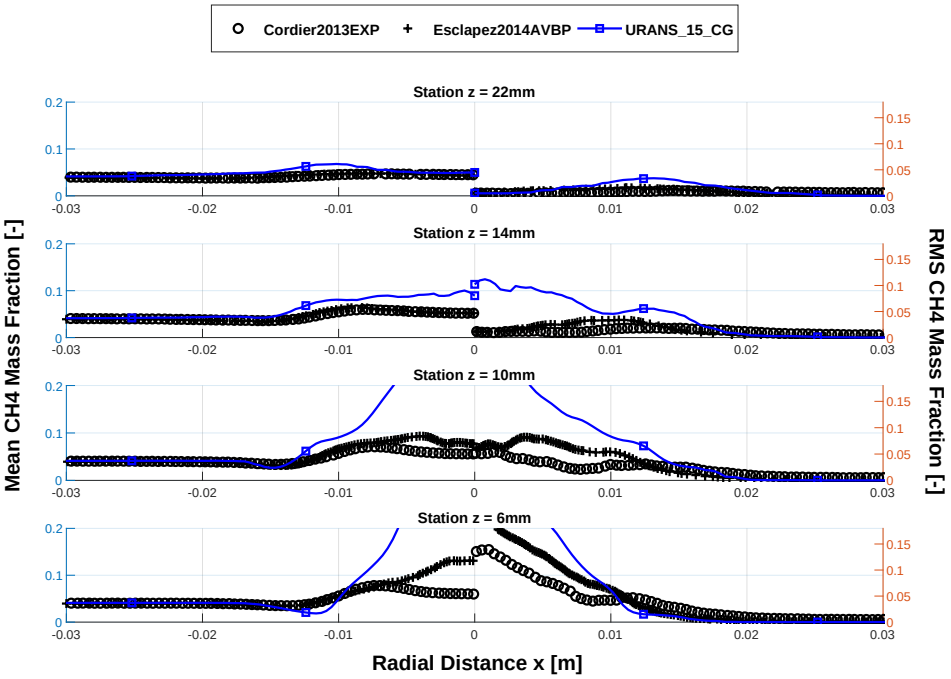


Figura 5.25: Figura 5.24 con zoom en la escala para mejor visualización.

Finalmente, la concentración de combustible mostrada en las figuras 5.24-5.25 se ve muy afectada por las velocidades axiales. Como para el caso URANS_15_CG es mayor de lo esperado hasta los $15mm$ aproximadamente, las concentraciones también lo son de forma exagerada, por lo que se debería tratar de ajustar el modelo para conseguir unas velocidades axiales más acertadas. Por otro lado, el Esclapez2014AVBP tiene mucha mejor precisión en las concentraciones debido al mismo motivo, aunque continúa sobrepredicciendo los valores tanto en concentraciones medias como RMS de las primeras estaciones.

A modo de conclusión es importante destacar que los resultados obtenidos en CONVERGE por aproximaciones RANS tienen un gran potencial, ya que se ajustan a las mediciones experimentales con una precisión más que aceptable en relación a la obtenida por las simulaciones LES en AVBP, con una malla muchísimo más pesada y unos tiempos de simulación excesivamente altos.

5.2.3. Contornos De Velocidad Y Energía Cinética Turbulenta

A continuación se representan los contornos de velocidades y de energía cinética turbulenta (tke) en el caso RANS seleccionado.

En la figura 5.26, la columna izquierda muestra las velocidades medias, mientras que la derecha muestra las RMS, y cada componente (axial, radial y tangencial) se corresponde con una fila.

En la primera gráfica la zona de recirculación central aparece desplazada aguas abajo, y es más débil ya que alcanza valores negativos de menos valor absoluto que el caso reactivo. También representa velocidades más altas en una zona mayor a la salida del inyector, por lo que todavía es necesario ajustar el modelo mejor en cuanto a la componente axial. Las velocidades radial y tangencial siguen siendo simétricas por el flujo torbellinado, y se aproximan en gran medida a los resultados que cabe esperar de dicha estructura. Por otra parte, las RMS axiales tienen valores máximos mayores que el caso premezclado, lo que indica que hay mayor dispersión en la zona de la entrada a la cámara, mientras que las radiales y tangenciales tienen valores máximos menores.

Tras mostrar los contornos de velocidades llega el turno de la turbulencia. La figura 5.27 representa el contorno de la energía cinética turbulenta obtenido con las rutinas de posprocesado en Matlab. Los valores obtenidos son menores que en el caso premezclado, y la zona de más intensidad ha menguado.

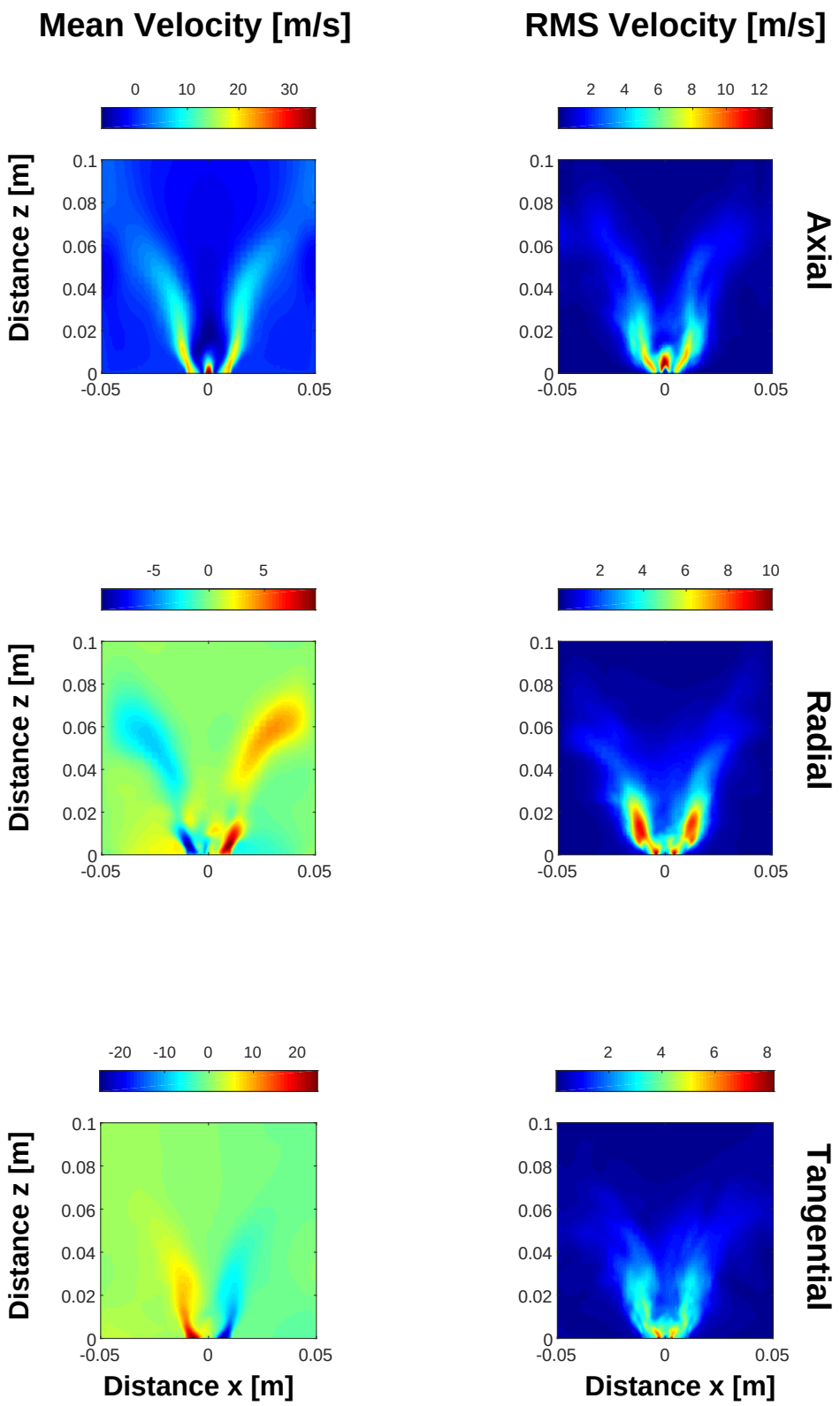


Figura 5.26: Contornos de velocidades del caso no premezclado URANS_15_CG.

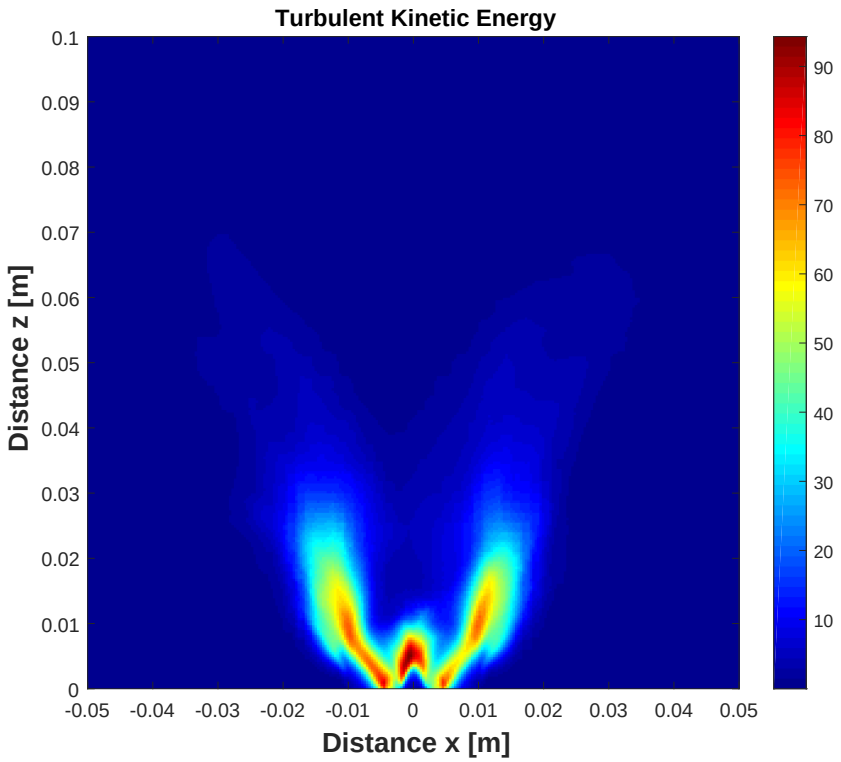


Figura 5.27: Contorno de energía cinética turbulenta del caso no premezclado URANS_15_CG.

Capítulo 6

Conclusiones y trabajos futuros

A fin de cerrar la memoria, se recogen las conclusiones principales a las que se ha llegado a lo largo del Trabajo de Fin de Máster y que pueden ser de utilidad para el desarrollo de trabajos futuros en la misma línea de investigación.

Se ha caracterizado el flujo no reactivo gaseoso premezclado en quemadores LDI mediante tratamientos URANS (Unsteady Reynolds Averaged Navier-Stokes) y LES (Large Eddy Simulation) de la turbulencia empleando dos códigos CFD: CONVERGE con mallado adaptativo, y OpenFOAM con mallado tradicional. Los resultados de ambos códigos han sido contrastados y validados con la bibliografía experimental disponible, quedando de esta manera calibrados los modelos.

Se ha caracterizado el flujo no reactivo gaseoso no premezclado en quemadores LDI mediante tratamientos URANS (Unsteady Reynolds Averaged Navier-Stokes) de la turbulencia empleando CONVERGE con mallado adaptativo. Los resultados han sido contrastados y validados con la bibliografía experimental disponible, allanando el camino para calibrar los modelos.

La estrategia seguida ha consistido en emplear una solución RANS estacionaria como punto de partida realista para las siguientes simulaciones, tanto URANS como LES. Éstas se han mantenido hasta que el flujo ha alcanzado un estado estadísticamente estable, monitorizando para ello el gasto másico a la salida. A partir de ese momento se ha comenzado a calcular los promediados temporales y fluctuaciones de las variables de interés.

Se ha conseguido reproducir satisfactoriamente la topología del flujo no reactivo y caracterizar las principales estructuras turbulentas formadas en el interior del quemador LDI: zona de recirculación central, chorro torbellinado y zona de recirculación de las esquinas. Esto es un avance importante para estudios posteriores de inyección,

atomización de spray líquido, mezcla y combustión, ya que al ser procesos transitorios se ven afectados de forma crítica por estas estructuras.

Las simulaciones del caso premezclado en CONVERGE han reportado muy buenos resultados, haciendo evidente que el método de mallado adaptativo es una herramienta potente capaz de reducir los tiempos de simulación manteniendo una precisión adecuada, a la altura de un código con mallado tradicional como OpenFOAM. Para el caso no premezclado también se han obtenido grandes resultados con aproximación URANS, en comparación a los que ofrecía AVBP (con mayor resolución y aproximación LES), aunque se deja abierta la puerta para explorar una configuración que capte mejor las velocidades axiales y, por ende, las concentraciones de combustible en la zona más cercana a la entrada de la cámara.

Este TFM puede suponer, por tanto, el punto de partida de una investigación con objetivos más ambiciosos, modelando la totalidad de los fenómenos inherentes al proceso de inyección de combustible líquido: atomización, evaporación, mezclado e interacción de las estructuras bifásicas, la combustión y las emisiones contaminantes.

Bibliografía

[1] ICAO. ICAO Environmental Report 2016. *Cambridge University Press*, page 250, 2016.

[2] Philippe Busquin, Pedro Argüelles, Manfred Bischoff, B.A.C. Droste, SirRichardH. Evans, Walter Kröll, Jean-Luc Lagardère, Alberto Lina, John Lumsden, Denis Ranque, Søren Rasmussen, Paul Reutlinger, SirRalph Robins, Helena Terho, and Arne Wittlöw. European aeronautics: a vision for 2020. *Air & Space Europe*, 3(3-4):16–18, 2001.

[3] Joyce E Penner. Aviation and the Global Atmosphere: a special report of IPCC Working Groups I and III. page 373, 1999.

[4] IATA. Fact Sheet Industry Statistics, 2018.

[5] Scott C. Herndon, Joanne H. Shorter, Mark S. Zahniser, David D. Nelson, John Jayne, Robert C. Brown, Richard C. Miake-Lye, Ian Waitz, Phillip Silva, Thomas Lanni, Ken Demerjian, and Charles E. Kolb. NO and NO2 emission ratios measured from in-use commercial aircraft during taxi and takeoff. *Environmental Science and Technology*, 38(22):6078–6084, 2004.

[6] O. Dessens, M. O. Köhler, H. Rogers, R. L. Jones, and J. A. Pyle. Aviation and climate change. *Transport Policy*, 34:14–20, 07 2014.

[7] P. Hoor, J. Borken-Kleefeld, D. Caro, O. Dessens, O. Endresen, M. Gauss, V. Grewe, D. Hauglustaine, I. S. A. Isaksen, P. Jöckel, J. Lelieveld, G. Myhre, E. Meijer, D. Olivie, M. Prather, C. Schnadt Poberaj, K. P. Shine, J. Staehelin, Q. Tang, J. van Aardenne, P. van Velthoven, and R. Sausen. The impact of traffic emissions on atmospheric ozone and oh: results from quantify. *Atmospheric Chemistry and Physics*, 9(9):3113–3136, 2009.

[8] D. W. Bahr, F. Culick, M.V. Heitor, and J.H. Whitelaw. Unsteady combustion. In *Aircraft Turbine Engine NOx Emission Abatement*, pages 243–264. Springer, Dordrecht, 1996.

[9] EASA. Notice of Proposed Amendment 2017-01. Implementation of the CAEP/10 amendments on climate change, emissions and noise. pages 1–338, 2018.

[10] A Seaton, W MacNee, K Donaldson, and D Godden. Particulate air pollution and acute health effects. *Lancet*, 345(8943):176–178, 1995.

[11] Arthur H. Lefebvre and Dilip R. Ballal. *Gas turbine combustion: alternative fuels and emissions*. Taylor & Francis, Boca Raton, FL, 3rd editio edition, 2010.

[12] D. Dewanji. *Flow Characteristics in Lean Direct Injection Combustors*. 2012.

[13] K. K. Rink and A. H. Lefebvre. The influences of fuel composition and spray characteristics on nitric oxide formation. *Combustion Science and Technology*, 68(1-3):1–14, 1989.

[14] G Leonard and S Correa. Second asme fossil fuel combustion symposium. *ASME/PD*, 30:69–74, 1990.

[15] Chi-Ming Lee, Kathleen M. Tacina, and Changlie Wey. High pressure low NOx emissions research: Recent progress at NASA Glenn Research Center. pages 1–8, 2007.

[16] S. Mosier and R. Pierce. Advanced combustion systems for stationary gas turbine engines: Volume i. review and preliminary evaluation., 1980.

[17] G. S. Samuelsen, J. Brouwer, M. A. Vardakas, and J. D. Holdeman. Experimental and modeling investigation of the effect of air preheat on the formation of nox in an rql combustor. *Heat and Mass Transfer*, 49(2):219–231, Feb 2013.

[18] D. W. Bahr. Technology for the design of high temperature rise combustors. *Journal of Propulsion and Power*, 3(2):179–186, Mar 1987.

[19] B. L. Koff. Aircraft gas turbine emissions challenge, 1993.

[20] Zhuohui J. He, Kristin Kopp-Vaughan, Changlie Wey, Clarence T. Chang, Albert K. Cheung, Chi-Ming Lee, and Angela D. Surgenor. chapter Emission Characteristics of A P&W Axially Staged Sector Combustor. AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, Jan 2016. 0.

[21] P. E. Sabla, J. R. Taylor, and D. J. Gauntner. Design and development of the combustor inlet diffuser for the nasa/ge energy efficient engine, 1981.

[22] A. Klein. Characteristics of combustor diffusers. *Progress in Aerospace Sciences*, 31(3):171 – 271, 1995.

[23] J. Cai. *Aerodynamics of lean direct injection combustor with multi-swirler arrays*. PhD thesis, University of Cincinnati, 2006.

[24] H. S. Alkabie, G. E. Andrews, and N. T. Ahmad. Lean low nox primary zones using radial swirlers, 1988.

[25] H. S. Alkabie and G. E. Andrews. Ultra low nox emissions for gas and liquid fuels using radial swirlers, 1989.

[26] H. S. Alkabie and G. E. Andrews. Radial swirlers with peripheral fuel injection for ultra-low nox emissions, 1990.

[27] H. S. Alkabie, G. E. Andrews, and M. N. Kim. An ultra low nox pilot combustor for staged low nox combustion. *11th ISABE-1993-7020*, 1:215–225, 1993.

[28] Waldemar Lazik, Thomas Dörr, and Sebastian Bake. Low nox combustor development for the e3e core engine demonstrator, 09 2007.

[29] Daniel A. Nickolaus, D. Scott Crocker, David L. Black, and Clifford E. Smith. Development of a lean direct fuel injector for low emission aero gas turbines, 2002.

[30] S. W. Shaffar and G. S. Samuelsen. A liquid fueled, lean burn, gas turbine combustor injector. 139:41–57, 10 1998.

[31] T. Terasaki. Lean non-premixed combustion for low-nox gas turbine combustor. *Proceedings of the 1995 Yokohama International Gas Turbine Congress(II)*, pages 353–358, 1995.

[32] Robert Tacina. Low NOx Potential of Gas Turbine Engines. *28th Aerospace Sciences Meeting*, page 20, 1990.

[33] Christopher M. Heath, Robert C. Anderson, Randy J. Locke, and Yolanda R. Hicks. Optical characterization of a multipoint lean direct injector for gas turbine combustors: Velocity and fuel drop size measurements, 2010.

[34] Robert Tacina, Chien-Pei Mao, and Changlie Wey. Experimental investigation of a multiplex fuel injector module with discrete jet swirlers for low emission combustors. 09 2004.

[35] R.R. Tacina, P. Lee, and C. Wey. A lean direct injection combustor using a 9 point swirl-venturi fuel injector, 2005.

[36] D. Dewanji and A. G. Rao. Spray combustion modeling in lean direct injection combustors, part II: Multi-point LDI. *Combustion Science and Technology*, 187(4):558–576, 2015.

[37] A. K. Gupta, D. G. Lilley, and N. Syred. *Swirl flows*. 1984.

[38] J.M. Beér and N.A. Chigier. *Combustion aerodynamics*. Fuel and energy science series. Applied Science Publishers Ltd, 1972.

[39] Erol Kilik. *The influence of swirler design parameters on the aerodynamics of downstream recirculation region*. PhD thesis, 1976.

[40] Robert Tacina, Jun Cai, and San-Mou Jeng. The Structure of a Swirl-Stabilized Reacting Spray Issued From an Axial Swirler. *43rd AIAA Aerospace Sciences Meeting & Exhibit*, (January), 2005.

[41] K. O. Smith, F. R. Kurzynske, and L. C. Angelo. Experimental Evaluation of Fuel Injection Configurations for a Lean-Premixed Low NOx Gas Turbine Combustor. *ASME Journal of Engineering for Gas Turbines and Power*, pages 1–9, 1987.

[42] L. H. Cowell and K. O. Smith. Development of a liquid-fueled, lean-premixed gas turbine combustor, 1992.

[43] S. Etemad and B. C. Forbes. Cfd modeling of a gas turbine combustor air swirler effect, computational fluid dynamics in aero-propulsion, 1995.

[44] Ashraf A. Ibrahim and Milind A. Jog. Nonlinear breakup model for a liquid sheet emanating from a pressure-swirl atomizer. *Journal of Engineering for Gas Turbines and Power*, 129(4):945–953, Jan 2007.

[45] J. Ebner, P. Schober, O. Schafer, R. Koch, and S. Wittig . Modelling of shear-driven liquid wall films: effect of accelerated air flow on the film flow propagation. *Progress in Computational Fluid Dynamics, an International Journal*, 4(3-5):183–190, 2004.

[46] John D Anderson. Computational Fluid Dynamics: The Basics with Applications, 1995.

[47] Hernan Tinoco, Hans Lindqvist, Wiktor Frid, and Kraftgrupp Ab. Numerical Simulation of Industrial Flows. 1990.

[48] Henk Kaarle Versteeg and Weeratunge Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education, 2007.

[49] H. Tennekes and J.L. Lumley. *A First Course in Turbulence*. MIT Press, 1972.

[50] P. Bradshaw, T. Cebeci, and J. H. Whitelaw. Engineering calculation methods for turbulent flow. *NASA STI/Recon Technical Report A*, 82, 1981.

[51] B.E. Launder and B.I. Sharma. Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Letters in Heat and Mass Transfer*, 1(2):131 – 137, 1974.

[52] W.P. Jones and J.H. Whitelaw. Calculation methods for reacting turbulent flows: A review. *Combustion and Flame*, 48:1 – 26, 1982.

[53] B. E. Launder, G. J. Reece, and W. Rodi. Progress in the development of a reynolds-stress turbulence closure, 1975.

[54] Wolfgang Rodi. Turbulent models and their application in hydraulics-a state of art review. 12 1980.

[55] K. Hanjalic. Will rans survive les?: A view of perspectives. *Journal of Fluids Engineering*, 127(5):831–839, Sep 2005.

[56] Stefan H Johansson, Lars Davidson, and Erik Olsson. Numerical simulation of vortex shedding past triangular cylinders at high reynolds number using ak-ε turbulence model. *International Journal for Numerical Methods in Fluids*, 16(10):859–878, 1993.

[57] Sven Perzon and Lars Davidson. On CFD and transient flow in vehicle aerodynamics. *SAE World Congress*, (724):2000–01–0873, 2000.

[58] Gopal Patnaik, Jay P Boris, Theodore R Young, and Fernando F Grinstein. Large scale urban contaminant transport simulations with miles. *Journal of Fluids Engineering*, 129(12):1524–1532, 2007.

[59] Joel H. Ferziger. Large eddy numerical simulations of turbulent flows. *AIAA Journal*, 15(9):1261–1267, Sep 1977.

[60] J. Smagorinsky. General circulation experiments with the primitive equations. *Monthly Weather Review*, 91(3):99–164, 1963.

[61] J.O. Hinze. *Turbulence*. McGraw-Hill classic textbook reissue series. McGraw-Hill, 1975.

[62] Massimo Germano, Ugo Piomelli, Parviz Moin, and William H. Cabot. A dynamic subgrid?scale eddy viscosity model. *Physics of Fluids A: Fluid Dynamics*, 3(7):1760–1765, jul 1991.

[63] S.V Patankar and D.B Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787–1806, 1972.

[64] Pedro Martí Gómez-Aldaraví. *Development of a computational model for a simultaneous simulation of internal flow and spray break-up of the Diesel injection process*. PhD thesis, 2014.

[65] R.I Issa. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, 62(1):40 – 65, 1986.

[66] Hrvoje Jasak. Error analysis and estimation for the finite volume method with applications to fluid flows. 1996.

[67] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, 12(6):620–631, 1998.

[68] J Frenillot. Etude phenomenologique des processus d'allumage et de stabilisation dans les chambres de combustion turbulente swirlees. 2011.

[69] M. Cordier, A. Vandel, G. Cabot, B. Renou, and A. M. Boukhalfa. Laser-induced spark ignition of premixed confined swirled flames. *Combustion Science and Technology*, 185(3):379–407, 2013.

[70] Matthieu Cordier. *Allumage et propagation de flamme dans les écoulements fortement swirlés : études expérimentales et numériques*. PhD thesis, 2013.

[71] C J Greenshields. OpenFOAM, July 2018.

[72] Convergent Science. Converge 2.4 manual, November 2017.

[73] Convergent Science. Converge studio 2.4 manual, December 2017.

[74] Convergent Science. Advanced converge cfd training 2.4 gas turbine combustion, December 2017.

[75] NJDEP. Laboratory Quality Assurance and Quality Control Data Quality Assessment and Data Usability Evaluation. (April), 2014.

[76] Lucas Esclapez. *Numerical study of ignition and inter-sector flame propagation in gas turbine.* PhD thesis, Institut National Polytechnique de Toulouse (INP Toulouse), 2015.

[77] David Barré, Matthias Kraushaar, Gabriel Staffelbach, Vincent Moureau, and Laurent Y.M. Gicquel. Compressible and low Mach number LES of a swirl experimental burner. *3rd INCA Colloquim*, 341(1-2):277–287, 2013.

[78] David Barre. *Simulation Numerique De L'Allumage Dans Les Chambres De Combustion Aeronautiques*. PhD thesis, 2014.

[79] David Barré, Lucas Esclapez, Matthieu Cordier, Eleonore Riber, Bénédicte Cuenot, Gabriel Staffelbach, Bruno Renou, Alexis Vandel, Laurent Y.M. Gicquel, and Gilles Cabot. Flame propagation in aeronautical swirled multi-burners: Experimental and numerical investigation. *Combustion and Flame*, 161(9):2387–2405, 2014.

[80] PAUL BILLANT, JEAN-MARC CHOMAZ, and PATRICK HUERRE. Experimental study of vortex breakdown in swirling jets. *Journal of Fluid Mechanics*, 376:183?219, 1998.

[81] O Lucca-Negro and T O'Doherty. Vortex breakdown: a review. *Progress in Energy and Combustion Science*, 27(4):431 – 481, 2001.

[82] S Leibovich. The structure of vortex breakdown. *Annual Review of Fluid Mechanics*, 10(1):221–246, 1978.

[83] Nicholas Syred. A review of oscillation mechanisms and the role of the precessing vortex core (pvc) in swirl combustion systems. *Progress in Energy and Combustion Science*, 32(2):93 – 161, 2006.

[84] G. Haller. An objective definition of a vortex. *Journal of Fluid Mechanics*, 525:1–26, 2005.

Apéndices

Apéndice A

Perfiles y contornos de velocidades y energía cinética turbulenta

A.1. CASOS PREMEZCLADOS

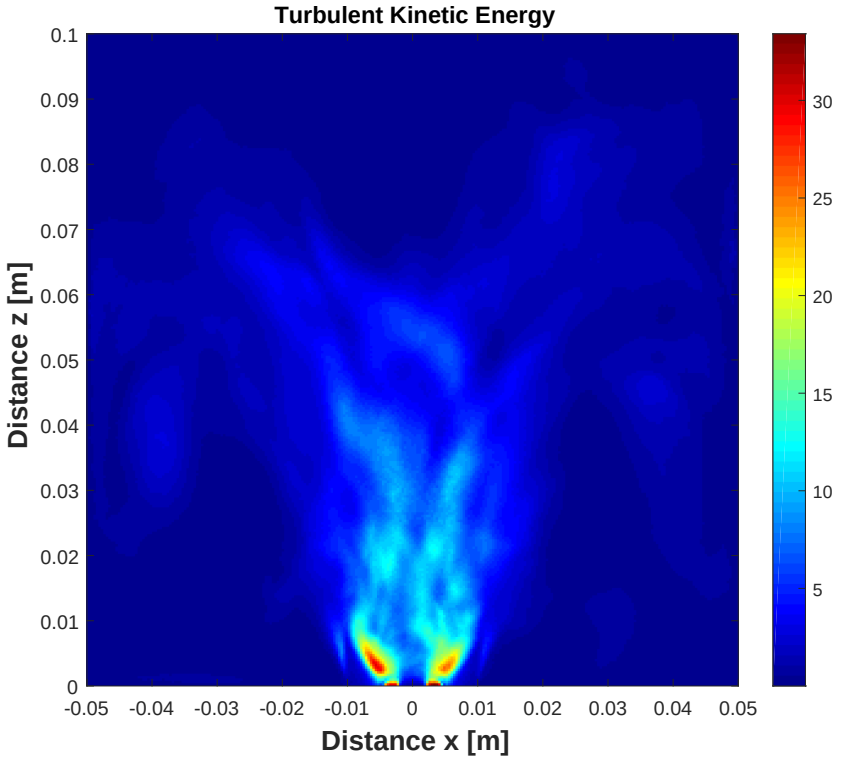


Figura A.1: Contorno de energía cinética turbulenta del caso premezclado LES_17_OF.

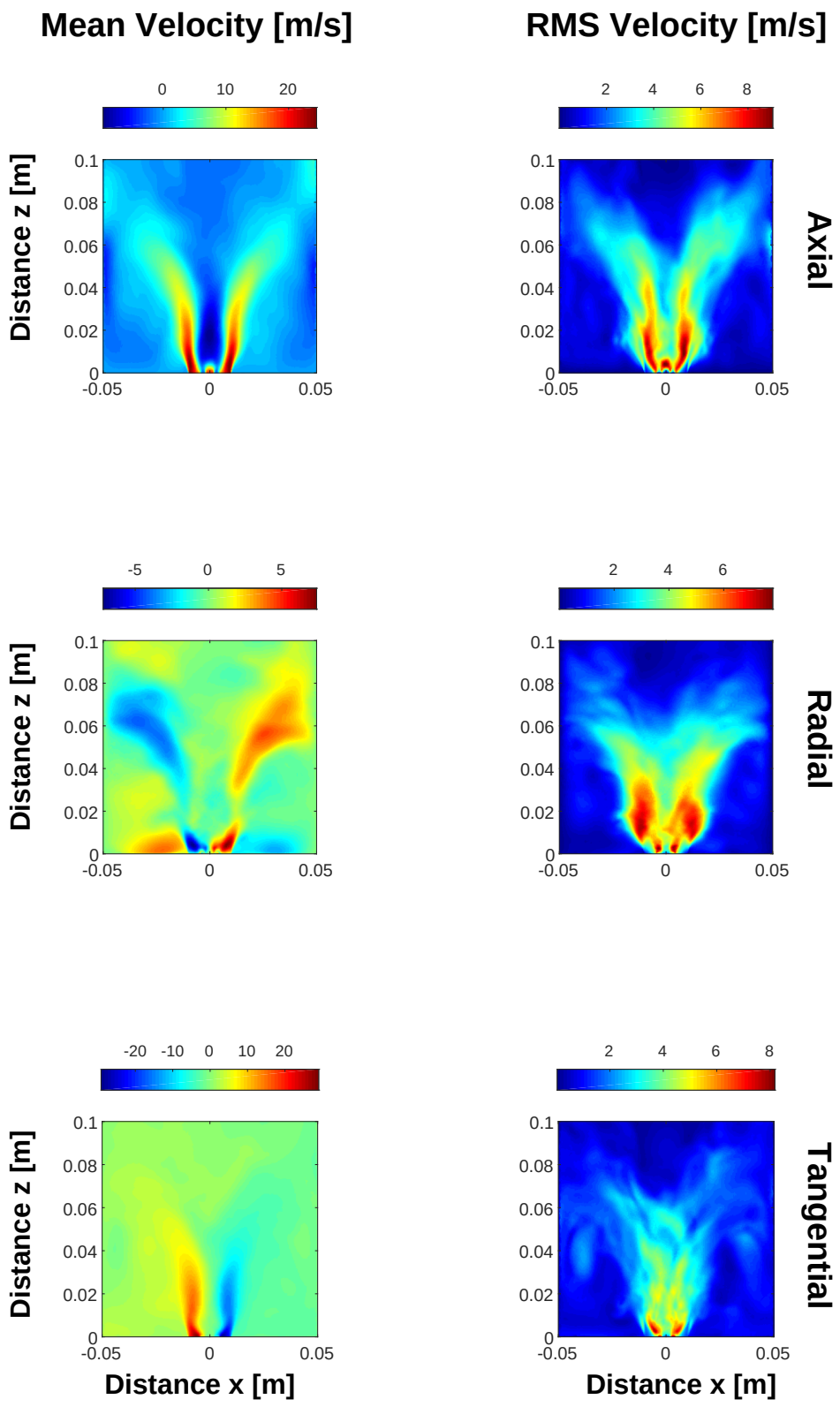


Figura A.2: Contornos de velocidades del caso premezclado LES_17_OF.

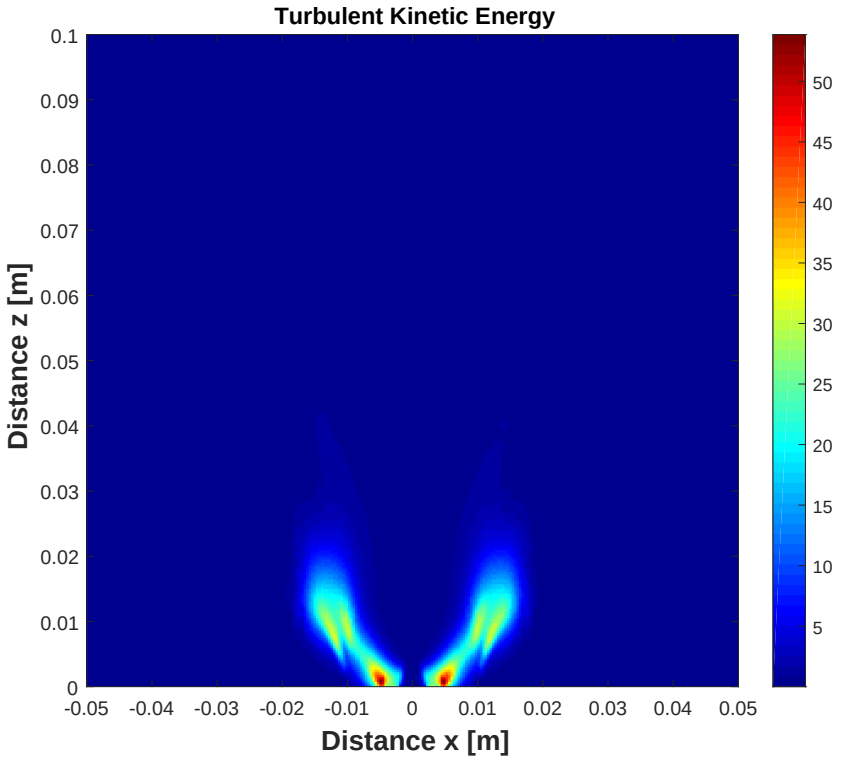


Figura A.3: Contorno de energía cinética turbulenta del caso premezclado RANS_17_CG.

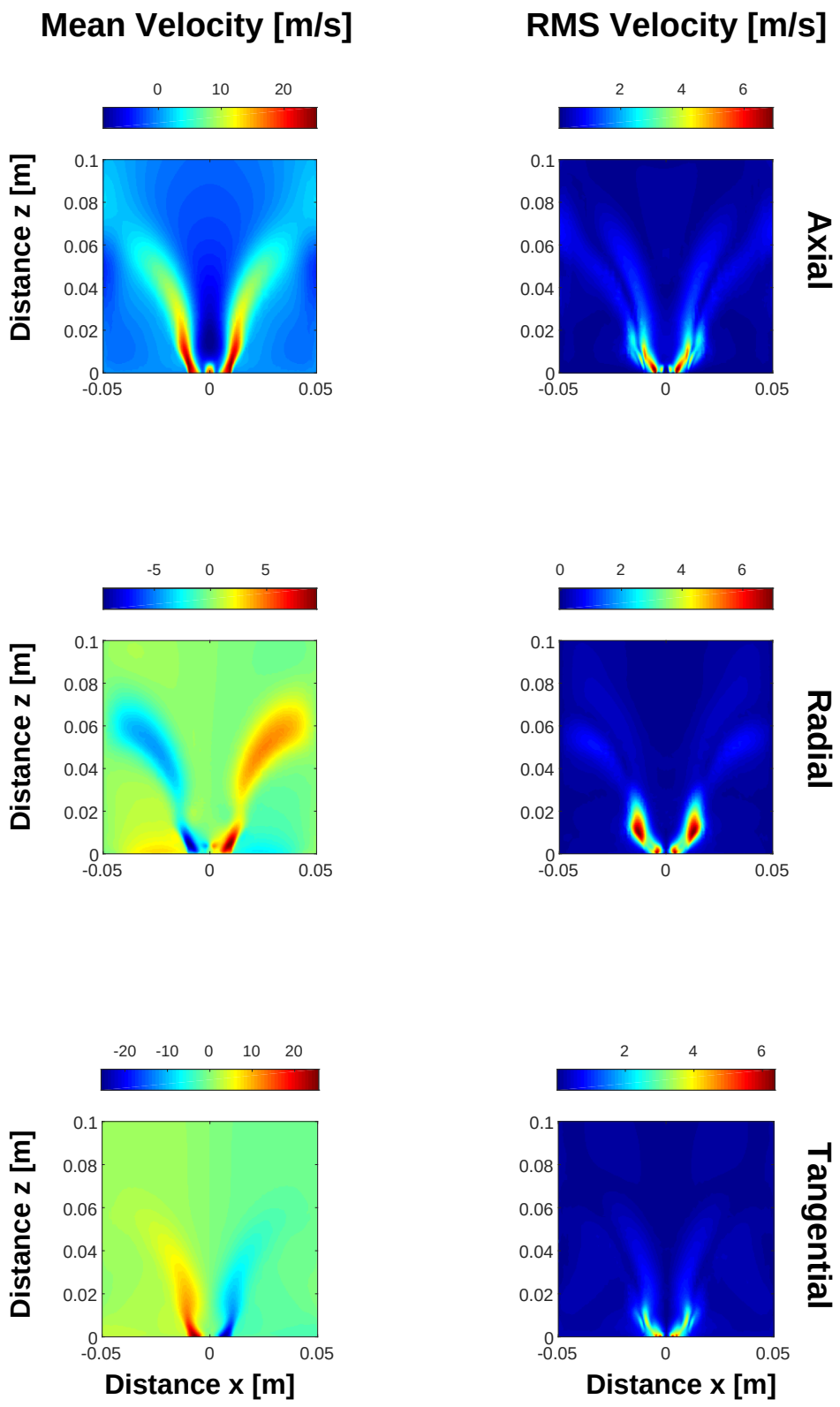


Figura A.4: Contornos de velocidades del caso premezclado RANS_17_CG.

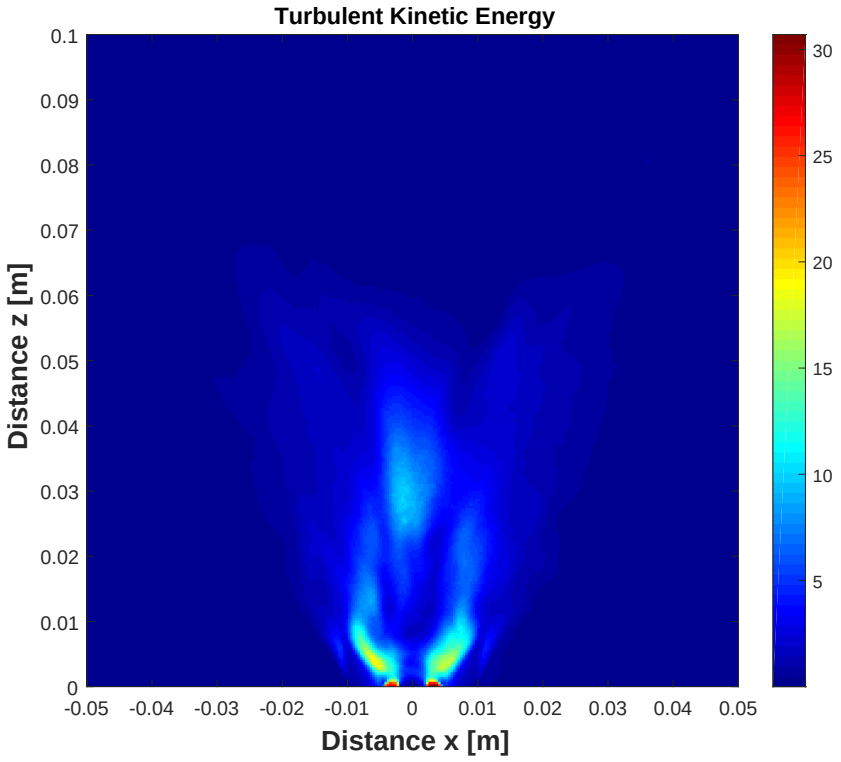


Figura A.5: Contorno de energía cinética turbulenta del caso premezclado URANS_18_OF.

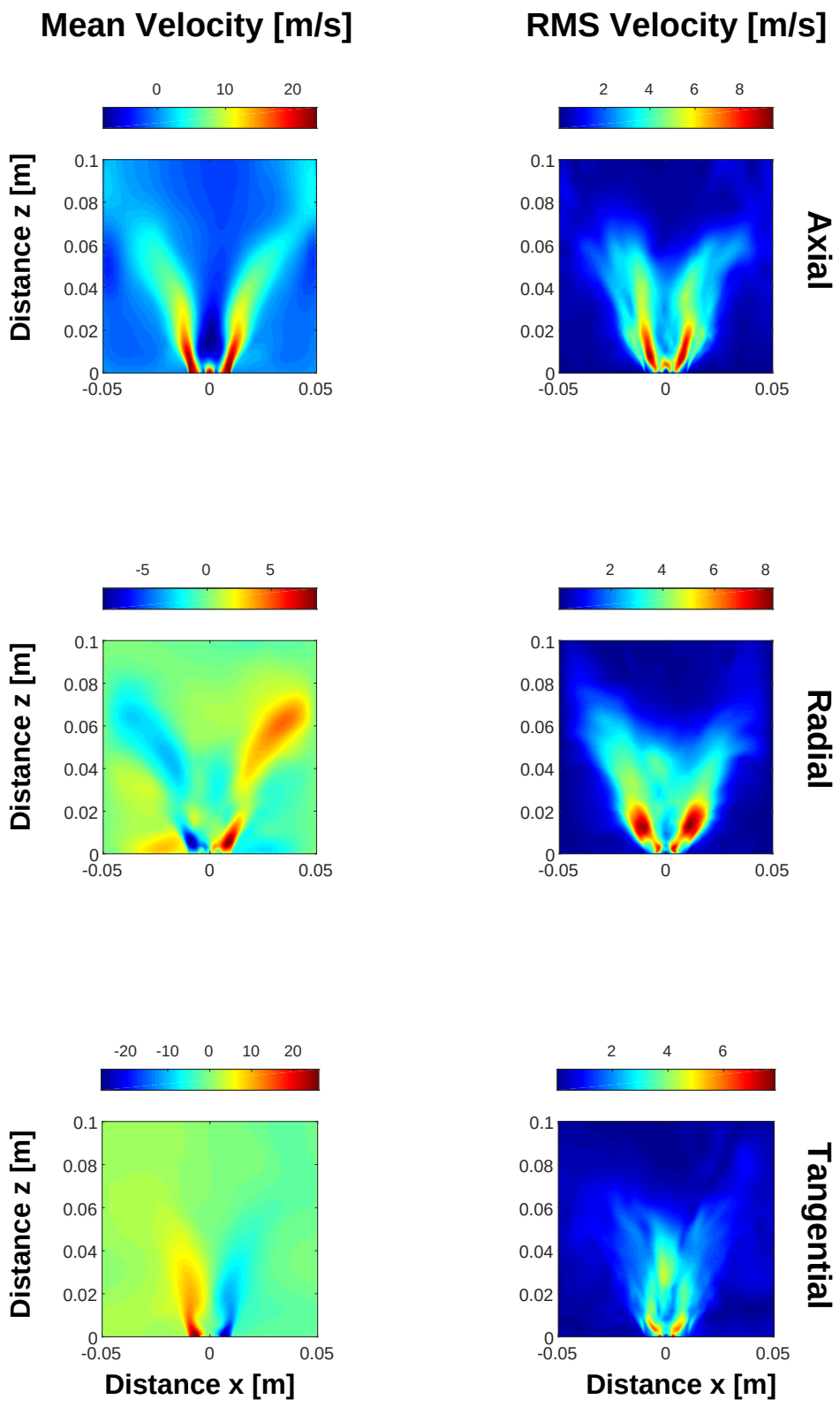


Figura A.6: Contornos de velocidades del caso premezclado URANS_18_OF.

Apéndice B

Código Matlab para posprocesar resultados

B.1. LANZA

codigo/lanza.m

```
% Carlos Moreno Montagud 21/02/2018
%
% Folder structure:
%   \'code'\'geometry'\'injection'\'fluid'\'inletType'\cases\'
   case'\'folder'\
%                                                          \mats\
%   \Literature\   "    \      "      \    "    \      "      \cases\'
   author'\
%                                                          \mats\
%   \Results\     "     \      "      \    "    \      "      \'
   solveMode'\'code'\
%
% Importing CONVERGE:
%   lanza.m
%   - Define miliSec (3 digits)    miliSec={'___'};
%   - Void mesh                      mesh={''};
%   importProperties.m
%   - Name of col files              col_files={'velocities','
   stress','species'};
%
% Importing OpenFOAM:
%   - Transient cases' instant folder must contain at least 3
   digits after
```

```
%   the point                                'case'/20000.060
%   - Coordinates files ccx, ccy and ccz must be placed in 0
    folder
%                                            'case'/0/ccx
%   lanza.m
%   - Mandatory mesh                         mesh={'6M'};
%
%Importing Literature:
%   lanza.m
%   - Just 1 file                            author={'XXX'};
%   - Void mesh                              mesh={''};
%   importProperties.m
%   - DON'T import stress file               col_files={...,'stress
    ',...}

%% Script
clearvars;
clc;

addpath(genpath('functions'));

%% Initialize variables
%Case variables
cases.code = {'OpenFOAM';'OpenFOAM';'CONVERGE';'CONVERGE'}; %
    'CONVERGE', 'OpenFOAM', 'Literature'
cases.geometry = {'KIAI'}; % 'NASA', 'KIAI'
cases.solveMode = {'RANS';'LES';'RANS';'LES'}; % 'RANS', 'LES'
cases.injection = {'premixed'}; % 'nonPremixed', 'premixed'
cases.fluid = {'gas'}; % 'gas', 'liquid'
cases.inletType = {'nonReactive'}; % 'nonReactive', 'reactive'
cases.numCase = {'3';'2';'17';'3'};
cases.miliSec = {'119';'128';'433';'215'}; % '', '165', '185'
cases.mesh = {'17M';'17M';'';''}; % '', '6M', '17M'
cases.author = {'Cordier2013EXP'}; % 'JunCai', 'PatelMenon', '
    Cordier2013EXP', 'Esclapez2014AVBP'

%Post-processing flags
flags.plot_flag = true;

%Import variables
```

```
import = importProperties();

% Plot variables
plotting = plotProperties();

% Path names
names = path_names(cases, plotting.legend_flags);

%% Load cases
[plotting.EXP, plotting.SIM] = loadCases(names, import, flags);

%% Post-processing
if flags.plot_flag

    % Variables
    plotting.legend_auto_exp = names.legend_exp;
    plotting.legend_auto_sim = names.legend_sim;
    plotting.results_folder = names.results_folder;

    % Plot
    num_comp = plotCase(plotting);
end

restoredefaultpath;
```

B.2. IMPORTPROPERTIES

codigo/importProperties.m

```
% 16/05/2018
% Carlos Moreno Montagud

% import.files refers to the name of the files .col exported
  from CONVERGE.
% import.*.variables refers to the variables you can obtain
  from that file.
% import.*.file_end refers to the final part of the filename
  where the data
%   will be read when importing EXP file.
% import.*.plot_type refers to the way of importing and
  showing the
```

```
%  information: a single line, many stations along z axis or
   contours.
%import.*.*.statistic is ignored when importing OpenFOAM case
   .
%
%The rest of substructures are self-explained by name.
%Unused variables will be skipped so the script won't return
   errors, but
%can be commented to save some memory. Don't delete them as
   they are good
%examples for new variables and structures.

function import = importProperties()
%% General
%Geometry and meshing properties
import.base_size = 0.003; %[m]
import.AMR = 3; %level of refinement
import.xlims = [-0.05 0.05]; %cc width [m]
import.ylims = [-0.05 0.05]; %cc height [m]
%import.zlims = [];

%%CONVERGE files
import.col_files = {'velocities','stress'}; %'velocities','
   stress','species'
import.out_files = {'monitor'};

%Import variables
import.velocities_file.variables = {'velocities','tke'};
import.stress_file.variables = {'stress'};
import.species_file.variables = {'species'};

import.monitor_file.variables = {'pressure'};
import.monitor_file.fs = 10000; %
%import.monitor_file.points = 2; %%%%%%%

%% OpenFOAM files
import.crd_files = {'ccx','ccy','ccz'};
import.var_files = {'UMean','UPrime2Mean'};

%Import variables
```

```
import.UMean.variables = {'velocities'};
import.UMean.numCols = 3;
import.UMean.impCols = [3 1 2]; % axial, radial, tangential (
    in order)
import.UPrime2Mean.variables = {'velocities','tke'};
import.UPrime2Mean.numCols = 6;
import.UPrime2Mean.impCols = [6 1 4]; % axial, radial,
    tangential (in order)

%%
% Velocities
import.velocities.file_end = {'velocity'};
import.velocities.plot_type = {'centerline','stations','
    contours'}; % 'centerline', 'stations', 'contours'
    % Centerline
    import.velocities.centerline.plane = {'z'}; %
    import.velocities.centerline.statistic = {'mean'}; % 'mean
        ', 'RMS'
    import.velocities.centerline.direction = {'axial'}; % '
        axial', 'radial', 'tangential'
    import.velocities.centerline.limits = [0 0.05]; %[0 0.05]
    % Stations
    import.velocities.stations.plane = {'xz'}; % 'xz', 'yz'
    import.velocities.stations.statistic = {'mean','RMS'}; % '
        mean', 'RMS'
    import.velocities.stations.direction = {'axial','radial','
        tangential'}; % 'axial', 'radial', 'tangential'
    import.velocities.stations.stations = [5 10 20 30 40]; %
        [5 15 29 46 76 92], [5 10 20 30 40]
    % Contours
    import.velocities.contours.plane = {'xz'}; % 'xz', 'yz'
    import.velocities.contours.statistic = {'mean','RMS'}; % '
        mean', 'RMS'
    import.velocities.contours.direction = {'axial','radial','
        tangential'}; % 'axial', 'radial', 'tangential'
    import.velocities.contours.limits = [0 0.1]; %[-0.05 0.05
        0 0.1]

% Tke
import.tke.file_end = {'turbulent','kinetic','energy'};
```

```
import.tke.plot_type = {'contours'}; % 'centerline', 'stations
    ', 'contours'
     %Contours
     import.tke.contours.plane = {'xz'};
     import.tke.contours.statistic = {'RMS'}; % 'RMS'
     import.tke.contours.direction = {'axial','radial','
        tangential'}; % 'axial', 'radial', 'tangential'
     import.tke.contours.limits = [0  0.1]; %[-0.05  0.05  0  0.1]

% Stress
import.stress.file_end = {'stress'};
import.stress.plot_type = {'contours'}; % 'centerline', '
    stations', 'contours'
     % Contours
     import.stress.contours.plane = {'xz'}; % 'xz', 'yz'
     import.stress.contours.direction = {'s11','s12','s13','s22
        ','s23','s33'}; % 's11', 's12', 's13'...
     import.stress.contours.limits = [0.001  0.1]; %[-0.05  0.05
         0  0.1]

% Species
import.species.file_end = {'mass','fraction'};
import.species.plot_type = {'stations'}; % 'centerline', '
    stations', 'contours'
     % Stations
     import.species.stations.plane = {'xz'}; % 'xz', 'yz'
     import.species.stations.statistic = {'mean','RMS'}; % '
        mean', 'RMS'
     import.species.stations.direction = {'CH4'}; %
     import.species.stations.stations = [6  10  14  22]; %[6  10
        14  22]

end
```

B.3. PLOTPROPERTIES

codigo/plotProperties.m

```
% 16/05/2018
% Carlos Moreno Montagud
```

```
function plotting = plotProperties()
%% Flags
plotting.lines_flag = true;
plotting.stations_flag = true;
plotting.contours_flag = true;
plotting.monitor_flag = false;

plotting.compare_flag = true;

%% Save
plotting.save_flag = true;
plotting.save_format = {'pdf'}; % 'pdf' 'epsc' 'png 'tiff'

%% Legend
plotting.legend_mode = 'manual'; % 'auto' 'manual'
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plotting.legend_flags = [1 0 0 0 1 1 1 0]; % boolean values [
   KR NP G NR X XXX CG 6M]
plotting.legend_manual_exp = {'Cordier2013EXP'};
plotting.legend_manual_sim = {'URANS_18_OF','LES_17_OF','
   RANS_17_CG','LES_3_CG'};

%% Centerline
plotting.centerline.type = {'velocities'};

plotting.centerline.velocities.plane = {'z'};
plotting.centerline.velocities.statistic = {'mean'};
plotting.centerline.velocities.directions = {'axial'};
plotting.centerline.velocities.file_end = {'velocity'};
plotting.centerline.velocities.y_units = '[m/s]';
%plotting.centerline.velocities.aspect_ratio = [5 1 1];
plotting.centerline.velocities.limits.mean.axial = [0 0.05 -15
    35]; %[0 0.05 -15 35] %%%%%%

%% Stations
plotting.stations.type = {'velocities'}; % 'velocities', '
   species'

plotting.stations.velocities.plane = {'xz'}; % 'xz', 'yz'
```

```
plotting.stations.velocities.statistic = {'mean','RMS'}; %'
    inst', 'mean', 'RMS'
plotting.stations.velocities.directions = {'axial','radial','
    tangential'}; %'axial', 'radial', 'tangential'
plotting.stations.velocities.stations = [5 10 20 30 40]; %[5
    15 29 46 76 92], [5 10 20 30 40]
plotting.stations.velocities.file_end = {'velocity'};
plotting.stations.velocities.y_units = '[m/s]';
%plotting.stations.velocities.aspect_ratio = [5 1 1];
plotting.stations.velocities.limits.xlimit = [-0.05 0.05];
plotting.stations.velocities.limits.mean.axial = [-10 25];
plotting.stations.velocities.limits.mean.radial = [-5 15];
plotting.stations.velocities.limits.mean.tangential = [-25 5];
plotting.stations.velocities.limits.RMS.axial = [0 16];
plotting.stations.velocities.limits.RMS.radial = [0 10];
plotting.stations.velocities.limits.RMS.tangential = [0 10];

plotting.stations.species.plane = {'xz'}; %'xz', 'yz'
plotting.stations.species.statistic = {'mean','RMS'}; %'mean
    ', 'RMS'
plotting.stations.species.directions = {'CH4'}; %
plotting.stations.species.stations = [6 10 14 22]; %[6 10 14
    22]
plotting.stations.species.file_end = {'mass','fraction'};
plotting.stations.species.y_units = '[-]';
%plotting.stations.species.aspect_ratio = [5 1 1];
plotting.stations.species.limits.xlimit = [-0.03 0.03];
plotting.stations.species.limits.mean.CH4 = [0 0.15];
plotting.stations.species.limits.RMS.CH4 = [0 0.20];

%% Contours
plotting.contours.type = {'velocities','tke','stress'}; %'
    velocities', 'tke', 'stress', 'yplus'

plotting.contours.velocities.plane = {'xz'}; %'xz', 'yz'
plotting.contours.velocities.statistic = {'mean','RMS'}; %'
    mean', 'RMS'
plotting.contours.velocities.directions = {'axial','radial','
    tangential'}; %'axial', 'radial', 'tangential'
plotting.contours.velocities.file_end = 'velocity';
```

```
plotting.contours.velocities.y_units = '[m/s]';
plotting.contours.velocities.title_end = 'Velocity';
plotting.contours.velocities.limits = [-0.05 0.05 0 0.1]; %
    [-0.05 0.05 0 0.1]
plotting.contours.velocities.aspect_ratio = [1 1 1];

plotting.contours.tke.plane = {'xz'}; % 'xz', 'yz'
plotting.contours.tke.file_end = 'tke';
plotting.contours.tke.y_units = '[-]';
plotting.contours.tke.title_end = 'Turbulent Kinetic Energy';
plotting.contours.tke.limits = [-0.05 0.05 0 0.1]; %[-0.05
    0.05 0 0.1]
plotting.contours.tke.aspect_ratio = [1 1 1];

plotting.contours.stress.plane = {'xz'}; % 'xz', 'yz'
plotting.contours.stress.file_end = 'stress';
plotting.contours.stress.y_units = '[-]';
plotting.contours.stress.title_end = 'Reynolds Stress';
%plotting.contours.stress.limits = [0 0.1]; %[-0.05 0.05 0
    0.1]
plotting.contours.stress.aspect_ratio = [1 1 1];

%plotting.contours.yplus.plane = {'xz'}; % 'xz', 'yz'
%plotting.contours.yplus.limits = [-0.05 0.05 0 0.1]; %[-0.05
    0.05 0 0.1]
%plotting.contours.yplus.

end
```

B.4. PLOTCONFIG

codigo/plotConfig.m

```
% 14/05/2018
% Carlos Moreno Montagud

function config = plotConfig()

%% Line style
config.sim_markers = ['s' '^' 'x' 'd' 'v' '.' '>' '*' '<']; %
    Markers for simulations
```

```
config.sim_marker_size = 6;
config.sim_line_width = 1.5;
config.line_style = ['␣-';'--';'-.';'␣:']; % Line order

config.exp_markers = ['o' '+']; % Markers for experimental
config.exp_marker_size = 6;
config.exp_line_width = 1.5;
config.exp_line_style = 'none';

config.colors = ['b' 'g' 'r' 'c' 'm' 'y' 'w']; % Color order
config.primes = [5, 7, 11, 13, 17]; % Spacing markers

%% Font sizes
config.lines.legend_fontsize = 10;
%config.lines.title_fontsize = 14;
config.lines.xlabel_fontsize = 14;
config.lines.ylabel_fontsize = 14;

config.stations.stations_fontsize = 10;
config.stations.legend_fontsize = 10;
%config.stations.title_fontsize = 14;
config.stations.xlabel_fontsize = 14;
config.stations.ylabel_fontsize = 14;

%config.contours.legend_fontsize = 10;
config.contours.title_fontsize = 12;
config.contours.xlabel1_fontsize = 10;
config.contours.ylabel1_fontsize = 10;
config.contours.xlabel2_fontsize = 14;
config.contours.ylabel2_fontsize = 14;

%% Contour
config.cmap = jet; %jet, gray

end
```

B.5. PATH_NAMES

codigo/functions/path_names.m

```
% 04/04/2018
% Carlos Moreno Montagud

function [names] = path_names(cases,legend_flags)

%% Cases data
lgeo = length(cases.geometry);
lsol = length(cases.solveMode);
linj = length(cases.injection);
lflu = length(cases.fluid);
linl = length(cases.inletType);
lnum = length(cases.numCase);
lmil = length(cases.miliSec);
lcod = length(cases.code);
lmes = length(cases.mesh);

casesNum = max([lgeo lsol linj lflu linl lnum lmil lcod lmes])
    ;
names.casesNum = casesNum;

geometry = repmat(cases.geometry,casesNum/lgeo,1);
solveMode = repmat(cases.solveMode,casesNum/lsol,1);
injection = repmat(cases.injection,casesNum/linj,1);
fluid = repmat(cases.fluid,casesNum/lflu,1);
inletType = repmat(cases.inletType,casesNum/linl,1);
numCase = repmat(cases.numCase,casesNum/lnum,1);
miliSec = repmat(cases.miliSec,casesNum/lmil,1);
code = repmat(cases.code,casesNum/lcod,1);
mesh = repmat(cases.mesh,casesNum/lmes,1);

injectionAbbr = cellfun(@abbr,injection,'un',0);
inletTypeAbbr = cellfun(@abbr,inletType,'un',0);
fluidAbbr = cellfun(@(x) upper(x(1)),fluid,'un',0);
codeAbbr = cellfun(@abbr,code,'un',0);

%% Names
% caseName        NASA_RANS_NP_G_NR_X / JunCai
% matName         NR_NP_G_NR_X_XXX_CG_(6M) / JunCai
geoSolve = mat2cell(cellfun(@(x) x(1),[geometry solveMode]),
    ones(1,casesNum),2);
```

```
fileCell = mat2cell([geoSolve injectionAbbr fluidAbbr
    inletTypeAbbr numCase miliSec codeAbbr mesh],ones(1,
    casesNum),8);

if strcmp(code,'Literature')
     file = cases.author(1);
     names.caseName = file;
     names.matName = file;
else
     folderCell = mat2cell([geometry solveMode injectionAbbr
         fluidAbbr inletTypeAbbr numCase],ones(1,casesNum),6);
     folder = cellfun(@(x) strjoin(x,'_'),folderCell,'un',0);
     names.caseName = folder;

     file = strrep(strcat(cellfun(@(x) strjoin(x,'_'),fileCell,
         'un',0),'_'),'__','_');
     file = cellfun(@(x) x(1:end-1),file,'un',0);
     names.matName = file;
end

%expName          JunCai
names.expName = cases.author;

%legend_sim      NR_NP_G_NR_X_XXX_CG_6M
names.legend_sim = strrep(cellfun(@(x) strjoin(x(logical(
    legend_flags)),'_'),fileCell,'un',0),'__','_');

%legend_exp      JunCai %%%%%%%%%%%%%
names.legend_exp = cases.author;

%mainFolder      CONVERGE\NASA\nonPremixed\(6M)\gas\nonReactive
    \___\
mainCell = mat2cell([code geometry injection mesh fluid
    inletType],ones(1,casesNum),6);
mainFolder = strrep(cellfun(@(x) strjoin(x,filesep),mainCell,'
    un',0),[filesep filesep],filesep);
names.casesFolder = strcat(mainFolder,filesep,'cases',filesep)
    ;
names.matsFolder = strcat(mainFolder,filesep,'mats',filesep);
```

```
%importFolder   CONVERGE\NASA\nonPremixed\(6M)\gas\nonReactive
    \cases\NASA_RANS_NP_G_NR_X\
names.importFolder = strcat(names.casesFolder,names.caseName,
    filesep);

%expFolder      Literature\NASA\nonPremixed\gas\nonReactive\
    mats\
expFolder = strrep(names.matsFolder,[filesep mesh{1} filesep],
    filesep);
names.expFolder = strrep(expFolder,code{1},'Literature');

%matFile        CONVERGE\NASA\nonPremixed\(6M)\gas\nonReactive
    \mats\NR_NP_G_NR_X_XXX_CG_(6M).mat
names.matFile = strcat(names.matsFolder,names.matName,'.mat');

%expFile        Literature\NASA\nonPremixed\gas\nonReactive\
    mats\JunCai.mat
names.expFile = strcat(names.expFolder,cases.author,'.mat');

%results        Results\NASA\nonPremixed\gas\nonReactive\RANS\
    CONVERGE\
results_folder = ['Results' filesep];
%resultsFilename = strjoin(file ', '_'); %%%%%%%%%%%%%%%

geo = unique(geometry); %NASA
if length(geo)==1
    results_folder = strcat(results_folder,geo,filesep);
end

inj = unique(injection); %nonPremixed
if length(inj)==1
    results_folder = strcat(results_folder,inj,filesep);
end

%msh = unique(mesh); %(6M)
%if length(msh)==1
%    resultsFolder = strcat(resultsFolder,msh,filesep);
%end

flu = unique(fluid); %gas
```

```
if length(flu)==1
    results_folder = strcat(results_folder,flu,filesep);
end

inl = unique(inletType); %nonReactive
if length(inl)==1
    results_folder = strcat(results_folder,inl,filesep);
end

sol = unique(solveMode); %RANS
if length(sol)==1
    results_folder = strcat(results_folder,sol,filesep);
else
    results_folder = strcat(results_folder,'VERSUS',filesep);
end

cod = unique(code); %CONVERGE
if length(cod)==1
    results_folder = strcat(results_folder,cod,filesep);
end

num = unique(numCase); %CASE_X
if length(num)==1
    results_folder = strcat(results_folder,'CASE_',num,filesep
        );
end

%mil = unique(miliSec); %XXX
%if length(mil)==1
%    resultsFolder = strcat(resultsFolder,mil,'ms',filesep);
%end

names.results_folder = results_folder{1};
names.code = code;
names.miliSec = miliSec;
end

function str = abbr(fstr)

switch fstr
```

```
    case 'CONVERGE'
        str = 'CG';
    case 'OpenFOAM'
        str = 'OF';
    case 'nonPremixed'
        str = 'NP';
    case 'premixed'
        str = 'P';
    case 'nonReactive'
        str = 'NR';
    case 'reactive'
        str = 'R';
    otherwise
        str = 'error';
end

end
```

B.6. UPPERFIRST

codigo/functions/upperFirst.m

```
function str = upperFirst(data)
str = [upper(data(1)) data(2:end)];
end
```

B.7. LOADCASES

codigo/functions/import/loadCases.m

```
%07/05/2018
%Carlos Moreno Montagud
function [EXP,SIM]=loadCases(names,import,flags)

expNum = length(names.expName);
EXP=cell(expNum,1);
SIM=cell(names.casesNum,1);

%Load SIM
for i=1:names.casesNum
```

```
    if exist(names.matFile{i},'file')
        %Load file.mat
        load(names.matFile{i,1},names.matName{i,1});
        SIM{i,1} = eval(names.matName{i,1});
    else
        %Import case and save file.mat
        SIM{i,1} = importCase(names,import,i);
        eval(strcat(names.matName{i},'=SIM{i,1};'));
        save(names.matFile{i},names.matName{i});
    end
end

%Load EXP
if flags.plot_flag || flags.compare_flag
    for i=1:expNum
        load(names.expFile{i,1},names.expName{i,1});
        EXP{i,1} = eval(names.expName{i,1});
    end
end

end
```

B.8. IMPORTCASE

codigo/functions/import/importCase.m

```
%04/04/2018
%Carlos Moreno Montagud

function dataArray = importCase(names,import,n)
%% Variables
dataArray = [];
code = names.code{n};
miliSec = names.miliSec{n};
importFolder = names.importFolder{n};

col_files = import.col_files;
out_files = import.out_files;

%% Import
```

```
switch code
    case 'CONVERGE'
        %Search files .col
        colFolder = strcat(importFolder,'output',filesep);
        miliSecStr = num2str(str2double(miliSec)/1000,'%1.5e')
            ;
        miliSecRead = 5-str2double(miliSecStr(end));
        readColFileStr = strcat(colFolder,col_files,'*_+',
            miliSecStr(1:miliSecRead),'*',miliSecStr(end-3:end)
            ,'.col');
        readColFile = cellfun(@dir,readColFileStr,'un',0);
        readColFileMat = cell2mat(readColFile);
        lastReadColFile = {readColFileMat(end,:).name};
        readColFilename = strcat(colFolder,lastReadColFile);

        %Search files .out
        readOutFileStr = strcat(importFolder,out_files,'*_','
            point_',{'1','2'},'_','volume','_avg.out');
        readOutFile = cellfun(@dir,readOutFileStr,'un',0);
        readOutFileMat = cell2mat(readOutFile);
        allReadOutFile = reshape({readOutFileMat.name},size(
            readOutFileMat));
        readOutFilename = strcat(importFolder,allReadOutFile);
            %siempre 1 columna

        %Read variables
        delimiter = '';
        startRow = 6;

        %Import CONVERGE files
        dataArray = importCVG(readColFilename,readOutFilename,
            import,delimiter,startRow);

    case 'OpenFOAM'
        %Search 0 and last 'miliSec' folder
        readFile_temp = dir(importFolder);
        readFile_temp = readFile_temp(~isnan(str2double({
            readFile_temp.name})));
        if ~isempty(miliSec)
```

```
            readFile_find = strfind({readFile_temp.name},['.',
                miliSec]);
            readFile_find = cellfun(@(x) ~isempty(x),
                readFile_find);
            readFile_find(1) = 1;
            readFile = readFile_temp(readFile_find);
        else
            readFile = readFile_temp([1,end]);
        end
        lastReadFile = {readFile.name};
        readFolders = strcat(importFolder,lastReadFile,filesep
            );

        %Read variables
        delimiter = ' ';
        startRow = 23;

        %Import OpenFOAM files
        dataArray = importFOAM(readFolders,import,delimiter,
            startRow); %%%%%%%%%%%%%

    case 'Literature'
        %Read variables
        delimiter = ';'; %%%%%%%%%%%%%%%%%%%%
        startRow = 2;

        %Import Literature files
        dataArray = importEXP(importFolder,import,delimiter,
            startRow);
end

end
```

B.9. IMPORTCVG

codigo/functions/import/importCVG.m

```
%01/02/2018
%Mario Belmar Gil
%21/02/2018
```

```
% Carlos Moreno Montagud

% Export output data from CONVERGE Studio in separated .col
    files so RAM is
% not an issue in postprocess. The name of the files is
    selected in
% importProperties.m

function dataArray = importCVG(readColFilename,readOutFilename
    ,import,delimiter,startRow)
%% Variables
dataArray=[];

base_size = import.base_size;
AMR = import.AMR;

%% Col files
col_files = strcat(import.col_files,'_file');
nFiles = length(col_files);
for ff=1:nFiles % 'velocities', 'stress', 'species'
    current_file = col_files{ff};
    F=[];

    % Import raw data
    CFD_raw = importdata(readColFilename{ff});
    variables_names = arrayfun(@(str) strrep(str,'.','_'),
        CFD_raw.colheaders);
    CFD_data = array2table(CFD_raw.data, 'VariableNames',
        variables_names);

    variables = import.(current_file).variables;
    nVar = length(variables);
    for vv=1:nVar % 'velocities', 'tke', 'stress', 'species'
        current_var = variables{vv};

        plot_type = import.(current_var).plot_type;
        nPlot = length(plot_type);
        for pp=1:nPlot % 'centerline', 'stations', 'contours'
            current_plot = plot_type{pp};
            struct_plot = import.(current_var).(current_plot);
```

```
plane = struct_plot.plane;
nPlane = length(plane);
for zz=1:nPlane
    current_plane = plane{zz};

    switch current_plot

        %CENTERLINE
        case 'centerline'

            % Coordinates
            coord_min = struct_plot.limits(1);
            coord_max = struct_plot.limits(2);
            discretization = round((coord_max-
                coord_min)*2^AMR/base_size);
            coords = linspace(coord_min, coord_max
                , discretization)';
            zero_vector = zeros(discretization, 1)
                ;
            dataArray.centerline.(current_plane).(
                current_var).coords = coords;

            direction = struct_plot.direction;
            nDirection = length(direction);
            for dd=1:nDirection % 'axial'
                current_direction = direction{dd};

                statistic = struct_plot.statistic;
                nStatistic = length(statistic);
                for ss=1:nStatistic % 'mean'
                    current_statistic = statistic{
                        ss};

                    %%%%%%plane z axial = W, etc
                    head = [headers(
                        current_statistic) headers(
                        current_direction)];
                    try
                        % Interpolation
```

```
                    if ~isfield(F,head)
                        F.(head) =
                            scatteredInterpolant
                            (CFD_data.x,
                            CFD_data.y,
                            CFD_data.z,
                            CFD_data.(head));
                    end
                    % Discretization
                    dataArray.centerline.(
                        current_plane).(
                        current_var).(
                        current_statistic).(
                        current_direction).
                        centerline = F.(head)(
                        zero_vector,
                        zero_vector, coords);
                catch
                    disp('error cvg centerline
                        ');
                end
            end
        end

    %STATIONS
    case 'stations'

        % Coordinates
        coord_min = import.([current_plane(1),
            'lims']){1};
        coord_max = import.([current_plane(1),
            'lims']){2};
        discretization = round((coord_max-
            coord_min)*2^AMR/base_size);
        coords = linspace(coord_min, coord_max
            , discretization)';
        zero_vector = zeros(discretization, 1)
            ;
        switch current_plane
            case 'xz'
```

```
                c1 = coords;
                c2 = zero_vector;
            case 'yz'
                c1 = zero_vector;
                c2 = coords;
        end
        dataArray.stations.(current_plane).(
            current_var).coords = coords;

        direction = struct_plot.direction;
        nDirection = length(direction);
        for dd=1:nDirection %'axial', 'radial
            ', 'tangential', 'CH4'
            current_direction = direction{dd};

            statistic = struct_plot.statistic;
            nStatistic = length(statistic);
            for ss=1:nStatistic %'mean', 'RMS
                '
                current_statistic = statistic{
                    ss};

                stations = struct_plot.
                    stations;
                nStations = length(stations);
                for mm=1:nStations %[5 10 20
                    30 40]
                    current_station = num2str(
                        stations(mm));
                    station_vector = repmat(
                        stations(mm)/1000,[
                        discretization 1]);

                    head = [headers(
                        current_statistic)
                        headers(
                        current_direction)];
                    try
                        %Interpolation
                        if ~isfield(F,head)
```

```
                                F.(head) =
                                    scatteredInterpolant
                                    (CFD_data.x,
                                    CFD_data.y,
                                    CFD_data.z,
                                    CFD_data.(head)
                                    );
                            end
                            % Discretization
                            dataArray.stations.(
                                current_plane).(
                                current_var).(
                                current_statistic)
                                .(current_direction
                                ).(['station_',
                                current_station,'mm
                                ']) = F.(head)(c1,
                                c2, station_vector)
                                ;
                        catch
                            disp('error cvg
                                stations');
                        end
                    end
                end
            end

        %CONTOURS
        case 'contours'

            % Coordinates
            limits = [import.([current_plane(1),'
                lims']) struct_plot.limits];
            discretization = round((limits(2)-
                limits(1))*2^AMR/base_size); % xy
                dependant
            vertex1 = linspace(limits(1), limits
                (2), discretization+1)'; center1 =
                vertex1(1:end-1)+diff(vertex1)/2;
```

```
vertex2 = linspace(limits(3), limits
    (4), discretization+1)'; center2 =
    vertex2(1:end-1)+diff(vertex2)/2;
coords = [center1,center2];
coord0 = zeros(discretization);
[coord1,coord2] = meshgrid(center1,
    center2);
switch current_plane
    case 'xz'
        c1 = coord1;
        c2 = coord0;
        c3 = coord2;
    case 'yz'
        c1 = coord0;
        c2 = coord1;
        c3 = coord2;
end
dataArray.contours.(current_plane).(
    current_var).coords = coords;

direction = struct_plot.direction;
nDirection = length(direction);
switch current_var

    % Velocities
    case 'velocities'
        for dd=1:nDirection % 'axial',
            'radial', 'tangential'
            current_direction =
                direction{dd};

            statistic = struct_plot.
                statistic;
            nStatistic = length(
                statistic);
            for ss=1:nStatistic % '
                mean', 'RMS'
                current_statistic =
                    statistic{ss};
```

```
                head = [headers(
                    current_statistic)
                    headers(
                    current_direction)
                    ];
                try
                    % Interpolation
                    if ~isfield(F,head
                        )
                        F.(head) =
                            scatteredInterpolant
                            (CFD_data.x
                            , CFD_data.
                            y, CFD_data
                            .z,
                            CFD_data.(
                            head));
                    end
                    % Discretization
                    dataArray.contours
                        .(current_plane
                        ).(current_var)
                        .(
                        current_statistic
                        ).(
                        current_direction
                        ).contour = F.(
                        head)(c1, c2,
                        c3);
                catch
                    disp('error cvg
                        contour
                        velocities');
                end
            end
        end

        % Tke
        case 'tke'
```

```
result = zeros(discretization,
    discretization,nDirection);

current_statistic =
    struct_plot.statistic{1}; %
     'RMS'

for dd=1:nDirection %'axial',
     'radial', 'tangential'
     current_direction =
         direction{dd};

     head = [headers(
         current_statistic)
         headers(
         current_direction)];

     try
          %Interpolation
          if ~isfield(F,head)
               F.(head) =
                   scatteredInterpolant
                   (CFD_data.x,
                   CFD_data.y,
                   CFD_data.z,
                   CFD_data.(head)
                   );
          end
          %Discretization
          vel = F.(head)(c1, c2,
               c3);
          %Calculate
          result(:,:,dd) = vel.*
              vel;
     catch
          disp('error cvg
              contour tke');
     end
end
```

```
            dataArray.contours.(
                current_plane).(current_var
                ).contour = sum(result,3)
                /2;

        % Stress
        case 'stress'
            total_stress = zeros(size(
                CFD_data.x));

            for dd=1:nDirection % 's11', '
                s12', 's13'...
                current_direction =
                    direction{dd};
                head = current_direction;

                % Calculate
                try
                    stress = CFD_data.(
                        head);
                    total_stress =
                        total_stress+stress
                        .^2;
                    % Symmetry
                    if rem(str2double(
                        current_direction
                        (2:end)),11)~=0
                        total_stress =
                            total_stress+
                            stress.^2;
                    end
                catch
                end
            end

            % Interpolation
            %if ~isfield(F,head)
            F.stress =
                scatteredInterpolant(
                CFD_data.x, CFD_data.y,
```

```
                                        CFD_data.z, sqrt(
                                        total_stress));
                                    %end
                                    % Discretization
                                    dataArray.contours.(
                                        current_plane).(current_var
                                        ).contour = F.stress(c1, c2
                                        , c3);

                                    % Otherwise
                            end
                        end
                    end
                end
            end
        end
    end
end

%% Out files
if ~isempty(readOutFilename)
    out_files = strcat(import.out_files{1},'_file'); % monitor
        only
    fs = import.(out_files).fs;
    F = [];

    out_subfiles = readOutFilename;
    [nSubfiles,nPoints] = size(out_subfiles);
    for oo=1:nPoints % 'point_1', ...

        dataRead = [];
        for ff=1:nSubfiles % 'monitor38', ...
            current_subfile = out_subfiles{ff,oo};

            % Import
            numCols = 4;
            dataRead = [dataRead; cell2mat(importTextFile(
                current_subfile,delimiter,startRow,numCols,'%17
                f',false))];
            %'MultipleDelimsAsOne',1
        end
```

```
        %F.(['point_',num2str(oo)]) = dataRead;
        variables = import.(out_files).variables;
        nVar = length(variables);
        for vv=1:nVar % 'pressure', 'W', 'massfrac_ch4'
            current_var = variables{vv};

            F.(current_var).(['point_',num2str(oo)]) =
               dataRead(:,vv+1);

            if oo==nPoints
                dataArray.psd.(current_var).cross_spectrum =
                   angle(cpsd(F.(current_var).point_1,F.(
                   current_var).point_2,[],[],[],fs));
               %dataArray.psd.(current_var).msqrd_coherence =
                   mscohere(F.(current_var).point_1,F.(
                   current_var).point_2,[],[],[],fs);
            end
        end
    end
end

end

%% Secondary functions
function str = headers(stat)

switch stat
%       case 'inst'
%           str = [];
    case 'mean'
        str = 'BAR_';
    case 'RMS'
        str = 'RMS_';
    case 'axial' %%%%%%%%%%%%%%%%%%% y plane
        str = 'W';
    case 'radial' %%%%%%%%%%%%%%%
        str = 'U';
    case 'tangential' %%%%%%%%%%%%%
        str = 'V';
    otherwise
```

```
        str = ['Y_' stat];  %%%%%%%%%
end

end
```

B.10. IMPORTEXP

codigo/functions/import/importEXP.m

```
% 21/02/2018
% Carlos Moreno Montagud

% Filename structure:
%   plane,statistic,direction,file_end,plot_type
% All together without space and upper case first letter of
   the variable
% Example: xzMeanAxialVelocityCenterline
%
%EXP files are separated by ';' delimiter, first row are
   titles and the
%number of columns is double of stations number (2 for
   centerline).

function dataArray = importEXP(readFolder,import,delimiter,
   startRow)
%% Variables
dataArray = [];

%% Loops
files = import.col_files;
nFiles = length(files);
for ff=1:nFiles % 'velocities', 'stress', 'species'
    current_file = [files{ff},'_file'];

    variables = import.(current_file).variables;
    nVar = length(variables);
    for vv=1:nVar % 'velocities', 'tke', 'stress', 'species'
        current_var = variables{vv};

        plot_type = import.(current_var).plot_type;
```

```
nPlot = length(plot_type);
for pp=1:nPlot % 'centerline', 'stations'
    current_plot = plot_type{pp};

    file_end = import.(current_var).file_end;
    struct_plot = import.(current_var).(current_plot);

    plane = struct_plot.plane;
    nPlane = length(plane);
    for zz=1:nPlane % 'z', 'xz'
        current_plane = plane{zz};

        direction = struct_plot.direction;
        nDirection = length(direction);
        for dd=1:nDirection % 'axial', 'radial', '
            tangential', 'CH4'
            current_direction = direction{dd};

            % error, don't import stress with
                Literature (see importProperties.m)
            statistic = struct_plot.statistic;
            nStatistic = length(statistic);
            for ss=1:nStatistic % 'mean', 'RMS'
                current_statistic = statistic{ss};

                fileCell = [{current_statistic,
                    current_direction},file_end,{
                    current_plot}];
                fileJoin = strjoin(cellfun(@upperFirst
                    ,fileCell,'un',0),'');
                readFilename = [readFolder,
                    current_plane,fileJoin,'.csv'];

                switch current_plot

                    %CENTERLINE
                    case 'centerline'
                        try
                            impData = importTextFile(
                                readFilename,delimiter,
```

```
                startRow,2,'%f',false);
        catch
            disp(['error import exp ',
                current_var,' 
                centerline']);
        end
        dataArray.centerline.(
            current_plane).(current_var
            ).(current_statistic).(
            current_direction).(
            current_plane)=cell2mat(
            impData);

    %STATIONS
    case 'stations'
        stations = struct_plot.
            stations;
        nStations = length(stations);
        try
            impData = importTextFile(
                readFilename,delimiter,
                startRow,2*nStations,'%
                f',false);
        catch
            disp(['error import exp ',
                current_var,' stations'
                ]);
        end
        for mm=1:nStations %[5 10 20
            30 40]
            current_station = num2str(
                stations(mm));
            dataArray.stations.(
                current_plane).(
                current_var).(
                current_statistic).(
                current_direction).(['
                station_',
                current_station,'mm'])
                = cell2mat(impData((2*
```

```
                                                    mm-1):(2*mm)));
                                            end
                                        end
                                    end
                                end
                            end
                        end
                    end
                end

            end
```

B.11. IMPORTFOAM

codigo/functions/import/importFOAM.m

```
% 04/04/2018
% Carlos Moreno Montagud

function dataArray = importFOAM(readFolders,import,delimiter,
    startRow)
%% Variables
dataArray=[];
CFD_data=[];

base_size = import.base_size;
AMR = import.AMR;
coord_files = import.crd_files;
var_files = import.var_files;

%% Folder 0 (coordinates)
readFilename = strcat(readFolders{1},coord_files);
nFiles = length(coord_files);
for ff=1:nFiles
    current_file = coord_files{ff};
    current_filename = readFilename{ff};

    numCols = 1;
    CFD_data.(current_file) = cell2mat(importTextFile(
        current_filename,delimiter,startRow,numCols,'%f',false)
        );
```

```
end

%% Folder end
readFilename = strcat(readFolders{2},var_files);
nFiles = length(var_files);
for ff=1:nFiles
    current_file = var_files{ff};
    current_filename = readFilename{ff};
    F=[];

    numCols = import.(current_file).numCols;
    CFD_data.(current_file) = cell2mat(importTextFile(
        current_filename,delimiter,startRow,numCols,'%f',true))
        ;

    if strcmp(current_file,'UPrime2Mean')
        CFD_data.(current_file) = real(sqrt(CFD_data.(
            current_file)));
    end

    variables = import.(current_file).variables;
    nVar = length(variables);
    for vv=1:nVar %'velocities', 'tke'
        current_var = variables{vv};

        plot_type = import.(current_var).plot_type;
        nPlot = length(plot_type);
        for pp=1:nPlot %'centerline', 'stations', 'contours'
            current_plot = plot_type{pp};
            struct_plot = import.(current_var).(current_plot);

            plane = struct_plot.plane;
            nPlane = length(plane);
            for zz=1:nPlane
                current_plane = plane{zz};
                impCols = import.(current_file).impCols;

                direction = struct_plot.direction;
                nDirection = length(direction);
```

```
for dd=1:nDirection %'axial', 'radial', '
    tangential'
    current_direction = direction{dd};

    %mean, RMS
    head = [headers(current_file) '_' headers(
        current_direction)];
    current_col = impCols(dd);

    switch current_plot
        %CENTERLINE
        case 'centerline'
            coord_min = struct_plot.limits(1);
            coord_max = struct_plot.limits(2);
            discretization = round((coord_max-
                coord_min)*2^AMR/base_size);
            coords = linspace(coord_min,
                coord_max, discretization)';
            zero_vector = zeros(discretization
                , 1);
            dataArray.centerline.(
                current_plane).(current_var).
                coords = coords;

            try
                %Interpolation
                if ~isfield(F,head)
                    F.(head) =
                        scatteredInterpolant(
                        CFD_data.ccx, CFD_data.
                        ccy, CFD_data.ccz,
                        CFD_data.(current_file)
                        (:,current_col));
                end
                %Discretization
                dataArray.centerline.(
                    current_plane).(current_var
                    ).(headers(current_file)).(
                    current_direction).
                    centerline = F.(head)(
```

```
                zero_vector, zero_vector,
                coords);
        catch
            disp('error foam centerline');
        end

    %STATIONS
    case 'stations'

        % Coordinates
        coord_min = import.([current_plane
            (1),'lims']){1};
        coord_max = import.([current_plane
            (1),'lims']){2};
        discretization = round((coord_max-
            coord_min)*2^AMR/base_size);
        coords = linspace(coord_min,
            coord_max, discretization)';
        zero_vector = zeros(discretization
            , 1);
        switch current_plane
            case 'xz'
                c1 = coords;
                c2 = zero_vector;
            case 'yz'
                c1 = zero_vector;
                c2 = coords;
        end
        dataArray.stations.(current_plane)
            .(current_var).coords = coords;

        stations = struct_plot.stations;
        nStations = length(stations);
        for mm=1:nStations %[5 10 20 30
            40]
            current_station = num2str(
                stations(mm));
            station_vector = repmat(
                stations(mm)/1000,[
                discretization 1]);
```

```
                try
                    % Interpolation
                    if ~isfield(F,head)
                        F.(head) =
                            scatteredInterpolant
                            (CFD_data.ccx,
                            CFD_data.ccy,
                            CFD_data.ccz,
                            CFD_data.(
                            current_file)(:,
                            current_col));
                    end
                    % Discretization
                    dataArray.stations.(
                        current_plane).(
                        current_var).(headers(
                        current_file)).(
                        current_direction).(['
                        station_',
                        current_station,'mm'])
                        = F.(head)(c1, c2,
                        station_vector);
                catch
                    disp('error foam stations'
                        );
                end
            end

        %CONTOURS
        case 'contours'

            % Coordinates
            limits = [import.([current_plane
                (1),'lims']) struct_plot.limits
                ];
            discretization = round((limits(2)-
                limits(1))*2^AMR/base_size); %
                xy dependant
```

```
vertex1 = linspace(limits(1),
   limits(2), discretization+1)';
   center1 = vertex1(1:end-1)+diff
   (vertex1)/2;
vertex2 = linspace(limits(3),
   limits(4), discretization+1)';
   center2 = vertex2(1:end-1)+diff
   (vertex2)/2;
coords = [center1,center2];
coord0 = zeros(discretization);
[coord1,coord2] = meshgrid(center1
   ,center2);
switch current_plane
    case 'xz'
        c1 = coord1;
        c2 = coord0;
        c3 = coord2;
    case 'yz'
        c1 = coord0;
        c2 = coord1;
        c3 = coord2;
end
dataArray.contours.(current_plane)
   .(current_var).coords = coords;

switch current_var

    % Velocities
    case 'velocities'
        try
            % Interpolation
            if ~isfield(F,head)
                F.(head) =
                    scatteredInterpolant
                    (CFD_data.x,
                    CFD_data.y,
                    CFD_data.z,
                    CFD_data.(
                    current_file)
                    (:,current_col)
```

```
                    );
                end
                % Discretization
                dataArray.contours.(
                    current_plane).(
                    current_var).(
                    headers(
                    current_file)).(
                    current_direction).
                    contour = F.(head)(
                    c1, c2, c3);
            catch
                disp('error foam
                    contour velocities'
                    );
            end

        % Tke
        case 'tke'
            result = zeros(
                discretization,
                discretization,
                nDirection);
            try
                % Interpolation
                if ~isfield(F,head)
                    F.(head) =
                        scatteredInterpolant
                        (CFD_data.x,
                        CFD_data.y,
                        CFD_data.z,
                        CFD_data.(
                        current_file)
                        (:,current_col)
                        );
                end
                % Discretization
                vel = F.(head)(c1, c2,
                    c3);
                % Calculate
```

```
                                            result(:,:,dd) = vel.*
                                                vel;
                                        catch
                                            disp('error foam 
                                                contour tke');
                                        end

                                        dataArray.contours.(
                                            current_plane).(
                                            current_var).contour =
                                            sum(result,3)/2;
                                end
                        end
                    end
                end
            end
        end
        CFD_data = rmfield(CFD_data,current_file);
end

end

%% Secondary functions
function str = headers(stat)

switch stat
%       case 'inst'
%           str = [];
    case 'UMean'
        str = 'mean';
    case 'UPrime2Mean'
        str = 'RMS';
    case 'axial' %%%%%%%%%%%%%%yz plane%
        str = 'W';
    case 'radial' %%%%%%%%%%%%%%%%
        str = 'U';
    case 'tangential' %%%%%%%%%%%%%
        str = 'V';
%       otherwise
%           str = ['Y_' stat]; %%%%%%%%%%%
```

```
end

end
```

B.12. IMPORTTEXTFILE

codigo/functions/import/importTextFile.m

```
% 21/02/2018
% Carlos Moreno Montagud

function impData = importTextFile(filename, delimiter, startRow,
    numCols, format, parenthesis)
%% Import data from text file.
% importTextFile(filename, delimiter, startRow, numCols)
% Script for importing data from the text file 'filename'
    separated by
% 'delimiter', starting in row 'startRow' with number of
    columns 'numCols'.

%% Format string for each line of text:
%   All columns: double (%f)
% formatSpec = %f%f%f%f...%[^\n\r]
% For more information, see the TEXTSCAN documentation.
formatSpec = '%*[^\n\r]';
for i=1:numCols
    formatSpec = strcat(format, formatSpec);
end
if parenthesis
    formatSpec = strcat('(',formatSpec);
end

%% Read columns of data according to format string.
fileID = fopen(filename,'r');
impData = textscan(fileID, formatSpec, 'Delimiter', delimiter, '
    CommentStyle', '//', 'EmptyValue',NaN, 'HeaderLines', startRow
    -1, 'ReturnOnError', true);
impData = cellfun(@(x) x(~isnan(x)), impData, 'un', 0);
fclose(fileID);
```

```
end
```

B.13. PLOTCASE

codigo/functions/plot/plotCase.m

```
% 21/02/2018
% Carlos Moreno Montagud

function num_comp = plotCase(plotting)

%% Variables
num_comp = [];
% Legends
if strcmp(plotting.legend_mode,'auto')
    plotting.legend_exp = plotting.legend_auto_exp;
    plotting.legend_sim = plotting.legend_auto_sim;
elseif strcmp(plotting.legend_mode,'manual')
    plotting.legend_exp = plotting.legend_manual_exp;
    plotting.legend_sim = plotting.legend_manual_sim;
end

%% Plot lines
if plotting.lines_flag
    num_comp.centerline = plotCenterline(plotting);
end

%% Plot stations
if plotting.stations_flag
    num_comp.stations = plotStationsSym(plotting);
end

%% Plot contours
if plotting.contours_flag
    plotContours(plotting); %%%%%
end

%% Plot monitor
if plotting.monitor_flag
    %plotPSD(plotting); %%%%%
```

```
end

end
```

B.14. PLOTCENTERLINE

codigo/functions/plot/plotCenterline.m

```
% 15/05/2018
% Carlos Moreno Montagud

function centerline = plotCenterline(plotting)

%% Variables
% Plotting
EXP = plotting.EXP; numEXP = length(EXP);
SIM = plotting.SIM; numSIM = length(SIM);
save_flag = plotting.save_flag;
%save_format = plotting.save_format;
legend_exp = plotting.legend_exp;
legend_sim = plotting.legend_sim;

centerline = [];

% Config
config = plotConfig();

sim_markers = config.sim_markers; lens = length(sim_markers);
exp_markers = config.exp_markers; lene = length(exp_markers);
line_style = config.line_style; lenl = length(line_style);
colors = config.colors; lenc = length(colors);
primes = config.primes; lenp = length(primes);

exp_marker_size = config.exp_marker_size;
exp_line_width = config.exp_line_width;
exp_line_style = config.exp_line_style;
sim_marker_size = config.sim_marker_size;
sim_line_width = config.sim_line_width;

legend_fontsize = config.lines.legend_fontsize;
```

```
xlabel_fontsize = config.lines.xlabel_fontsize;
ylabel_fontsize = config.lines.ylabel_fontsize;

%% Plot
type = plotting.centerline.type;
nType = length(type);
for ii=1:nType % 'velocities'
    current_type = type{ii};

    struct_type = plotting.centerline.(current_type);
    file_end = struct_type.file_end;
    y_units = struct_type.y_units;

    plane = struct_type.plane;
    nPlane = length(plane);
    for jj=1:nPlane % 'z'
        current_plane = plane{jj};

        if all(isfield(struct_type,{'statistic' 'directions'})
           ) %%%%%%%%%%
            statistic = struct_type.statistic;
            nStatistic = length(statistic);
            for kk=1:nStatistic % 'mean'
                current_statistic = statistic{kk};

                directions = struct_type.directions;
                nDirections = length(directions);
                for ll=1:nDirections % 'axial'
                    current_direction = directions{ll};

                    titCell = [{current_statistic},{
                       current_direction},file_end,{'
                       centerline'}];
                    titulo = strjoin(cellfun(@upperFirst,
                       titCell,'un',0),' ');
                    fig = figure('Units','normalized','
                       OuterPosition',[0 0 1 1],'Name',titulo,
                       'NumberTitle','off','PaperUnits','
                       centimeters'); % 'PaperOrientation','
                       landscape'
```

```
save_fig = true;
legends = {};

hold on
grid on
for mm=1:numEXP
    try
        expline = EXP{mm}.centerline.(
            current_plane).(current_type).(
            current_statistic).(
            current_direction).centerline;
        expx = expline(:,1);
        expy = expline(:,2);
        %marker_vector = round(linspace(1,
            length(expy),30));
        plot(expx,expy,...
            'LineStyle',exp_line_style,...
            'Color','k',...
            'marker',exp_markers(rem(mm-1,
                lene)+1),...
            'MarkerSize',exp_marker_size
                ,...'MarkerIndices',
                marker_vector,...
            'LineWidth',exp_line_width);
        legends{end+1} = legend_exp{mm};
    catch
        disp('error plot exp centerline');
    end
end
for mm=1:numSIM
    %fieldx = strcat('SIM{mm}.centerline
        .',current_type,'.coords');
    %fieldy = strcat('SIM{mm}.centerline
        .',current_type,'.',
        current_statistic,'.',
        current_direction,'.line');
    try
        simx = SIM{mm}.centerline.(
            current_plane).(current_type).
            coords;
```

```
simy = SIM{mm}.centerline.(
    current_plane).(current_type).(
    current_statistic).(
    current_direction).centerline;
marker_vector = round(linspace(1,
    length(simy),primes(rem(mm-1,
    lenp)+1)));
plot(simx,simy,...
     'LineStyle',line_style(rem(mm
         -1,lenl)+1,:),...
     'Color',colors(rem(mm-1,lenc)
         +1),...
     'marker',sim_markers(rem(mm-1,
         lens)+1),...
     'markersize',sim_marker_size
         ,...
     'MarkerIndices',marker_vector
         ,...
     'LineWidth',sim_line_width);
legends{end+1} = legend_sim{mm};

%Numeric comparison %%%%%%%
if plotting.compare_flag
    for nn=1:numEXP
        expline = EXP{nn}.
            centerline.(
            current_plane).(
            current_type).(
            current_statistic).(
            current_direction).
            centerline;
        expx = expline(:,1);
        expy = expline(:,2);

        %Interpolation
        simy2 = interp1(simx,simy,
            expx,'linear','extrap')
            ;
        centerline.(current_type)(
            nn,mm) = mean(abs(expy-
```

```
                        simy2)./(abs(expy)+abs(
                        simy2)));  %%%%%%
                end
                disp(['centerline ',
                    current_type]);
                disp(centerline.(current_type)
                    );
            end
        catch
            disp('error plot sim centerline');
        end
    end

    % if error save_fig = false

    % Axis properties
    %limits = eval(['struct_type.limits.',
        current_statistic,'.',current_direction
        ]);
    axis(struct_type.limits.(current_statistic
        ).(current_direction));
    %warning(''); % Clear last warning message
    %title(strcat('\bf\fontsize{',num2str(
        stations_fontsize),'}Station z = ',{'
        '},current_station,'mm'));
    %pbaspect(plotting.velocities_aspect_ratio
        );

    % Legend
    legend_str = strcat('\bf\fontsize{',
        num2str(legend_fontsize),'}',strrep(
        legends,'_','\_'));
    legend(legend_str);

    % Labels
    xlab = ['\bf\fontsize{',num2str(
        xlabel_fontsize),'} Centerline Distance
        z [m]'];
    xlabel(xlab);
```

```
                    ylabCell = [{current_statistic},{
                        current_direction},file_end,{y_units}];
                    ylab = ['\bf\fontsize{',num2str(
                        ylabel_fontsize),'}␣',strjoin(cellfun(
                        @upperFirst,ylabCell,'un',0),'␣')];
                    ylabel(ylab);

                    %Save graphics
                    if save_flag && save_fig

                        plotting.paperSize = [21 12]; %21 14.8
                        plotting.paperPos = [0 0.2 plotting.
                            paperSize];
                        plotting.results_file = strjoin(
                            titCell,'_');

                        plotSave(fig,plotting);
                    end
                end
            end
        else
            %%%%%%%%%%%%%%%%%%%
        end
    end
end

end
```

B.15. PLOTCONTOURS

codigo/functions/plot/plotContours.m

```
% 16/05/2018
% Carlos Moreno Montagud

function plotContours(plotting)

%% Variables
% Plotting
%EXP = plotting.EXP; numEXP = length(EXP);
```

```
SIM = plotting.SIM; numSIM = length(SIM);
save_flag = plotting.save_flag;
%save_format = plotting.save_format;
%legend_exp = plotting.legend_exp;
legend_sim = plotting.legend_sim;

%Config
config = plotConfig();

%sim_markers = config.sim_markers; lens = length(sim_markers)
    ;
%exp_markers = config.exp_markers; lene = length(exp_markers)
    ;
%line_style = config.line_style; lenl = length(line_style);
%colors = config.colors; lenc = length(colors);
%primes = config.primes; lenp = length(primes);

%exp_marker_size = config.exp_marker_size;
%exp_line_width = config.exp_line_width;
%exp_line_style = config.exp_line_style;
%sim_marker_size = config.sim_marker_size;
%sim_line_width = config.sim_line_width;

%legend_fontsize = config.contours.legend_fontsize;
title_fontsize = config.contours.title_fontsize;
xlabel1_fontsize = config.contours.xlabel1_fontsize;
ylabel1_fontsize = config.contours.ylabel1_fontsize;
xlabel2_fontsize = config.contours.xlabel2_fontsize;
ylabel2_fontsize = config.contours.ylabel2_fontsize;

cmap = config.cmap;

%% Plot
type = plotting.contours.type;
nType = length(type);
for ii=1:nType % 'velocities', 'tke'
    current_type = type{ii};
    struct_type = eval(['plotting.contours.',current_type]);

    file_end = struct_type.file_end;
```

```
y_units = struct_type.y_units;
title_end = struct_type.title_end; %%%%%%%%%%%%%

plane = struct_type.plane;
nPlane = length(plane);
for jj=1:nPlane % 'xz'
    current_plane = plane{jj};

    % Figure %%%%%%%%%%%%%%%
    if all(isfield(struct_type,{'statistic' 'directions'})
       ) %%%%%%%%%%%%%%%%%%%%%
        statistic = struct_type.statistic;
        nStatistic = length(statistic);

        directions = struct_type.directions;
        nDirections = length(directions);

        for mm=1:numSIM % case
            % Figure
            titulo = [upperFirst(current_type) ' Contours
               ' legend_sim{mm}];
            fig = figure('Units','normalized','
               OuterPosition',[0 0 1 1],'Name',titulo,'
               NumberTitle','off','PaperUnits','
               centimeters'); % 'PaperOrientation','
               landscape'
            save_fig = true;
            legends = {};

            for kk=1:nStatistic % 'mean', 'RMS'
                current_statistic = statistic{kk};

                for ll=1:nDirections % 'axial', 'radial',
                   'tangential'
                    current_direction = directions{ll};

                    index = sub2ind([nStatistic
                       nDirections],kk,ll);
                    subAxPlot(index) = subplot(nDirections
                       ,nStatistic,index);
```

```
                %fieldx = strcat('SIM{mm}.contours.',
                    current_plane,'.',current_type,'.
                    coords');
                %fieldy = strcat('SIM{mm}.contours.',
                    current_plane,'.',current_type,'.',
                    current_statistic,'.',
                    current_direction,'.contour');
                try
                    simx = SIM{mm}.contours.(
                        current_plane).(current_type).
                        coords;
                    simy = SIM{mm}.contours.(
                        current_plane).(current_type).(
                        current_statistic).(
                        current_direction).contour;
                    imagesc(simx(:,1),simx(:,2),simy);
                    axis xy
                    colormap(cmap);
                    colorbar('northoutside');

                    %Legend and labels
                    %title(['\bf\fontsize{',num2str(
                        title_fontsize),'} ',upperFirst
                        (current_statistic) ' '
                        upperFirst(current_direction) '
                         ' title_end]);
                    %xlabel(['\bf\fontsize{',num2str(
                        xlabel1_fontsize),'} Distance
                        ',current_plane(1),' [m]']);
                    %ylabel(['\bf\fontsize{',num2str(
                        ylabel1_fontsize),'} Distance
                        ',current_plane(2),' [m]']);
                    pbaspect(struct_type.aspect_ratio)
                        ;
                end
            end
        end

        %Labels
```

```
            p1 = get(subAxPlot(1),'position');   % position
                 of upper-left axes
            p2 = get(subAxPlot(end),'position'); %
                position of lower-right axes
            xtit = strcat('\bf\fontsize{',num2str(
                title_fontsize),'}␣',cellfun(@upperFirst,
                statistic,'un',0),{'␣'},title_end,{'␣'},
                y_units);
            ytit = strcat('\bf\fontsize{',num2str(
                title_fontsize),'}␣',cellfun(@upperFirst,
                directions,'un',0));
            xlab = ['\bf\fontsize{',num2str(
                xlabel1_fontsize),'}␣Distance␣',
                current_plane(1),'␣[m]'];
            ylab = ['\bf\fontsize{',num2str(
                ylabel1_fontsize),'}␣Distance␣',
                current_plane(2),'␣[m]'];
            subplotLabels(nDirections,nStatistic,p1,p2,
                xlab,ylab,xtit,ytit);

            %Save graphics figure %%%%%
            if save_flag && save_fig

                %Properties
                subWidth = 9.9;
                subHeight = 9.9;
                offset = [0 -0.5];

                plotting.paperSize = [subWidth*nStatistic
                    subHeight*nDirections];
                plotting.paperPos = [offset subWidth*
                    nStatistic subHeight*nDirections+1];
                plotting.results_file = strcat(legend_sim{
                    mm},'_','contour','_',file_end);

                plotSave(fig,plotting);
            end
        end
    else
        for mm=1:numSIM
```

```
titulo = [title_end ' ' legend_sim{mm}];
fig = figure('Units','normalized','
   OuterPosition',[0 0 1 1],'Name',titulo,'
   NumberTitle','off','PaperUnits','
   centimeters'); %'PaperOrientation','
   landscape'
save_fig = true;
legends = {};
fieldx = strcat('SIM{mm}.contours.',
   current_plane,'.',current_type,'.coords');
fieldy = strcat('SIM{mm}.contours.',
   current_plane,'.',current_type,'.contour');
try
    simx = eval(fieldx);
    simy = eval(fieldy);
    imagesc(simx(:,1),simx(:,2),simy);
    axis xy
    colormap(cmap);
    colorbar

    %Legend and labels
    title(['\bf\fontsize{',num2str(
       title_fontsize),'} ',title_end]);
    xlabel(['\bf\fontsize{',num2str(
       xlabel2_fontsize),'} Distance ',
       current_plane(1),' [m]']);
    ylabel(['\bf\fontsize{',num2str(
       ylabel2_fontsize),'} Distance ',
       current_plane(2),' [m]']);
    pbaspect(struct_type.aspect_ratio);

    %Save graphics figure
       %%%%%%%%%%%%%%%%%%%%
    if save_flag && save_fig

        %Properties
        subWidth = 21;
        subHeight = 14.8;
        offset = [0.4 0.4];
```

```
                        plotting.paperSize = [subWidth
                            subHeight];
                        plotting.paperPos = [offset subWidth
                            -0.5 subHeight-0.5];
                        plotting.results_file = strcat(
                            legend_sim{mm},'_','contour','_',
                            file_end);

                        plotSave(fig,plotting);
                    end
                end
            end
        end
    end
end

end
```

B.16. PLOTSAVE

codigo/functions/plot/plotSave.m

```
%31/05/2018
%Carlos Moreno Montagud

function plotSave(fig,plotting)

results_folder = plotting.results_folder;
results_file = plotting.results_file;
results_filename = strcat(results_folder,results_file);
save_format = plotting.save_format;

paperSize = plotting.paperSize;
paperPos = plotting.paperPos;
set(fig,'PaperSize',paperSize,'PaperPosition',paperPos);

%Make dir if it does not exist
if ~exist(results_folder,'dir')
    mkdir(results_folder)
end
```

```
% Export to all formats needed
for mm=save_format
    format = char(mm);
    writeFile = strcat(results_filename,'.',format(1:3));
    writeSearch = strcat(results_filename,'*.',format(1:3));

    % Check if already exist
    if exist(writeFile,'file')
        files = dir(writeSearch);
        filename = strrep(files(end).name,['.' format(1:3)],''
            );
        version = str2double(filename(end));
        if isnan(version)
            version = 0;
        end
        % Versions from 1-10. Further versions overwrite 10
        results_filename = strcat(results_filename,num2str(
            version+1));
    end

    % Save figure
    format = strcat('-d',format);
    print(results_filename,format);
    % print(results_filename,'-dtiff','-r400') % resolution
end

end
```

B.17. PLOTSTATIONSSYM

codigo/functions/plot/plotStationsSym.m

```
% 15/05/2018
% Carlos Moreno Montagud

function stations = plotStationsSym(plotting)

%% Variables
% Plotting
```

```
EXP = plotting.EXP; numEXP = length(EXP);
SIM = plotting.SIM; numSIM = length(SIM);
save_flag = plotting.save_flag;
%save_format = plotting.save_format;
legend_exp = plotting.legend_exp;
legend_sim = plotting.legend_sim;

stations = [];

% Config
config = plotConfig();

sim_markers = config.sim_markers; lens = length(sim_markers);
exp_markers = config.exp_markers; lene = length(exp_markers);
line_style = config.line_style; lenl = length(line_style);
colors = config.colors; lenc = length(colors);
primes = config.primes; lenp = length(primes);

exp_marker_size = config.exp_marker_size;
exp_line_width = config.exp_line_width;
exp_line_style = config.exp_line_style;
sim_marker_size = config.sim_marker_size;
sim_line_width = config.sim_line_width;

stations_fontsize = config.stations.stations_fontsize;
legend_fontsize = config.stations.legend_fontsize;
xlabel_fontsize = config.stations.xlabel_fontsize;
ylabel_fontsize = config.stations.ylabel_fontsize;

%% Plot
type = plotting.stations.type;
nType = length(type);
for ii=1:nType % 'velocities', 'species'
    current_type = type{ii};

    struct_type = plotting.stations.(current_type);
    file_end = struct_type.file_end;
    y_units = struct_type.y_units;

    plane = struct_type.plane;
```

```
nPlane = length(plane);
for jj=1:nPlane % 'xz'
    current_plane = plane{jj};

    if all(isfield(struct_type,{'statistic' 'directions'})
       ) %%%%%%%%%%%%%%%%%%%%%
        statistic = struct_type.statistic; % 'mean', 'RMS'
        nStatistic = length(statistic);

        directions = struct_type.directions;
        nDirections = length(directions);
        mean_d_temp = zeros(numEXP,numSIM,nDirections);
        RMS_d_temp = zeros(numEXP,numSIM,nDirections);
        for kk=1:nDirections % 'axial', 'radial', '
           tangential', 'CH4'
            current_direction = directions{kk};

            titCell = {current_direction,file_end{:}};
            titulo = strjoin(cellfun(@upperFirst,titCell,'
               un',0),'␣');
            fig = figure('Units','normalized','
               OuterPosition',[0 0 1 1],'Name',titulo,'
               NumberTitle','off','PaperUnits','
               centimeters'); % 'PaperOrientation','
               landscape'
            save_fig = true;


            station_nums = struct_type.stations;
            nStations = length(station_nums);
            mean_s_temp = zeros(numEXP,numSIM,nStations);
            RMS_s_temp = zeros(numEXP,numSIM,nStations);
            for ll=1:nStations %[5 10 20 30 40]
                current_station = num2str(station_nums(ll)
                   );

                plot_index = nStations+2-ll;
                subAxPlot(plot_index) = subplot(nStations
                   +1,1,plot_index);
                legends = {};
```

```
hold on
grid on
for mm=1:numEXP
    try
        meann = EXP{mm}.stations.(
            current_plane).(current_type).(
            statistic{1}).(
            current_direction).(['station_'
            ,current_station,'mm']);
        mean_expx = -meann(:,1); %
            %%%%%%%%%%
        mean_expy = meann(:,2);
        RMS = EXP{mm}.stations.(
            current_plane).(current_type).(
            statistic{2}).(
            current_direction).(['station_'
            ,current_station,'mm']);
        RMS_expx = RMS(:,1);
        RMS_expy = RMS(:,2);

        %mean
        yyaxis left
        %marker_vector = round(linspace(1,
            length(mean_expy),30));
        plot(mean_expx,mean_expy,...
            'LineStyle',exp_line_style,...
            'Color','k',...
            'marker',exp_markers(rem(mm-1,
                lene)+1),...
            'MarkerSize',exp_marker_size
                ,...'MarkerIndices',
                marker_vector,...
            'LineWidth',exp_line_width);

        %RMS
        yyaxis right
        %marker_vector = round(linspace(1,
            length(RMS_expy),30));
        plot(RMS_expx,RMS_expy,...
```

```
                    'LineStyle',exp_line_style,...
                    'Color','k',...
                    'marker',exp_markers(rem(mm-1,
                        lene)+1),...
                    'MarkerSize',exp_marker_size
                        ,...'MarkerIndices',
                        marker_vector,...
                    'LineWidth',exp_line_width);
                legends{end+1} = legend_exp{mm};
            end
        end
        for mm=1:numSIM
            try
                simx = SIM{mm}.stations.(
                    current_plane).(current_type).
                    coords;
                mean_simx = -simx(simx>=0);
                RMS_simx = simx(simx>=0);
                simy1 = SIM{mm}.stations.(
                    current_plane).(current_type).(
                    statistic{1}).(
                    current_direction).(['station_'
                    ,current_station,'mm']);
                mean_simy = simy1(simx>=0);
                simy2 = SIM{mm}.stations.(
                    current_plane).(current_type).(
                    statistic{2}).(
                    current_direction).(['station_'
                    ,current_station,'mm']);
                RMS_simy = simy2(simx>=0);

                %mean
                yyaxis left
                marker_vector = round(linspace(1,
                    length(mean_simy),primes(rem(mm
                    -1,lenp)+1)));
                plot(mean_simx,mean_simy,...
                    'LineStyle',line_style(rem(mm
                        -1,lenl)+1,:),...
```

```
        'Color',colors(rem(mm-1,lenc)
            +1),...
        'marker',sim_markers(rem(mm-1,
            lens)+1),...
        'markersize',sim_marker_size
            ,...
        'MarkerIndices',marker_vector
            ,...
        'LineWidth',sim_line_width);

    %RMS
    yyaxis right
    marker_vector = round(linspace(1,
        length(RMS_simy),primes(rem(mm
        -1,lenp)+1)));
    plot(RMS_simx,RMS_simy,...
        'LineStyle',line_style(rem(mm
            -1,lenl)+1,:),...
        'Color',colors(rem(mm-1,lenc)
            +1),...
        'marker',sim_markers(rem(mm-1,
            lens)+1),...
        'markersize',sim_marker_size
            ,...
        'MarkerIndices',marker_vector
            ,...
        'LineWidth',sim_line_width);
    legends{end+1} = legend_sim{mm};

    %Numeric comparison
    if plotting.compare_flag
        for nn=1:numEXP
            meann = EXP{nn}.stations.(
                current_plane).(
                current_type).(
                statistic{1}).(
                current_direction).(['
                station_',
                current_station,'mm']);
```

```
mean_expx = -meann(:,1); %
    %%%%%%%%%%
mean_expy = meann(:,2);
RMS = EXP{nn}.stations.(
    current_plane).(
    current_type).(
    statistic{2}).(
    current_direction).(['
    station_',
    current_station,'mm']);
RMS_expx = RMS(:,1);
RMS_expy = RMS(:,2);

%mean
mean_simy2 = interp1(
    mean_simx,mean_simy,
    mean_expx,'linear','
    extrap');
stations.(current_type).
    mean.(current_direction
    ).(['station_',
    current_station,'mm'])(
    nn,mm) = mean(abs(
    mean_expy-mean_simy2)
    ./(abs(mean_expy)+abs(
    mean_simy2)));  %%%
mean_s_temp(nn,mm,ll) =
    stations.(current_type)
    .mean.(
    current_direction).(['
    station_',
    current_station,'mm'])(
    nn,mm);

%RMS
RMS_simy2 = interp1(
    RMS_simx,RMS_simy,
    RMS_expx,'linear','
    extrap');
```

```
                                    stations.(current_type).
                                        RMS.(current_direction)
                                        .(['station_',
                                        current_station,'mm'])(
                                        nn,mm) = mean(abs(
                                        RMS_expy-RMS_simy2)./(
                                        abs(RMS_expy)+abs(
                                        RMS_simy2)));  %%%
                                    RMS_s_temp(nn,mm,ll) =
                                        stations.(current_type)
                                        .RMS.(current_direction
                                        ).(['station_',
                                        current_station,'mm'])(
                                        nn,mm);
                                end

                                if ll==nStations
                                    stations.(current_type).
                                        mean.(current_direction
                                        ).total = mean(
                                        mean_s_temp,3);
                                    mean_d_temp(:,:,kk) =
                                        stations.(current_type)
                                        .mean.(
                                        current_direction).
                                        total;
                                    stations.(current_type).
                                        RMS.(current_direction)
                                        .total = mean(
                                        RMS_s_temp,3);
                                    RMS_d_temp(:,:,kk) =
                                        stations.(current_type)
                                        .RMS.(current_direction
                                        ).total;

%                                       disp(['stations [mean
   RMS] ' current_type ' ' current_direction]);
%                                       disp([stations.(
   current_type).mean.(current_direction).total,stations.(
   current_type).RMS.(current_direction).total]);
```

```
                                    if kk==nDirections
                                        stations.(current_type
                                            ).mean.total = mean
                                            (mean_d_temp,3);
                                        stations.(current_type
                                            ).RMS.total = mean(
                                            RMS_d_temp,3);

                                        disp(['stations ' ...
                                            current_type ' [...
                                            mean RMS]']);
                                        disp([stations.(...
                                            current_type).mean.
                                            total,stations.(...
                                            current_type).RMS.
                                            total]);
                                    end
                                end

%                                   disp(['stations [mean RMS] ' ...
    current_type ' ' current_direction ' ' current_station 'mm...
    ']);
%                                   disp([stations.(current_type...
    ).mean.(current_direction).(['station_',current_station,'mm...
    ']) stations.(current_type).RMS.(current_direction).(['...
    station_',current_station,'mm'])]);
                                end
                            end
                        end

                        %if error save_fig = false

                        %Axis properties
                        %eval(strcat('limits = plotvar.axis.',...
                            velDir,';'));
                        xlim(struct_type.limits.xlimit);
                        yyaxis left
                        ylim(struct_type.limits.mean.(...
                            current_direction));
```

```
        yyaxis right
        ylim(struct_type.limits.RMS.(
            current_direction));
        %warning(''); % Clear last warning message
        title(strcat('\bf\fontsize{',num2str(
            stations_fontsize),'}Station z = ',{' '
            },current_station,'mm'));
        %pbaspect(plotting.velocities_aspect_ratio
            );
    end

    % Legend
    legends = strcat('\bf\fontsize{',num2str(
        legend_fontsize),'}',strrep(legends,'_','\_
        '));
    lg = legend(legends,'Orientation','horizontal'
        ); % create legend
    lpos = get(lg,'position'); % get legend
        position
    subAxPlot(1) = subplot(nStations+1,1,1); %
        create subplot
    set(subAxPlot(1),'Visible', 'off'); % hide
        subplot
    pos = get(subAxPlot(1),'position'); % get
        subplot position
    set(lg,'position',[pos(1)+(pos(3)-lpos(3))/2
        pos(2)+pos(4)/4-lpos(4) lpos(3) lpos(4)*2])
        ;

    % Labels
    p1 = get(subAxPlot(2),'position');   % position
         of upper-left axes
    p2 = get(subAxPlot(end),'position'); %
        position of lower-right axes
    posOuter = [p1(1) p2(2) p2(1)+p2(3)-p1(1)  p1
        (2)+p1(4)-p2(2)];
    hAxOuter = axes('position',posOuter,'color','
        none','visible','off'); % an outer axis for
         global p
```

```
            xlab = ['\bf\fontsize{',num2str(
                xlabel_fontsize),'}␣Radial␣Distance␣',
                current_plane(1),'␣[m]'];
            hX=text(0.5,-0.05,xlab,'rotation',0,'
                horizontalalignment','center','
                verticalalignment','top');
            ylabCell = strjoin({titulo,y_units},'␣');
            ylabFirst = cellfun(@upperFirst,statistic,'un'
                ,0);
            ylab = strcat('\bf\fontsize{',num2str(
                ylabel_fontsize),'}',{'␣'},ylabFirst,{'␣'},
                ylabCell);
            hY1=text(-0.05,0.5,ylab{1},'rotation',90,'
                horizontalalignment','center','
                verticalalignment','bottom');
            hY2=text(1.05,0.5,ylab{2},'rotation',-90,'
                horizontalalignment','center','
                verticalalignment','bottom');

            %Save graphics figure
            if save_flag && save_fig

                %Properties
                subHeight = 6;
                subWidth = 25;
                offset = [-0.5 0];

                plotting.paperSize = [subWidth subHeight*(
                    nStations-1)];
                plotting.paperPos = [offset subWidth
                    subHeight*(nStations-3/4)];
                plotting.results_file = strjoin(titCell,'_
                    ');

                plotSave(fig,plotting);
            end
        end
    else
        %%%%%%%%%%%%%%%%%%%%
    end
```

```
    end
end

end

%% Secondary functions
function str = upperFirst(data)
str = [upper(data(1)) data(2:end)];
end
```

B.18. SUBPLOTLABELS

codigo/functions/plot/subplotLabels.m

```
% 24/05/2018
% Carlos Moreno Montagud

function subplotLabels(m,n,p1,p2,xlab,ylab,xtit,ytit)

posOuter = [p1(1) p2(2) p2(1)+p2(3)-p1(1) p1(2)+p1(4)-p2(2)];
hAxOuter = axes('position',posOuter,'color','none','visible','
    off'); %an outer axis for global p

x_pos_x = ([1:n]*2-1)/(2*n)+linspace(-1,1,n)*0.05;
xlab_pos_y = repmat(-0.02,size(x_pos_x));
xtit_pos_y = repmat(1.02,size(x_pos_x));
text(x_pos_x,xlab_pos_y,xlab,'rotation',0,'horizontalalignment
    ','center','verticalalignment','top');
text(x_pos_x,xtit_pos_y,xtit,'rotation',0,'horizontalalignment
    ','center','verticalalignment','top');

y_pos_y = ([1:m]*2-1)/(2*m)-([m:-1:1]-0.9)*0.04;
ylab_pos_x = repmat(0,size(y_pos_y));
ytit_pos_x = repmat(0.95,size(y_pos_y));
text(ylab_pos_x,y_pos_y,ylab,'rotation',90,'
    horizontalalignment','center','verticalalignment','bottom')
    ;
text(ytit_pos_x,y_pos_y(end:-1:1),ytit,'rotation',-90,'
    horizontalalignment','center','verticalalignment','bottom')
    ;
```

```
end
```

Documento II

PLIEGO DE CONDICIONES

Índice de documento

Capítulo 1

Introducción

En este apartado se detalla la normativa de las ordenanzas de seguridad e higiene en el trabajo, en aquellos artículos que se deberían tener en cuenta para la realización del presente Trabajo de Fin de Máster.

Capítulo 2

Obligaciones y derechos de los trabajadores

Establece, para los trabajadores, la obligación de cooperar en la prevención de riesgos profesionales en la empresa y el mantenimiento de la máxima higiene en la misma, a cuyos fines deberán cumplir fielmente los preceptos de esta Ordenanza y de sus instrucciones complementarias, así como las órdenes e instrucciones que a tales efectos les sean dados por sus superiores.

Los trabajadores, expresamente, están obligados a:

- Recibir las enseñanzas sobre materia en Seguridad e Higiene y sobre salvamento y socorrismo en los centros de trabajo que les sean facilitados por la empresa o en las instrucciones del Plan Nacional.
- Usar correctamente los medios de protección personal y cuidar de su perfecto estado de conservación.
- Dar cuenta inmediatamente a sus superiores de las averías y las deficiencias que puedan ocasionar peligros en cualquier centro o puesto de trabajo.
- Cuidar y mantener su higiene personal para evitar enfermedades contagiosas o molestias a los compañeros de trabajo.
- Someterse a los reconocimientos médicos preceptivos y vacunaciones o inmunizaciones ordenados por las Autoridades Sanitarias competentes o por el Servicio Médico de las Empresas.
- No introducir bebidas y otras sustancias no autorizadas en los centros de trabajo. Tampoco se podrá presentar o permanecer en los mismos en estado de embriaguez o de cualquier otro género de intoxicación.
- Cooperar en la extinción de siniestros y en el salvamento de las víctimas de accidentes de trabajo en las condiciones que, en cada caso, fueren racionalmente exigibles.

- Todo trabajador, después de solicitar de su inmediato superior medios de protección personal de carácter preceptivo para la realización de su trabajo, queda facultado para demostrar la ejecución de éste, en tanto no le sean facilitados dichos medios, si bien debería dar cuenta del hecho al Comité de Seguridad e Higiene o a uno de sus compañeros, sin perjuicio, además de ponerlo en conocimiento de la Inspección Provincial del Trabajo.

Capítulo 3

Seguridad estructural

Todos los edificios permanentes o provisionales, serán de construcción segura y firme para evitar riesgos de desplome y los derivados de los fenómenos atmosféricos. Por este motivo, los cimientos, pisos y demás elementos de los edificios ofrecerán resistencia suficiente para sostener y suspender con seguridad las cargas para las que han sido calculados. Además, para preservar esta seguridad, se indicará por medio de rótulos o inscripciones las cargas que los locales pueden soportar o suspender, quedando totalmente prohibido sobrecargar los pisos y plantas de los edificios.

Capítulo 4

Superficie y cubicación

Los locales de trabajo reunirán las siguientes condiciones mínimas: Tres metros de altura desde el piso al techo. Dos metros cuadrados de superficie por cada trabajador. Diez metros cúbicos para cada trabajador. No obstante, en los establecimientos comerciales, de servicios y locales destinados a oficinas y despachos, la altura a la que se refiere el primer punto puede quedar reducida hasta 2:5 m, pero respetando la cubicación que establece el tercer punto, y siempre que el aire se renueve suficientemente. Para el cálculo de la superficie y el volumen, no se tendrán en cuenta los espacios ocupados por máquinas, aparatos, instalaciones y materiales.

Capítulo 5

Suelos, techos y paredes

El pavimento constituirá un conjunto homogéneo, llano y liso, sin soluciones de continuidad; será de material consistente, no resbaladizo o susceptible de serlo con el uso y además, de fácil limpieza. Estará a un mismo nivel; de no ser así, se salvarán las diferencias de altura por medio de rampas con pendientes no superiores al 10 Por otro lado, tanto los techos como las paredes, deberán reunir las condiciones suficientes para resguardar a los trabajadores de las inclemencias del tiempo. Si han de soportar o suspender cargas, deberán reunir las condiciones que se establecen para estos en el apartado correspondiente.

Capítulo 6

Disposiciones generales

Todos los lugares de trabajo o tránsito tendrán iluminación natural, artificial o mixta, en cualquier caso, apropiado a las operaciones que en ellos se tengan que realizar. Aunque la luz, como se acaba de comentar, puede ser natural o artificial, se intentará que sea natural, en medida de lo posible. Se deberá intensificar la iluminación en máquinas peligrosas, lugares de tránsito con riesgo de caídas, escaleras y salidas de emergencia.

Capítulo 7

Iluminación de emergencia

En todos los centros de trabajo, se dispondrá de medios de iluminación de emergencia adecuados a las dimensiones de los locales y al número de trabajadores ocupados simultáneamente, capaces de mantener, al menos durante una hora, la intensidad de cinco luxes y mediante una fuente de energía independiente del sistema normal de iluminación.

Capítulo 8

Ventilación, temperatura y humedad

En los lugares de trabajo y sus anexos se mantendrá, por medios naturales o artificiales, unas condiciones atmosféricas adecuadas, evitando el aire viciado, exceso de calor o de frío, humedad o sequía y los olores desagradables. En ningún caso, el anhídrido carbónico ambiental, podrá sobrepasar la proporción de 50/10000, y el monóxido de carbono la de 1/10000. En los locales de trabajo cerrados, el suministro de aire fresco y limpio por hora y por trabajador, será de al menos $30m^3$, salvo que se efectúe una renovación total del aire varias veces por hora, no inferior a seis veces para trabajos sedentarios, ni a diez veces para trabajos que exijan un esfuerzo físico superior al normal. En el otro extremo, la circulación de aire en locales cerrados se acondicionará de modo que los trabajadores no estén expuestos a corrientes molestas y que la velocidad del aire no exceda de $15m/min$ con temperatura normal, ni de $45m/min$ en ambientes extremadamente calurosos. En los centros de trabajo expuestos a altas y bajas temperaturas, serán evitadas las variaciones bruscas por el medio que se considere más eficaz. Cuando la temperatura sea extremadamente distinta entre los lugares de trabajo, deberán existir locales de paso para que los operarios se adapten gradualmente de unas condiciones a las otras. De acuerdo con todo lo anterior, se fijan como límites de temperatura y humedad en locales y para los distintos trabajos, siempre que el procedimiento de fabricación lo permita, los siguientes:

- Para trabajos sedentarios: de 17°C a 22°C.
- Para trabajos ordinarios: de 15°C a 18°C.
- Para trabajos de acusado esfuerzo muscular: de 12°C a 15°C.

A pesar de estas limitaciones, todos los trabajadores estarán debidamente protegidos contra las irradiaciones directas y excesivas de calor. La humedad relativa de la atmósfera oscilará del 40 al 60 %, salvo en instalaciones en las que haya peligro de generarse electricidad estática. En dicho caso, el valor se deberá limitar necesariamente por

debajo del 50 %. En aquellos trabajos en los que por exigencias del proceso los locales estén sometidos a calor o frío extremo, se eliminará la permanencia de los operarios, estableciendo los turnos adecuados en cada caso.

Capítulo 9

Ruidos, vibraciones y trepidaciones

Los ruidos y vibraciones se evitarán o reducirán en la medida de lo posible en su foco de origen, tratando de aminorar su propagación en los locales en los que se encuentren personas trabajando. De esta forma, el anclaje de las máquinas y aparatos que produzcan ruidos, vibraciones y trepidaciones se realizarán con las técnicas más eficaces, a fin de lograr su óptimo equilibrio estático y dinámico, tales como bancadas cuyo peso sea superior a 1.5 y 2.5 veces el peso de la máquina que soportan, por aislamiento de la estructura general o por otros recursos técnicos. Además del anclaje, las máquinas que produzcan ruidos o vibraciones molestas se aislarán adecuadamente y, en el recinto de aquellas, sólo trabajará el personal para su mantenimiento durante el tiempo indispensable. Se extremará el cuidado y mantenimiento de las máquinas y aparatos que produzcan vibraciones molestas o peligrosas a los trabajadores y, muy especialmente, los órganos móviles y los dispositivos de transmisión de movimiento de las vibraciones que generan estas máquinas. El control de ruidos agresivos en los centros de trabajo no se limitará al aislamiento del foco que los produce, sino que también deberán adoptarse las prevenciones técnicas necesarias para evitar que los fenómenos de reflexión y resonancia alcancen niveles peligrosos para la salud de los trabajadores. Las máquinas y/o herramientas que originen trepidaciones deberán, además, estar provistas de horquillas u otros dispositivos amortiguadores y, al trabajador que las utilice se le proveerá de equipo de protección personal antivibratorio (cinturón, guantes, almohadillas y botas).

Capítulo 10

Protección contra contactos en equipos eléctricos

En las instalaciones y equipos eléctricos, para la protección de las personas contra los contactos con partes habitualmente en tensión, se adoptaran algunas de las siguientes prevenciones:

- Se alejarán las partes activas de la instalación a distancia suficiente del lugar donde las personas habitualmente se encuentran o circulan, para evitar un contacto fortuito o por la manipulación de objetos conductores, cuando éstos puedan ser utilizados cerca de estas partes activas de la instalación.
- Se recubrirán las partes activas con el aislamiento apropiado, que permita conservar indefinidamente las propiedades del conductor y que limiten la corriente de contacto a un valor inocuo para las personas.
- Se interpondrán obstáculos que impidan todo contacto accidental con las partes activas de la instalación. Los obstáculos de protección deben estar fijados en forma segura y ser capaces de resistir los esfuerzos mecánicos usuales.

Para la protección contra los riesgos de contacto con las masas de las instalaciones que puedan qudar accidentalmente con tensión, se adoptarán, en corriente alterna, uno o varios de los siguientes dispositivos de seguridad.

- Puesta a tierra de las masas. Las masas deben estar unidas eléctricamente a una toma de tierra o a un conjunto de tomas de tierra interconectadas, que tengan una resistencia apropiada. Las instalaciones, tanto con neutro aislado como con neutro a tierra, deben estar permanentemente controladas por un dispositivo que indique automáticamente la existencia de cualquier defecto de aislamiento, o que separe automáticamente la instalación o la parte defectuosa de la fuente de energía de la que se alimenta.

- Dispositivos de corte automático o de aviso, sensibles a la corriente de defecto (interruptores diferenciales), o a la tensión de defecto (relés de tierra).
- Unión equipotencial o por superficie aislada de tierra o de las masas (conexiones equipotenciales).
- Separación de los circuitos de utilización de las fuentes de energía, por medio de transformadores o grupos convertidores, manteniendo aislados de tierra todos los conductores del circuito de utilización, incluido el neutro.
- Poner doble aislamiento de los equipos y máquinas eléctricas.

En caso de que existan circuitos alimentados mediante corriente contínua, se adoptarán sistemas de protección adecuados para cada caso, similares a los que se acaban de mencionar para la corriente alterna.

Capítulo 11

Electricidad estática

Para evitar peligros por la acumulación de electricidad estática y, especialmente, aquellos que pueden venir propiciados por la producción de una chispa en ambientes inflamables, se adoptarán en general una o incluso ambas precauciones que a continuación se especifican:

- La humedad relativa del aire se mantendrá siempre con un valor por debajo del 50 %.
- Las cargas de electricidad estática que puedan acumularse en los cuerpos metálicos, serán neutralizadas por medio de la conexión de conductores a tierra. La forma de realizar estas conexiones puede variar dependiendo del tipo de máquina.

Capítulo 12

Recomendaciones sobre materias inflamables

Con respecto al almacenamiento y a la manipulación de las sustancias inflamables, dada su peligrosidad, estas serán las precauciones que se tendrán que cumplir para evitar cualquier accidente:

- Se prohíbe el almacenamiento conjunto de materiales que al reaccionar entre ellos puedan originar incendios.
- Sólo podrán almacenarse materias inflamables en los lugares y con los límites cuantitativos señalados por los reglamentos técnicos vigentes. De esta forma, se almacenarán en locales distintos a los de trabajo.Si el local de trabajo es único, se deberán construir recintos aislados. En los puestos de trabajo, sólo se depositará la cantidad estrictamente necesaria para el proceso de fabricación.
- En los almacenes de materias inflamables, los pisos deberán ser incombustibles e impermeables, a fin de evitar la propagación del posible fuego y evitar escapes hacia sótanos, sumideros o desagües.
- Antes de almacenar sustancias inflamables pulverizadas, se comprobará su enfriamiento.
- El llenado de los depósitos de líquidos inflamables se efectuará lentamente y evitando la caída libre desde orificios de la parte superior, para evitar la mezcla de los vapores explosivos con el aire. Estos recipientes de líquidos se rotularán, indicando su contenido y las precauciones necesarias para su manipulación.
- Antes de almacenar envases de productos inflamables, se comprobará su cierre hermético y si han sufrido algún deterioro o rotura, para evitar posibles fugas.
- El envasado y embalaje de sustancias inflamables se efectuará, siempre que sea posible, fuera de los almacenes de donde procedan, con las precauciones y equipos de protección adecuados a cada caso.

- El transporte de materias inflamables se efectuará con estricta sujeción a las normas fijadas en dispositivos legales vigentes y acuerdos internacionales sobre tal materia, ratificados por el Estado Español.

Capítulo 13

Prevención y extinción de incendios

En los centros de trabajo que ofrezcan peligro de incendios, con o sin explosión, se adoptarán las prevenciones que se indican a continuación, combinando su empleo con la protección general más próxima que puedan prestar los servicios públicos contra incendios:

- Donde existan conducciones de agua a presión, se instalarán suficientes tomas o bocas de agua, a distancia conveniente entre sí y cercanas a los puestos fijos de trabajo y lugares de paso del personal, colocando junto a las mismas las correspondientes mangueras, con la sección y resistencia adecuadas para soportar la presión.
- Cuando se carezca de agua a presión o sea insuficiente, se instalarán depósitos con volumen suficiente para poder combatir posibles incendios.
- En los incendios provocados por líquidos, grasas, pinturas inflamables o polvos orgánicos, sólo se debe emplear agua, además muy pulverizada.
- No se debe emplear agua para extinguir fuegos en polvos de aluminio o magnesio en presencia de carburo de calcio u otras sustancias que, en contacto con el agua, produzcan explosiones, gases inflamables o incluso nocivos.
- En incendios que afecten a instalaciones eléctricas con tensión, se prohibirá el empleo de extintores de espuma química, soda ácida o agua.
- En la proximidad a los puestos de trabajo con mayor riesgo de incendio se dispondrán, colocados en un sitio visible y accesible fácilmente, extintores portátiles o móviles sobre ruedas, de espuma física, química o mezcla de ambas, o polvos secos, anhídrido carbónico o agua, según convenga a la causa determinante del fuego a extinguir.

- Cuando se empleen distintos tipos de extintores deberán estar rotulados con carteles indicadores del lugar o clase de incendio en el que deban emplearse. Estos extintores serán revisados periódicamente y cargados según las normas de las casas constructoras, inmediatamente después de usarlos.
- Se instruirá al personal, cuando sea necesario, del peligro que presenta el empleo de tetracloruro de carbono y cloruro de metilo en atmósferas cerradas, y de las reacciones químicas peligrosas que puedan producirse en los locales de trabajo entre los líquidos extintores y las materias sobre las que puedan proyectarse.
- En las dependencias con alto riesgo de incendio, queda terminantemente prohibido fumar o introducir cerillas, mecheros o útiles de ignición. Esta prohibición se indicará con carteles visibles a la entrada y en los espacios libres de las paredes de tales dependencias.
- Se prohíbe, igualmente, al personal introducir o emplear útiles de trabajo no autorizados por la empresa, que puedan ocasionar chispas por contacto o proximidad a sustancias inflamables.
- Es obligatorio el uso de guantes, manoplas, mandiles o trajes ignífugos y de calzado especial contra incendios, que las empresas facilitarán a los trabajadores para uso individual.

Documento III

PRESUPUESTO

Índice de documento

Capítulo 1

Introducción

En este documento se va a realizar un estudio de los costes que lleva consigo la realización del presente proyecto. La moneda para la estimación de los costes en el presente documento es el euro (EUR).

El presupuesto necesario para abordar el presente trabajo puede desglosarse en dos partidas principales:

- Costes referidos a los recursos humanos: tienen en cuenta la participación de un becario, de un profesor ayudante en calidad de doctor y de un doctorando.
- Costes referidos a los equipos: en este caso particular, dado que el Instituto Universitario CMT-Centro de Motores Térmicos disponía con anterioridad de los equipos informáticos necesarios para el desarrollo del presente trabajo, únicamente se contabilizarán las amortizaciones en la utilización de los equipos.
- Costes generales: permiten incluir otras partidas asociadas a imprevistos, gestión y medios auxiliares.

Con todo, teniendo en cuenta que el Trabajo Final de Máster únicamente ha constado de un estudio computacional, se realizará un presupuesto parcial de la misma y un presupuesto total. Adicionalmente, se aplican los precios correspondientes a las tarifas legales vigentes en el caso de ser conocidas o aplicado estimaciones de no serlo, siempre manteniendo una coherencia en la misma. El presupuesto total se define como la suma de los presupuestos parciales, con un 50 % adicional en concepto de los medios auxiliares y/o costes imprevistos. Finalmente, se le aplica el IVA vigente. A continuación se muestran cada uno de los costes descritos anteriormente.

Capítulo 2

Costes referidos a los recursos humanos

En esta fase han participado como recursos humanos un total de tres profesionales:

- Becario Ingeniero Aeronáutico: encargado de lanzar simulaciones, programar las rutinas de posprocesado, analizar resultados, sacar conclusiones y proponer posibles soluciones.
- Doctorando Ingeniero Aeronáutico: propone la línea general de los estudios y simulaciones que hay que llevar a cabo, analiza resultados y ayuda con la búsqueda de soluciones. Su dedicación es alta ya que este trabajo forma parte de la línea de investigación concreta para una tesis doctoral.
- Doctor Ingeniero Aeronáutico: su labor es la de supervisar que todo el trabajo transcurre conforme a lo establecido. En cualquier situación problemática, ayuda a resolver la misma. Su dedicación es parcial.

Además de los recursos humanos, se deben incluir en este apartado los costes de amortización del equipo informático utilizado como herramienta de trabajo. Esta etapa ha abarcado un período de seis meses. También se incluye en este apartado el período de amortización del software utilizado.

2.1. METODOLOGÍA APLICADA

A continuación, en la tabla 2.1 se muestra una relación de los costes anuales y tasas horarias para presupuestar la partida de recursos humanos, considerando 220 días laborables al año y 8 horas de trabajo al día, a excepción del caso del becario, con 4 horas de trabajo diarias:

Categoría	Coste anual [EUR]	Tasa horaria [EUR/h]
Becario Ingeniero Aeronáutico	4400	5
Doctorando Ingeniero Aeronáutico	11000	6.67
Doctor Ingeniero Aeronáutico	24481.42	14.84

2.2. PRESUPUESTO DE RECURSOS HUMANOS

Una vez obtenidos los datos de tasa horaria recogidas en la tabla 2.1, estimando las horas de dedicación de cada uno de los integrantes del trabajo se puede obtener la cuantía del coste de cada uno de ellos, ver tabla 2.2:

Categoría	Dedicación [h]	Tasa horaria [EUR/h]	Coste [EUR]
Becario Ingeniero Aeronáutico	600	5	3000
Doctorando Ingeniero Aeronáutico	300	6.67	2001
Doctor Ingeniero Aeronáutico	120	14.84	1780.8
		TOTAL	6781.8

El coste en recursos humanos se estima por tanto en SEIS MIL SETECIENTOS OCHENTA Y UN EUROS Y OCHENTA CÉNTIMOS.

Capítulo 3

Coste referidos a los equipos

A continuación se contabilizará la amortización del uso de equipos informáticos y el coste de las licencias de software utilizadas.

3.1. METODOLOGÍA APLICADA

Durante el desarrollo del proyecto se han empleado una estación de trabajo y un ordenador portátil particular, así como las licencias del software MATLAB, Microsoft Office, ANSYS y CONVERGE. Estas licencias son anuales y se han obtenido de otros trabajos similares, sin embargo la licencia de OpenFOAM es libre.

El coste de amortización por año del equipo informático se calcula como la ecuación 3.1.

$$a = \frac{V_C - V_R}{n} \tag{3.1}$$

Donde:

- V_C es el valor de compra del equipo. Para el caso del portátil particular es de 670 EUR, y para la estación de trabajo 5000 EUR.
- V_R es el valor residual del equipo. Para el caso de un portátil y una estación de trabajo se estima en un 20 % del valor de compra.
- n es el periodo de amortización. Para el caso de un portátil particular se ha estimado en 5 años, y para la estación de trabajo 4.

Para calcular el coste de las horas de cálculo se estima el precio por núcleo y hora de 0.01 EUR. Las simulaciones se han realizado en aproximadamente 5 meses, utilizando Altamira, 2 cuentas de RIGEL con 64 núcleos cada una y la estación de trabajo con 32 núcleos. En la tabla 3.1 se recoge el desglose de núcleos y horas.

$$Costesimulaciones = 0.01[EUR/coreh] \ * \ 826000[coreh] = 8260[EUR] \tag{3.2}$$

Concepto	Número de horas [h]	Núcleos [cores]	Coste computacional [core h]
Altamira	-	-	250000
RIGEL	7200	64	460800
Estación	3600	32	115200
		TOTAL	826000

Por otra parte se incluye el coste de electricidad de los equipos en este apartado por la brevedad del mismo. Se ha calculado teniendo en cuenta la aproximación de horas de dedicación en la tabla 2.2 y que el precio de la electricidad ronda los 0,15 EUR/kWh.

$$Consumo = 0.09[kW] \ * \ 1020[h] = 91.8[kWh] \tag{3.3}$$

$$Consumo = 91.8[kWh] \ * \ 0.15[EUR/kWh] = 13.77[EUR] \tag{3.4}$$

3.2. PRESUPUESTO DE LOS EQUIPOS

En la tabla 3.2 se recogen todos y cada uno de los costes de licencia y equipos necesarios en el desarrollo del presente trabajo. Los precios anuales se han multiplicado por 0.67 debido a que el TFM ha durado 8 meses.

Concepto	Coste [EUR]
Estación de trabajo	670
Portátil particular	71.824
Coste simulaciones	8260
Coste electricidad	13.77
Microsoft Office	77.05
Licencia ANSYS	1340
Licencia MATLAB	1523
Licencia OpenFOAM	0
Licencia CONVERGE	1340
TOTAL	13295.644

El coste en equipos se estima por tanto en TRECE MIL DOSCIENTOS NOVENTA Y CINCO EUROS Y SESENTA Y CUATRO CÉNTIMOS.

Capítulo 4

Costes generales

En este apartado se tienen en cuenta los costes relacionados con los imprevistos y medios auxiliares. Se estiman como el 5 % de la suma de los costes de personal y de equipos, según lo recogido en la tabla 4:

Concepto	Coste [EUR]
Costes de personal	6781.8
Costes de equipos	13295.644
Subtotal	20077.444
Costes generales (5 %)	1003.872

El coste general se estima en MIL TRES EUROS Y OCHENTA Y SIETE CÉNTIMOS.

Capítulo 5

Presupuesto total

El presupuesto total queda definido como la suma de los costes de recursos humanos, costes de equipo y costes generales, aplicándoles el IVA correspondiente. Pueden verse en la tabla 5:

Concepto	Coste [EUR]
Costes de personal	6781.8
Costes de equipos	13295.644
Costes generales	1003.872
Subtotal	21081.316
IVA (21 %)	4427.076
TOTAL	25508.392

Por lo tanto, puede decirse que el presupuesto total estimado para la realización del Trabajo Final de Máster es de VEINTICINCO MIL QUINIENTOS OCHO EUROS Y TRENTA Y NUEVE CÉNTIMOS.