Universidad Zaragoza
1542

Proyecto Fin de Carrera

MODELADO DEL FLUJO DE AIRE A TRAVÉS DE GRUPOS DE ÁRBOLES

Autor

José Miguel Maldonado Jiménez

Director

Mathias Cehlin

Ponente

Guillermo Hauke

Departamento de Mecánica de Fluidos

Escuela de Ingeniería y Arquitectura

2012

Resumen

Los emplazamientos más apropiados para instalar un parque eólico, están ocupados actualmente o por motivos medioambientales no se puede hacer uso de los mismos. Por este motivo, es necesario comprobar la posibilidad de si otras localizaciones disponibles, donde el terreno es más complejo, son viables. Para este fin, se utilizan herramientas de Fluido dinámica Computacional o CFD (Computational Fluid Dynamics). Para la realización de este trabajo se ha seleccionado el software llamado OpenFOAM, un software libre que, a diferencia de otros comerciales, su código (programado en C++) es suficientemente flexible y el programa es gratuito.

El proyecto básicamente consiste en desarrollar una aplicación que permita simular el efecto de los árboles y bosques sobre las corrientes de aire. Hasta el momento esta tarea se venía haciendo considerando los arboles como parte de la rugosidad del terreno, pero es recomendable añadir la distorsión producida por estos de manera separada a la rugosidad.

Se puede separar el proyecto en dos partes, coincidiendo cada una con una aplicación a desarrollar:

• El primer objetivo es desarrollar una aplicación que cargue los parámetros que definen el tipo de árboles y el tamaño de éstos. Esta herramienta de pre-procesado situará los diferentes grupos de árboles a lo largo de la malla del terreno en estudio.

• El segundo objetivo consiste en programar un “solver” el cual resuelve una configuración teniendo en cuenta el efecto producido por el follaje sobre el flujo de aire.

Gracias a estas aplicaciones, se podrá estudiar de manera más detallada si un terreno es adecuado para instalar aerogeneradores. La finalidad de este tipo de simulaciones es prever la velocidad con la que el viento llega al molino, para así estimar la producción energética anual del mismo.

Índice.

Lista de Figuras

Lista de Tablas

1. Introducción.

1.1. Energía eólica.

La energía eólica es un recurso energético abundante, renovable y limpio, el cual obtiene electricidad gracias a la fuerza del viento. Una vez que los aerogeneradores han sido instalados, no requieren de combustibles para funcionar y tampoco emiten contaminantes perjudiciales para el medio ambiente; por ello la sustitución de plantas energéticas que utilizan combustibles fósiles por granjas eólicas, ayuda a reducir las emisiones de gases de efecto invernadero. Este tipo de energía es de las más famosas entre las energías renovables, y la sociedad la relaciona con la energía verde por excelencia. Cuando una compañía desea llevar a cabo un nuevo proyecto para instalar una planta eólica, al ser respetuosa con el medio ambiente, normalmente se encuentran menos trabas en el desarrollo del proyecto.

Sin embargo, este método de producción energético no es totalmente fiable, es decir no produce electricidad a voluntad del consumidor sino cuando las condiciones climáticas lo permiten. Por este motivo, una granja eólica debe ser respaldada por otro tipo de planta energética, la cual sea capaz de suministrar la demanda eléctrica en caso de que la primera no sea capaz. En ocasiones las palas de los aerogeneradores impactan con las aves que sobrevuelan la zona donde están instalados; se ha dado el caso de plantas eólicas que han sido bloqueadas debido a que fueron instaladas en zonas donde especies de aves protegidas habitan. Por lo general los aerogeneradores se colocan en puntos altos sobre el perfil geográfico, normalmente son zonas de gran belleza natural, por lo que los molinos de viento rompen esa hegemonía de la naturaleza. Se ha dado la situación de grupos, que desean la protección del medio ambiente, se oponen al desarrollo de planta eólicas por los motivos anteriormente mencionados. Un ejemplo de esta situación sería el grupo ecologista “Agnaden” el cual ha denunciado la instalación de un parque eólico en “El Padul (Granada)” por estar situado este en una zona donde el “Alondra Ricotí” (un tipo de ave considerada en peligro) habita **[16]**.

Para valorar la relevancia de energía eólica se debe conocer la electricidad que es capaz de producir. Por ejemplo actualmente Suecia tan solo cubre aproximadamente un 2% de su demanda eléctrica en el año 2009, sin embargo en España con los datos ofrecidos por la International Energy Agency (IEA) **[1]** también del año 2009, se puede comprobar como el 13% de la electricidad consumida es suministrada por aerogeneradores. El mismo caso se da en Dinamarca donde la energía eólica abasteció el 18% de la electricidad del país en ese mismo año. Estos dos últimos ejemplos indican porqué merece la pena invertir en el desarrollo de esta tecnología.

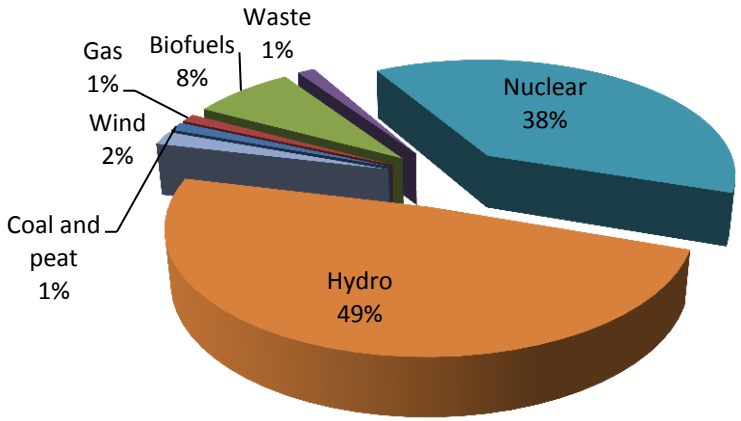


Fig 1.1 Producción eléctrica en Suecia en 2009.

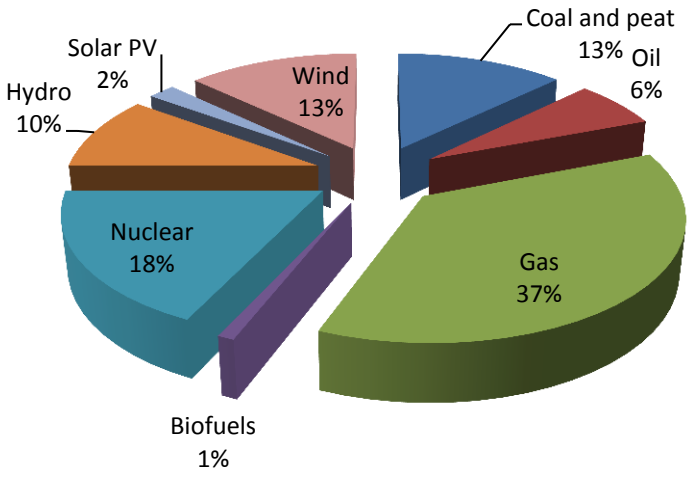


Fig 1.2 Producción eléctrica en España en 2009.

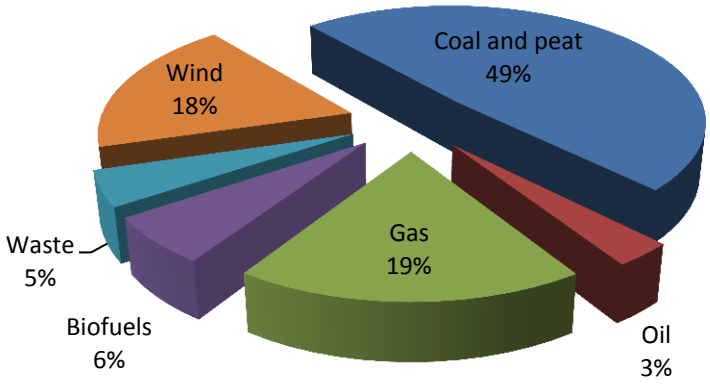


Fig 1.3 Producción eléctrica en Dinamarca en 2009.

1.1. Motivación.

La electricidad producida por los aerogeneradores está limitada por la ley de Betz (ver APÉNDICE I) y depende de la velocidad del viento, del tamaño del molino, de la eficiencia del generador eléctrico, pero principalmente de la velocidad del viento. Por lo que se busca situar el parque eólico donde la velocidad del viento sea lo más alta posible sin superar la velocidad de corte (velocidad máxima a la que un aerogenerador deja de aumentar la potencia de generación) normalmente 24 m/s. Si los vientos son extremadamente fuertes se deberá bloquear el molino para no poner en riesgo su infraestructura. Y más importante se buscan zonas donde el viento sea constante a lo largo del tiempo.

La mejor situación para instalar una granja eólica es en el mar, porque el viento sopla con fuerza y de manera constante. Este tipo de parques, llamados offshore, incurren en mayores gastos de mantenimiento e instalación por la corrosión que produce el agua de mar. Dinamarca fue el país pionero en esta práctica en 1991, desde entonces 39 parques eólicos offshore han sido instalados in Europa repartidos entre Dinamarca, Bélgica, Finlandia, Alemania, Irlanda, Holanda, Noruega, Reino Unido y Suecia.

En España a día de hoy no existen parque eólicos offshore y las zonas más apropiadas para instalar aerogeneradores en el interior han sido ocupadas o no están disponible por motivos ambientales. Por este motivo se deben buscar áreas que puedan acoger nuevas granjas eólicas, para ello se debe estudiar el terreno en profundidad antes de involucrarse en un

proyecto de gran envergadura. Aquí entran en juego los programas de Computación Fluido Dinámica (CFD), los cuales permiten simular el comportamiento del viento en la zona de estudio.

En este proyecto se plantea la instalación de aerogeneradores en los alrededores de bosques o zonas arboladas, los cuales no van a ser talados solo por el buen desarrollo del viento porque no sería una medida respetuosa con el medio ambiente. Cuando las corrientes de aire atraviesan el follaje de los árboles, su velocidad disminuye y su turbulencia aumenta. Varios estudios experimentales se han realizado con el fin de obtener el comportamiento del viento a lo largo de zonas arboladas.

1.2. Objetivos.

El objetivo principal es desarrollar una herramienta de CFD la cual permita determinar la mejor situación para instalar una granja eólica sobre terreno complejo. Terreno complejo quiere decir que la rugosidad del suelo, las zonas boscosas y los otros aerogeneradores deben ser tenidos en cuenta en la simulación del flujo de aire. Basándonos en otro proyecto anterior el cual incorporaba la rugosidad a la simulación, en este proyecto ampliaremos el programa añadiendo el efecto de las zonas arboladas. Para ello serán necesarias dos aplicaciones distintas.

Fase 1: Pre proceso. Las dimensiones y características del bosque deben ser añadidas al software CFD antes de realizar la simulación. Para ello esta aplicación leerá estos datos desde un archivo en formato ".map" y los incorporará al caso de estudio.

Fase 2: Los distintos "solvers" que puede utilizar el software CFD, se basan en los distintos modelos turbulentos que incorpora el software CFD. Una vez las zonas arboladas forman parte del caso en estudio, se debe modificar el "solver" que se desea utilizar para añadir las ecuaciones generadas por la resistencia al viento del follaje, a las ecuaciones de Navier-Stokes.

1.3. Método.

A lo largo de este proyecto se va a utilizar OpenFOAM como software CFD, esta herramienta trabaja sobre el sistema operativo Linux.

Se ha facilitado un caso de estudio real, la colina Bolund situada en la costa norte de Risø DTU, para poder realizar las simulaciones y testear el funcionamiento del código desarrollado. Esta malla está realizada por FLUENT pero mediante la aplicación *FluentToFoam,* la malla puede ser utilizada por OpenFOAM.

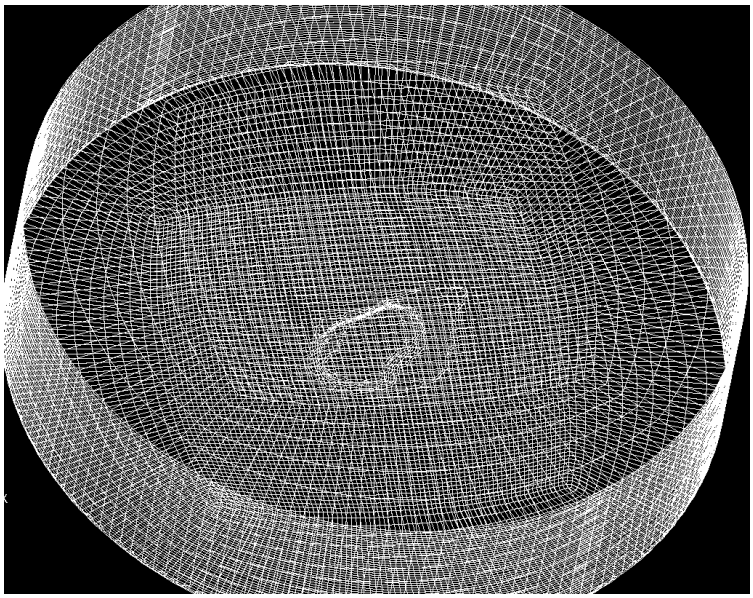


Fig 1.4 La colina Bolund representada en OpenFOAM.

Una vez se tiene la malla para poder crear un caso de estudio, se necesita desarrollar un archivo mapa el cual incluya las distintas zonas arboladas que se reparten por el área de estudio.

Por último se probará el buen funcionamiento de las aplicaciones. La primera será llamada "*leafIndexToFoam*", tras ejecutarla se deberá comprobar si los bosques descritos por el archivo mapa, corresponden con los datos cargados a la simulación. La segunda aplicación llamada "*treeFoam*" se comprobará comparando el comportamiento del viento en varias simulaciones:

Simulación 1: Se utiliza un solver estándar creado por OpenFOAM ("*simpleFoam*") de esta manera se obtienen datos validos sobre el flujo de aire en el caso de estudio.

Simulación 2: La simulación se realiza con el nuevo solver ("*treeFoam*"), pero sin cargar las zonas arboladas previamente. Así se puede comparar la simulación 2 frente a la simulación 1, para comprobar que el nuevo solver al menos funciona correctamente.

Simulación 3: Se realizan varias pruebas cargando previamente los bosques con la aplicación "*leafIndexToFoam*", de esta manera se comprueba si el solver "*treeFoam*" tiene en cuenta el efecto de los árboles sobre el flujo de aire. Se varían las características del bosque para comprobar que el viento se ve distorsionado de manera correcta.

2. Fundamentos Teóricos.

A la hora de trabajar con herramientas de fluido dinámica computacional, se debe tener en cuenta que existen varios modelos numéricos que pueden otorgar una solución a un problema. La diferencia de estos modelos determinará su precisión, es decir su similitud con la situación real y también su complejidad a la hora de resolver el problema. Asimismo, las condiciones del flujo escogidas o asumidas también juegan un papel importante.

Por ello se debe aceptar o elegir varias hipótesis:

- Flujo incompresible. La densidad del fluido permanece contante, por lo que las variaciones de la densidad debidas a la altura y a la temperatura no se tienen en cuenta.
- Comportamiento NEUTRAL del flujo en la "Capa límite atmosférica". Se considera neutral debido a que la densidad del fluido es constante (ver APÉNDICE VII apartado "2.1 Atmospheric Boundary Layer").
- Efecto de fuerzas externas como la fuerza de Coriolis, o la rugosidad del terreno, se desprecian. Estas fuerzas externas pueden dificultar el análisis del efecto del follaje sobre la corriente de aire, por lo que es mejor añadirlas tras comprobar el buen funcionamiento del nuevo solver.
- Modelo turbulento a utilizar. Se elige el modelo turbulento *k-epsilon*, por ser el mas común en las aplicaciones ingenieriles dentro de los métodos de resolución RANS (ver Apartado 2.3).

2.1. Turbulencia.

2.1.1. Efecto de la turbulencia en las Ecuaciones del Transporte promediadas.

Primero de todo, se puede encontrar una breve definición de la turbulencia en el apéndice VII apartado 2.1.1 ("Turbulence" "Definition").

Calcular el comportamiento de los vórtices en una corriente turbulenta, no es una tarea sencilla; además no existe una capacidad computacional tal para poder resolver las ecuaciones

de Navier-Stokes para flujos turbulentos, de manera completamente detallada, con dependencia del tiempo y con altos números de Reynolds.

Normalmente el promedio de las propiedades del flujo, como la velocidad media o las tensiones medias, suministran suficiente información para los usuarios de CFD. No es necesario predecir el comportamiento de cada vórtice.

Las propiedades como la velocidad (u) o la presión (p), se pueden establecer como la suma de una componente media estacionaria (U o P) y una componente fluctuante que varía con el tiempo (u´ o p´):

$$u = U + u' \qquad \textbf{Eqn 2.1}$$

$$p = P + p' \qquad \textbf{Eqn 2.2}$$

La media de las componentes fluctuantes son cero por definición.

$$\overline{u'} = \frac{1}{\Delta t} \cdot \int_0^{\Delta t} u'(t)dt = 0 \qquad \textbf{Eqn 2.3}$$

De la media cuadrática "root-mean-square" (rms) de las fluctuaciones, se puede obtener la componente de la velocidad fluctuante.

$$u_{rms} = \sqrt{\overline{(u')^2}} = \left[\frac{1}{\Delta t} \cdot \int_0^{\Delta t} (u')^2 dt = 0\right]^{1/2} \qquad \textbf{Eqn 2.4}$$

La media cuadrática de la velocidad se puede medir con una sonda (por ejemplo un "*hot-wire*" anemómetro) y un circuito eléctrico. Con u_{rms} la velocidad fluctuante (u´) se puede extraer.

Introduciendo las componentes fluctuantes en las ecuaciones de Navier-Stokes, se puede estudiar el efecto de la turbulencia sobre el flujo (ver APÉNDICE II.1).

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0 \qquad \textbf{Eqn 2.5}$$

$$\frac{\partial (\rho u)}{\partial t} + u\big(\nabla \cdot (\rho u)\big) = -\nabla \cdot p + \mu \nabla^2 u + S_M \qquad \textbf{Eqn 2.6}$$

Donde ρ es la densidad del fluid y S_M representa las fuerzas externas.

Utilizando coordenadas cartesianas:

$\mathrm{u} = (u, v, w)\ \mathrm{U} = (U, V, W)\ \ \mathrm{u}' = (u', v', w')$

$u = U + u'$

$v = V + v'$

$w = W + w'$

$\mathrm{p} = \mathrm{P} + \mathrm{p}'$

Las ecuaciones de Navier-Stokes promediadas son:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathrm{U}) = 0 \qquad \textbf{Eqn 2.7}$$

$$\frac{\partial(\rho U)}{\partial t} + \nabla \cdot (\rho U.\mathrm{U}) = -\frac{\partial \mathrm{P}}{\partial x} + \mu \nabla^2 U - \left[\frac{\partial \rho \overline{u'^2}}{\partial x} + \frac{\partial \rho \overline{u'v'}}{\partial y} + \frac{\partial \rho \overline{u'w'}}{\partial z}\right] + S_M \quad \textbf{Eqn 2.8.a}$$

$$\frac{\partial(\rho V)}{\partial t} + \nabla \cdot (\rho V.\mathrm{U}) = -\frac{\partial \mathrm{P}}{\partial y} + \mu \nabla^2 V - \left[\frac{\partial \rho \overline{v'u'}}{\partial x} + \frac{\partial \rho \overline{v'^2}}{\partial y} + \frac{\partial \rho \overline{v'w'}}{\partial z}\right] + S_M \quad \textbf{Eqn 2.8.b}$$

$$\frac{\partial(\rho W)}{\partial t} + \nabla \cdot (\rho W.\mathrm{U}) = -\frac{\partial \mathrm{P}}{\partial z} + \mu \nabla^2 W - \left[\frac{\partial \rho \overline{w'u'}}{\partial x} + \frac{\partial \rho \overline{w'v'}}{\partial y} + \frac{\partial \rho \overline{w'^2}}{\partial z}\right] + S_M \quad \textbf{Eqn 2.8.c}$$

Estas son las llamadas ecuaciones de Reynolds. Aparecen seis términos nuevos después del proceso de promediado, estos términos son conocidos como “las tensiones de Reynolds”. Tres de ellos son tensiones normales y las otras tres son tensiones cortantes. Se necesita un modelo turbulento para calcular estas nuevas tensiones de manera suficientemente precisa, pero sin olvidar que una solución extremadamente precisa requiere una capacidad computacional no disponible.

2.2. Modelado Numérico.

2.2.1. Modelos Turbulentos

Resolver las fluctuaciones turbulentas de una manera muy detallada, es innecesario. Por eso, normalmente se buscan los efectos turbulentos que distorsionan el flujo medio. Un código CFD que sea útil, el modelo turbulento en el que se basa debe ser: suficientemente preciso, simple, informáticamente económico y sobre todo debe ser aplicable a un amplio número de casos.

Existen varios métodos para estudiar el comportamiento de flujos turbulentos, a continuación se mencionan algunos de ellos.

- **Direct Numerical Simulations (DNS):** Este modelo resuelve directamente las ecuaciones de Navier-Stokes, sin promedios ni aproximaciones. Es la manera mas precisa de resolver el comportamiento de un flujo turbulento, sin embargo conlleva un coste computacional tal que no se puede llevar a la práctica.

- **Large Eddy Simulation (LES):** En los flujos turbulentos se pueden encontrar varias escalas de tiempo y espacio. Las escalas mas grandes tienen una capacidad de transporte mayor y albergan casi toda la energía. Este método se basa en resolver las grandes escalas del flujo y modelar las menores. Exige gran capacidad computacional y tiempo de cáculo.

- **Reynolds-Average Simulation (RAS) o Reynolds-Averaged Navier-Stokes (RANS):** Es el modelo más utilizado para aplicaciones ingenieriles. Este modelo se basa en el promedio de las ecuaciones de Navier-Stokes, que da lugar a las ecuaciones de Reynolds.

- **Detached Eddy Simulation (DES):** Este método es un híbrido entre RANS y LES. Se utiliza principalmente el método RANS pero para algunas regiones del flujo se usa la metodología LES.

Como se ha dicho, el método de resolución RANS es el más común a la hora de resolver flujos turbulentos en el mundo de la ingeniería. Antes de que la turbulencia apareciera en el flujo existían cuatro incógnitas (*u,v,w* y *p*), sin embargo si se tiene en cuenta la turbulencia a través del promedio (time-averaging) de Reynolds, aparecen seis incógnitas nuevas. Para poder resolver el nuevo sistema de ecuaciones, formado por las ecuaciones de Reynolds (Eqn 2.7, Eqn 2.8.a, Eqn 2.8.b, Eqn 2.8.c), se requiere un modelo turbulento para cerrar el sistema.

Dentro del método de resolución RANS existen varios modelos turbulentos, a continuación se mencionan aquellos que provee OpenFOAM para flujos incompresibles (Tabla 2.1).

Laminar	Dummy turbulence model for laminar flow
kEpsilon	Standard k-ε model
kOmegaSST	k-ε-SST model
RNGkEpsilon	RNG k-ε model
LaunderSharmaKE	Launder-Sharma low-Re k-ε model
LRR	Launder-Reece-Rodi RSTM
LaunderGibsonRSTM	Launder-Gibson RSTM
realizableKE	Realizable k-ε model
SpalartAllmaras	Spalart-Allmaras 1-eqn mixing-length model

Tabla 2.1 Modelos RANS suministrados por OpenFOAM

De todos estos, dos de ellos se utilizan comúnmente en aplicaciones ingenieriles, el modelo *k-epsilon standard* y el modelo *k-epsilon RNG*. En este proyecto se utiliza el primero de ellos por lo que posteriormente, se desarrolla con mayor detalle.

2.2.2. Aproximación de Boussinesq

Se puede suponer que las tensiones viscosas y las tensiones de Reynolds, están conectadas entre ellas y que ambas tienen casi el mismo efecto sobre el flujo medio. Por lo que las tensiones debidas a la turbulencia se pueden escribir como:

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \qquad \textbf{Eqn 2.9}$$

De acuerdo con Boussinesq, las tensiones de Reynolds se pueden relacionar con la velocidad media de deformación, de tal manera que las tensiones turbulentas aumentan con el la velocidad media de deformación.

$$\tau_{ij} = -\rho\overline{u_i' u_j'} = \mu_t \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) \qquad \textbf{Eqn 2.10}$$

Donde μ_t es la viscosidad turbulenta (dimensiones Pa.s), se utiliza para modelar las fuerzas turbulentas como fuerzas viscosas. Si se obtiene μ_t, el sistema se puede resolver. La viscosidad total (μ_{Total}) es la suma de la viscosidad del fluido (μ) y de la viscosidad turbulenta (μ_t) **[2 y 3]**.

2.2.3. Modelo turbulento k-epsilon.

El modelo turbulento *k-epsilon* es un modelo de dos ecuaciones. Se introducen dos nuevos términos con sus respectivas ecuaciones de transporte, las cuales representan las propiedades turbulentas que aparecen en el flujo medio **[2 y 3]**.

Estas dos nuevas variables son la energía cinética (*k*) y la disipación de energía cinética turbulenta (ε). La última de ellas determina la escala de la turbulencia mientras que la primera representa la energía en la turbulencia.

La energía cinética turbulenta por unidad de masa se puede escribir como:

$$k = \frac{1}{2}\left(\overline{u'^2} + \overline{v'^2} + \overline{w'^2}\right) \qquad \textbf{Eqn 2.11}$$

Según el modelo de k-epsilon estándar (Launder y Spalding 1974), estas dos variables *k* y ε, se utilizan para definir las escalas de longitud (*l*) y velocidad (v) características:

$$\vartheta = k^{1/2} \qquad \textbf{Eqn 2.12}$$

$$l = \frac{k^{3/2}}{\varepsilon} \qquad \textbf{Eqn 2.13}$$

La escala de velocidad y de longitud, definen la viscosidad turbulenta:

$$\mu_t = C_\mu \rho\, l\, \vartheta = C_\mu \rho \frac{k^2}{\varepsilon} \qquad \textbf{Eqn 2.14}$$

Donde C_μ es una constante adimensional.

Finalmente las ecuaciones del transporte que introduce el modelo k-epsilon son:

Ecuación de la energía cinética:

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k U_i)}{\partial x_j} = \tau_{ij}\frac{\partial U_i}{\partial x_j} - \rho\varepsilon + \frac{\partial}{\partial x_i}\left[\left(\mu + {}^{\mu_t}/_{\sigma_t}\right)\frac{\partial k}{\partial x_i}\right] \qquad \textbf{Eqn 2.15}$$

Disipación de la energía cinética:

$$\frac{\partial(\rho\varepsilon)}{\partial t} + \frac{\partial(\rho\varepsilon U_i)}{\partial x_j} = C_{\varepsilon 1}\frac{\varepsilon}{k}\tau_{ij}\frac{\partial U_i}{\partial x_j} - C_{\varepsilon 2}\rho\frac{\varepsilon^2}{k} + \frac{\partial}{\partial x_i}\left[\left(\mu + {}^{\mu_t}/_{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_i}\right] \qquad \textbf{Eqn 2.16}$$

Las constantes que aparecen en las ecuaciones anteriores (Eqn 2.15 and 2.16) toman el siguiente valor:

C_μ	$C_{\varepsilon 1}$	$C_{\varepsilon 2}$	σ_k	σ_ε
0,09	1,44	1,92	1,0	1,3

Tabla 2.2 Coeficientes en las ecuaciones del transporte k-epsilon.

2.3. Modelado de la fuerza de arrastre debida al follaje.

2.3.1. Parámetros característicos.

Coeficiente de arrastre (C_D).

El comportamiento del flujo de aire a través de un bosque y la arquitectura del follaje de los árboles, están conectadas con el coeficiente de arrastre (C_D). Para poder definir el mapa de árboles que se deben insertar en el domino es necesario calcular este parámetro. El problema radica en que no existe una tabla o lista donde se clasifique el coeficiente C_D para cada tipo de árbol.

Hay varios estudios que intentan obtener ese parámetro mediante métodos experimentales, pero no hay ninguna ecuación que establezca una manera fiable para calcularlo (ver APÉNDICE II.2).

Densidad de hojas por área (α).

Este parámetro significa la superficie de follaje (hojas, ramas o tronco) por metro cúbico de bosque. Al contrario que C_D se puede medir y no presenta problemas para su obtención.

2.4.2 Implementación de la fuerza de arrastre.

A pesar de la dificultad para calcular el C_D, la fuerza de arrastre que produce el follaje sobre el flujo de aire se puede añadir a la ecuación del momento como una fuerza externa de arrastre (Dalpe and Masson 2008 **[7; 8]**):

$$S_u = -\rho C_D \alpha |u| u_i \qquad \textbf{Eqn 2.18}$$

El modelo turbulento elegido para realizar las simulaciones durante este estudio, es el modelo $k - \varepsilon$, el cual debe ser modificado para insertar la turbulencia producida por el follaje en la corriente de aire. Aparecen dos nuevos términos en las ecuaciones del transporte (S_K y S_ε).

$$S_k = \rho C_D \alpha \left(\beta_p |u|^3 - \beta_d k |u|\right) \qquad \textbf{Eqn 2.19}$$

$$S_\varepsilon = \rho C_D \alpha \frac{\varepsilon}{k} \left(C_{\varepsilon 4} \beta_p |u|^3 - C_{\varepsilon 5} \beta_d k |u|\right) \qquad \textbf{Eqn 2.20}$$

3. OPENFOAM

3.1. Introducción.

OpenFOAM es un software de CFD el cual es gratis y cuyo código esta abierto, fue desarrollado a finales de los años ochenta por la Imperial College de Londres; sin embargo no sería un software libre hasta el 2004. Al ser un software libre, se debe utilizar un sistema operativo también libre, como Linux, para poder utilizarlo.

Como es un programa abierto, los usuarios pueden modificar, ampliar, modificar o simplemente personalizar las aplicaciones actuales de OpenFOAM; por este motivo es una herramienta CFD común entre la comunidad ingenieril. Se pueden resolver una amplia gama de problemas fluido dinámicos, flujos complejos con turbulencia, transferencia de calor e incluso reacciones químicas. También se puede utilizar para analizar la dinámica del sólido y sistemas electromagnéticos.

Además de los "solvers", OpenFOAM facilita herramientas de pre-procesado como por ejemplo, creadores de mallas (e.g. *BlockMesh* o *snappyHexMesh*) o conversores para adaptar mallas de otros softwares (e.g. FLUENT) a OpenFOAM. También incluye herramientas de pos-proceso, como *paraView*, que permiten trabajar con la solución del solver de manera gráfica.

Existe una gran comunidad en la red, la cual comparte sus aplicaciones y ofrece ayuda entre los usuarios de OpenFOAM [14]. Las capacidades de los "solvers" estándares, las herramientas de pre y post procesado, así como las librerías de las que dispone OpenFOAM, están listadas en el apéndice VII apartado 3.1 "OPENFOAM Introduction".

Se puede decir que OpenFOAM es solo una librería, la cual permite crear archivos ejecutables (aplicaciones) para resolver problemas fluido dinámicos. Existen dos tipos de aplicaciones:

- "Solvers", los cuales resuelven los sistemas de ecuaciones que se aplican en cada caso.
- Utilidades, que se utilizan para tratar los datos antes y después de ser resuelto un problema.

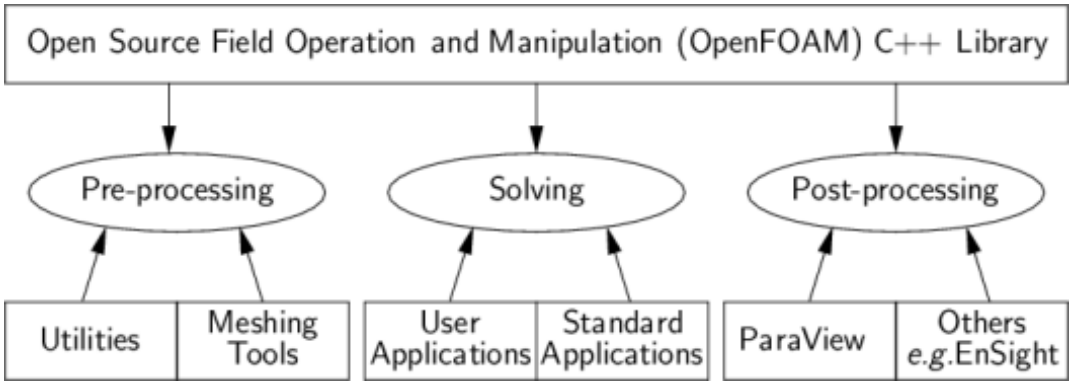


Fig 3.1 Estructura de OpenFOAM [14]

OpenFOAM utiliza C++ como lenguaje de programación en sus archivos. Este lenguaje es uno de los más comunes hoy en día. No es necesario ser un experto programador para modificar los códigos que por defecto se suministran con OpenFOAM, pero se sugiere familiarizarse con este lenguaje previamente. En la red se pueden encontrar algunos tutoriales, los cuales se especializan en los conocimientos de C++ necesarios para poder trabajar con OpenFOAM [20]. Además se puede descargar un manual para este fin directamente de la web de OpenFOAM **[15]** (Programmer´s Guide).

3.2. Aplicaciones en OpenFOAM.

3.2.1. Código principal de la aplicación, archivo ".C"

El archivo principal, o archivo fuente (*newApp.C*) con extensión ".C" es el primer código que la aplicación lee cuando se ejecuta, contiene la definición de cada clase que en el código. Una clase para la creación o construcción de objetos, las clases de las funciones y almacén de datos.

Cada clase y cada operación utilizada por el código principal debe existir, por eso se deben declarar todas y cada una de las clases utilizadas, normalmente esto se hace en los archivos cabecera (headers files ".H"). Los nombres de las clases y sus funciones se declaran en estos archivos, los cuales son incluidos al comienzo del archivo ".C", mediante la línea de comando:

#include "class.H"

Algunas aplicaciones de OpenFOAM requieren un archivo llamado *createFields.H*, el cual crea los campos necesarios y lee los datos de entrada.

Fig 3.2 Estructura de una aplicación [14].

Los sistemas operativos UNIX/Linux suministran un compilador llamado "Make", sin embargo el propio OpenFOAM ofrece otro más versátil y simple, "wmake". Para que OpenFOAM reconozca la aplicación y se pueda ejecutar, el código completo debe ser compilado previamente. Se necesita de un directorio llamado "*Make"*, donde se encuentran dos archivos: *"options"* y *"files"*.

El archivo *"options"* contiene todos los nombres de las librerías que se utilizan y las rutas donde se hallan. Se utiliza la siguiente codificación:

```
EXE_INC = \
        -I<directoryPath1>\
        -I<directoryPath2>\
        …
        -I<directoryPathN>
EXE_LIBS = \
        -L<libraryPath1>\
        -L<libraryPath2>\
        …
        -L<libraryPathN>\
```

-I<library1>\

-I<library2>\

…

-I<libraryN>

El compilador necesita una lista de todos los archivos fuente (.C) que se deben compilar. El archivo *"files"* contiene una lista con el nombre de los archivos fuente a compilar y la ruta donde se desea que el compilador genere los ejecutables. En esta lista, al menos el nombre del código principal *"newApp.C"* aparecerá, pero también lo harán otras fuentes creadas para la aplicación y que no están en ninguna clase o librería; por ejemplo en este proyecto, la aplicación "treeFoam" requiere de archivos fuente que no están en las librerías estándar (*kEpsilon.C* y *RASModelhig.C* los cuales se encuentran en el directorio "*src"* del usuario) y estos deben ser compilados. Se utiliza la siguiente codificación:

newApp.C

EXE = $(FOAM_USER_APPBIN)/newApp

Las nuevas aplicaciones desarrolladas por el usuario se deben almacenar en la ruta $FOAM_USER_APPBIN; y las aplicaciones estándar suministradas con OpenFOAM se hallan en la ruta $FOAM_APPBIN.

El compilador "wmake" crea un archivo que contiene una lista de dependencias (extensión ".dep") en este caso "*newApp.dep"*. Si se requiere recompilar una clase existente para incorporar algunos cambios, se deben eliminar las antiguas dependencias previamente, esto se hace introduciendo "wclean" en la terminal.

3.2.2. Codificación de las ecuaciones en C++.

Una de las principales ventajas de usar OpenFOAM es que la forma en que se escriben las ecuaciones codificadas en C++, no dista mucho de la forma diferencial. Por ejemplo, la ecuación del momento para un flujo incompresible laminar es:

$$\frac{\partial \rho U}{\partial t} + \nabla \cdot \phi U - \nabla \cdot \mu \nabla U = -\nabla p \qquad \textbf{Eqn 3.1}$$

Esta ecuación codificada en C++ es:

```
Solve
(
    fvm::ddt(rho, U)
  + fvm::div(phi, U)
  - fvm::lapalcian(mu, U)
    ==
  - fvc::grad(p)
);
```

Para desarrollar nuevas aplicaciones, se necesitan algunos conocimientos de programación en C++, sin embargo se requieren también nociones sobre los modelos de ecuaciones, los métodos y algoritmos que se utilizan para resolver problemas fluido dinámicos. Los últimos son incluso más importantes para crear nuevos “solvers”.

3.3. Casos en OpenFOAM.

Cada caso que se desea estudiar en OpenFOAM, requiere unos archivos mínimos para poder aplicarle alguna aplicación; en la imagen 3.3 se muestran estos archivos. Hay varios casos que vienen con el software, los cuales pueden servir de ejemplo; estos se hallan en el directorio $FOAM_TUTORIALS. Estos diversos casos se dividen en función del “solver” que se usa para resolverlos. Cuando se desea crear un nuevo caso, los usuarios normalmente eligen un caso dado el cual sea similar a la situación de estudio, y se modifica para adaptarlo a las necesidades del usuario.

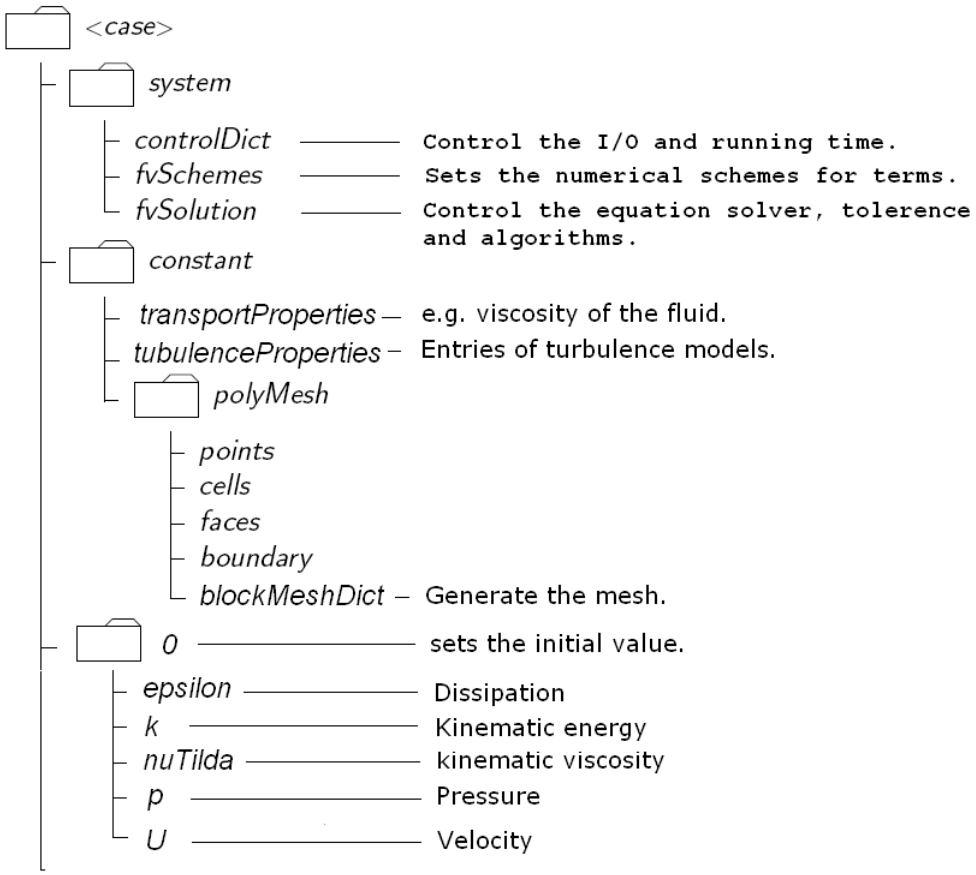


Fig 3.3 Estructura de un caso en OpenFOAM [14].

3.3.1. Fichero "System"

Como mínimo se hallan los tres archivos siguientes:

- *controlDict*: Se definen los parámetros de control de la simulación. Tiempo inicial y final ("Start/end") y el intervalo de tiempo ("time step"), también se definen los parámetros de salida (ver APÉNDICE IV.1).
- *fvSchemes*: En este archivo se definen los métodos numéricos a utilizar. El gradiente, la divergencia, el laplaciano, el método de interpolación de los valores, se definen este archivo (ver APÉNDICE IV.2).
- *fvSoluton*: Control alas ecuaciones del "solver", las tolerancias y los algoritmos (ver APÉNDICE IV.3).

3.3.2. Fichero "Constant"

En el fichero "Constant" se especifican las propiedades físicas del caso. Se encuentran dos archivos en su interior:

***transportProperties*:** Define las propiedades del fluido que va a ser utilizado, por ejemplo la viscosidad del mismo (ver APÉNDICE IV.4)

***turbulenceProperties*:** El modelo turbulento que se va a usar para resolver el caso, se define o escoge en este archivo.

Dentro del directorio "Constant" se encuentra el fichero ***polyMesh*.** En este fichero se define la malla que representa el dominio de estudio. Los siguientes archivos definen la malla:

points: Contiene una lista con todos los puntos que definen la figura de estudio. Cada punto se define por tres coordenadas. No se pueden definir dos o más puntos en la misma localización (ver APÉNDICE IV.6)

cells: Este archivo se utiliza solo si la malla se define manualmente.

faces: Alberga una lista de puntos que definen las caras de la malla. Estos puntos están definidos en esta lista por su etiqueta (posición que ocupan en la lista de puntos) en lugar de por sus coordenadas. Los puntos están ordenados de tal manera que se unen cada dos puntos vecinos. Cada cara tiene su propia etiqueta para poder referirse a ella, que corresponde a la posición que ocupa en la lista (ver APÉNDICE IV.7).

Se pueden definir dos tipos de caras.

- Caras internas.

  Estas caras son compartidas por dos celdas.

- Caras de contorno.

  Estas caras solo pertenecen a una celda, porque están situadas en el contorno del dominio.

boundary: El contorno se define con una lista de parcelas a las cuales se les asocian unas condiciones de contorno. Una parcela es simplemente una lista de caras externas o de contorno, las caras se declaran este archivo mediante su etiqueta (ver APÉNDICE IV.8), las caras internas no pueden formar parte del contorno.

owner: En este archivo asocia una celda a cada cara (ver APÉNDICE IV.9).

neighbour: Es una lista de las celdas vecinas por su etiqueta.

blockMeshDict: Este archivo aparece si la malla se define utilizando "blockMesh". Este archivo determina el procedimiento que la herramienta sigue para crear la malla (ver APÉNDICE V).

La malla se puede definir a mano, lo cual es una tarea costosa si se quiere estudiar una situación levemente compleja; utilizando la utilidad "blockMesh" (ver APÉNDICE V) o la aplicación "snappyHexMesh" (ver **[14]** apartado 5.4 "Mesh generation with the *snappyHexMesh* utility") también suministrada con OpenFOAM; o importando la malla desde otra herramienta de mallado como por ejemplo "GAMBIT".

3.3.3. Fichero "0"

Las variables que se utilizan en el caso de estudio se deben declarar en unos archivos, los cuales se encuentran en el interior del fichero "0". La velocidad, presión, energía cinética o las nuevas variables necesarias para insertar el efecto de los bosques sobre el flujo de aire como el coeficiente de arrastre (C_D) y la densidad de hojas por área (α). En estos archivos se fija el valor inicial para cada variable. El perfil de velocidades inicial se define en el archivo *"U";* la presión en el archivo *"p"*, y así sucesivamente (ver APÉNDICE VI.1 y VI.2). Para las nuevas variables se crean dos archivos *"Cd"* y *"alphaL"* (ver APÉNDICE III.6) en los cuales los valores de las variables se escriben tras ejecutar la herramienta de pre-procesado "leafIndexToFoam" que carga los bosques al dominio (ver APÉNDICE VI.3).

4. Proceso y resultados

4.1. Caso de estudio, la colina Bolund.

El caso Bolund es una colina real, la cual se ha estudiado en múltiples ocasiones mediante procesos experimentales y programas CFD. Ha sido utilizada para validar modelos del flujo sobre terreno complejo. Los efectos de cizalla y la turbulencia son muy importantes en el comportamiento del flujo de aire, y estos se acrecientan en terrenos complejos. Es por esta razón que las herramientas de CFD se están haciendo cada vez más imprescindibles cuando se quiere encontrar la mejor localización para instalar aerogeneradores. Los modelos que los programas de CFD suelen utilizar solo se comprueban con túneles de viento o sobre terreno simple.

Fig 4.1 Vista panorámica de la colina Bolund.

La malla que simula este caso, será la utilizada para testear las aplicaciones desarrolladas. La colina tiene 130 m de largo, 12 m de alto y 75 m de ancho, está situada en la costa, al norte de of Risø DTU (Dinamarca).

Fig 4.2 La colina Bolund.

Esta colina es un buen desafío para cualquier solver de flujos en 3-D, ya que cambia radicalmente de rugosidad, del agua a la hierba, y además posee una gran pendiente entre el nivel del agua y la superficie de la colina que proporciona una cara vertical totalmente expuesta al flujo. Sin embargo como se puede apreciar en las imágenes (Fig 4.1 y Fig 4.2) no hay árboles sobre la colina, por lo que se probarán las aplicaciones mediante la creación de bosques virtuales de forma rectangular (para facilitar la prueba) sobre la colina. Por este motivo el proyecto necesitará de una comprobación empírica para asegurar que funciona correctamente al 100 %.

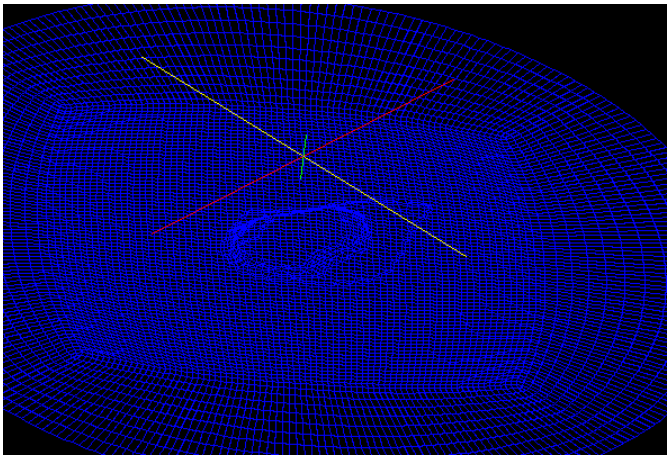


Fig 4.3 Malla correspondiente a la colina Bolund.

4.2. Adición del follaje como variable en OpenFOAM.

Como se ha dicho con anterioridad, el efecto del follaje de los árboles sobre el flujo de aire, se introducirá mediante el coeficiente de arrastre (C_D) y la densidad de hojas por área (α). Ambos parámetros modifican las ecuaciones de Navier-Stokes que se utilizan para resolver el comportamiento del flujo. Este efecto aparece en forma de fuerza exterior (S_u) en la ecuación del momento.

El modelo turbulento k-epsilon también se verá afectado por estos parámetros, y los términos S_k y S_ε (Eqn 2.19 y 2.20) aparecerán en las ecuaciones de transporte, en ambas, en la ecuación de la energía cinética y en la ecuación de disipación de la energía cinética (Eqn 2.15 y 2.16 respectivamente).

Para que el simulador pueda tener en cuenta estos parámetros, primero deben ser declarados e introducidos al caso de estudio, para ello se ha desarrollado una herramienta de pre-procesado que convierte una zona arbolada dada (en un archivo .map) en archivos que OpenFOAM puede leer y trabajar con ellos (archivos “Cd.H” y “alphaL.H” en el directorio “0”).

Para que el solver pueda tener en cuenta las variables C_D y α, primero de todo se deben modificar algunas librerías que el solver utilizará. Las librerías que incorpora OpenFOAM se encuentran en la carpeta “src” dentro del directorio “OpenFOAM”, por ello para evitar modificar el núcleo del programa, se crea un nuevo directorio “src” en el directorio de usuarios ($WM_PROJECT_USER_DIR), donde se crean las nuevas librerías. Las nuevas librerías también se deben compilar antes de correr el simulador, pero se usa el comando “wmake libso” **[14]** en lugar del utilizado para compilar las aplicaciones “wmake”. Los términos S_k y S_ε se añaden modificando los archivos *kEpsilon.C* y *kEpsilon.H* los cuales definen el modelo turbulento k-epsilon. C_D y α son definidos en estos archivos también, al igual que todas las constantes necesarias que utilizan S_k y S_ε.

A continuación se muestra parte del código modificado en estos archivos para introducir las fuerzas externas debidas al efecto del paso del flujo de aire a través del follaje:

```
// Dissipation equation
tmp<fvScalarMatrix> epsEqn
(
  fvm::ddt(epsilon_)
+ fvm::div(phi_, epsilon_)
- fvm::Sp(fvc::div(phi_), epsilon_)
- fvm::laplacian(DepsilonEff(), epsilon_)
==
C1_*G*epsilon_/k_
- fvm::Sp(C2_*epsilon_/k_, epsilon_)
+          fvm::Sp(Cd_*alphaL_/k_*(CEps4_*betap_*pow(mag(U_),3)-
CEps5_*betad_*k_*mag(U_)), epsilon_); //Source Term)

// Turbulent kinetic energy equation
tmp<fvScalarMatrix> kEqn
(
  fvm::ddt(k_)
+ fvm::div(phi_, k_)
- fvm::Sp(fvc::div(phi_), k_)
- fvm::laplacian(DkEff(), k_)
==
G
- fvm::Sp(epsilon_/k_, k_)
+                fvm::Sp(Cd_*alphaL_/k_*(betap_*pow(mag(U_),3)-
betad_*k_*mag(U_)), k_); //Source term)

// Momentum equation
tmp<fvVectorMatrix> UEqn
(
fvm::div(phi, U)
+ turbulence->divDevReff(U)
+ 2*(Omega ^ U)   //coriolis force 2*(omega x U)
+ gravity         //gravity force

+ fvm::Sp(Cdalpha*mag(U)*U,U) //canopy drag force
==
sources(U)
```

```
);
```

El nuevo "solver" desarrollado *("treeFoam")* requerirá las nuevas librerías cuando empiece la simulación. El "solver" incluye el nuevo término (S_u) en la ecuación del momento, la cual se define en el archivo *UEqn.H*. Una vez se compile el código del "solver", la aplicación será creada por el compilador en el directorio $FOAM_USER_APPBIN.

El problema que surge al crear una nueva librería para la nueva aplicación, es que esas librerías están conectadas con una gran cantidad de otros archivos y librerías, por eso es necesario un buen programador para poder encajar las nuevas con todas las demás que también son necesarias. Finalmente tras muchos intentos infructuosos para crear estas, se optó por desbloquear los archivos de OpenFOAM y modificar *kEpsilon.C* y *kEpsilon.H* directamente en el núcleo del programa y añadir ahí las nuevas fuerzas de arrastre debidas al follaje y sus constantes.

4.3. Aplicación de pre-procesado.

4.3.1. Archivos ".map"

Las características de las distintas zonas arboladas que se encuentran en la malla de estudio, se suministran a la simulación a través de un archivo con extensión ".map". En este archivo se hayan: el coeficiente de arrastre (C_D), la densidad de hojas por área (α) y la altura del follaje de los árboles (z_o and Δz). El área sobre el cual se extienden los distintos bosques también esta definida en este archivo, mediante una serie de coordenadas en el plano X-Y que definen polígonos que se corresponden con la situación de los árboles en el mapa.

Las primeras cuatro líneas del archivo ".map" el código no las tiene en cuenta, en ese encabezado aparece el nombre del archivo, las dimensiones de C_D y α, por lo que no es información relevante para la aplicación que convierte el archivo ".map" a un formato de trabajo para OpenFOAM. Tras esas líneas, se definen los polígonos y características del follaje que determinan un bosque. Se separan en grupos, cada grupo representa una zona boscosa y estos están formados por (ver APÉNDICE III.5):

- C_D, α, Δz, z_o y el número de vértices del polígono que define el área del bosque sobre el plano X-Y. Estos se hayan en la primera línea de cada grupo de datos.

- Después del encabezado del grupo de datos, se declaran las coordenadas en 2-D de cada vértice, estas están ordenadas en sentido horario.

Con todos los datos mencionados anteriormente, se define un volumen en el dominio, cuyas celdas contienen los valores de C_D y α. En este caso se definirá un bosque con forma de prisma rectangular, cuya proyección en el plano X-Y será un simple rectángulo.

4.3.2. Aplicación de pre procesado "leafIndexToFoam"

Esta aplicación convierte el archive ".map" a otros archivos con un formato que puede ser leído y cargado al solver de OpenFOAM (*Cd.H* and *alphaL.H*). Para conseguir este objetivo, el código implementado comprobará cada celda de la malla, y si el punto central de esta pertenece a un volumen de los definidos en el archivo ".map", los valores de C_D y α serán copiados a los campos internos de la celda. Cuando toda la malla se haya comprobado, los valores C_D y α se escribirán en los archivos *Cd.H* y *alphaL.H,* los cuales se encuentran en el directorio "0" (ver Fig 3.3).

A continuación se puede ver un esquema que explica el proceso seguido por la aplicación *"leafIndexToFoam"* para leer el archivo ".map" y escribir los archivos *Cd.H* y *alphaL.H.*

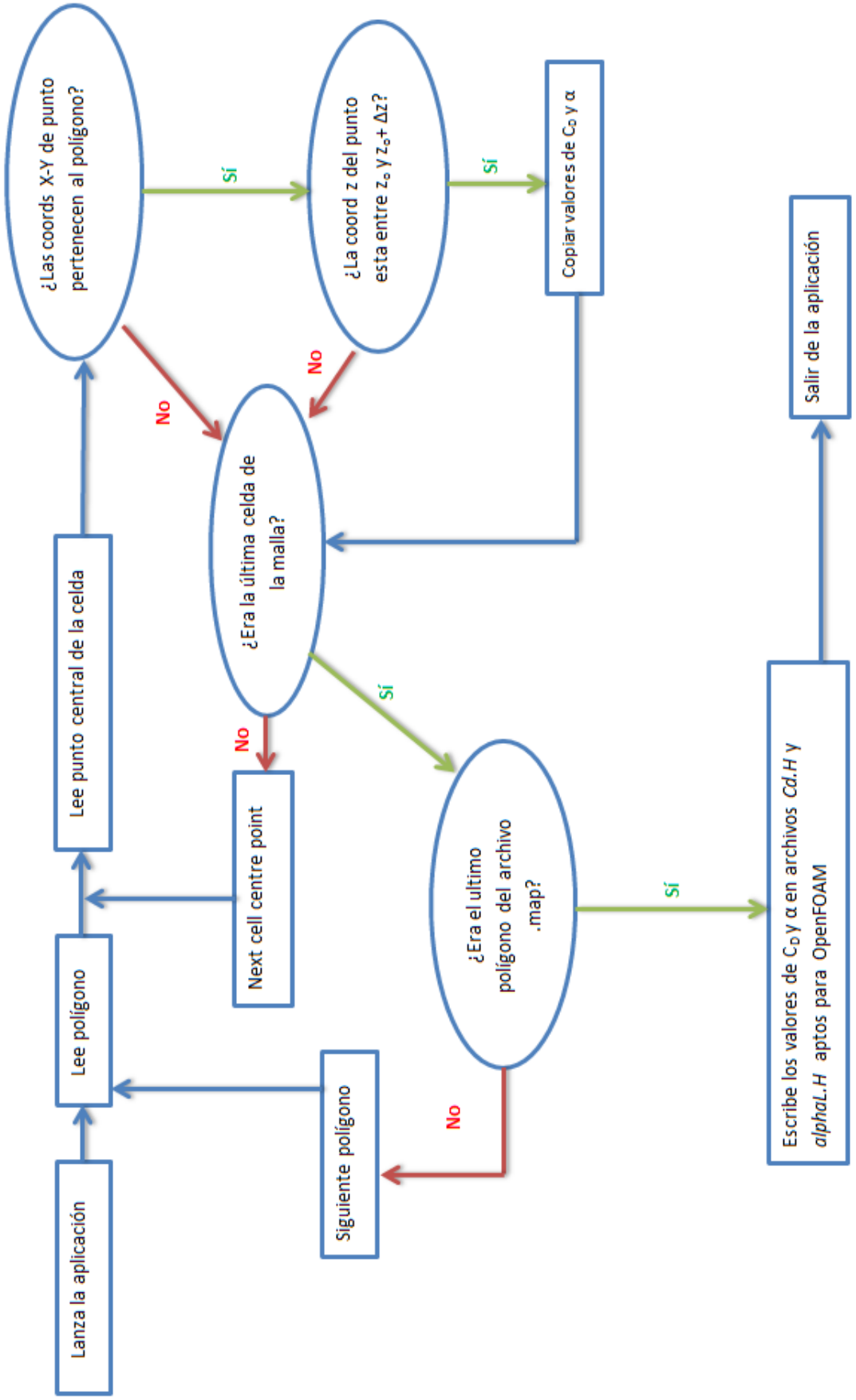


Fig 4.4 Proceso seguido para leer un archivo .map.

Para llevar a cabo el programa mostrado en la imagen anterior (Fig 4.4) se necesitan codificar en C++ los siguientes archivos:

- **leafIndexToFoam.C :** Al compilar este archivo se creará el ejecutable que se usa para lanzar la aplicación, este archivo conecta todos los demás y los invoca según son necesarios. Es el código principal del programa.

- **createCdAlphaL.H :** Como su nombre indica, crea los archivos donde se almacena los valores de C_D y α de cada celda de la malla.

- **Check_inside_cell.H :** En este archivo se codifica la parte del proceso donde se comprueba si el punto central de las celdas pertenecen al volumen definido en el archivo ".map". Repite el proceso para todas las celdas de la malla.

- **Readmapfile_CdAlphaL.H :** Abre el archivo ".map", lee el polígono y los parámetros (C_D,α,z_o,Δz), tras ello invoca al código Check_inside.H. Repite el proceso tantas veces como polígonos son definidos.

El código complete de cada archivo puede verse en el APÉNDICE III.

Se pueden utilizar varios métodos para comprobar si un punto pertenece a un polígono. Se pueden ver algunos de ellos en la referencia **[19]**, en internet. Sin embargo se exponen solo el método seleccionado a continuación.

El método consiste en comparara las coordenadas X-Y de cada pareja de vértices consecutivos que forman el polígono, con las coordenadas X-Y del punto central de la celda.

Existen dos grupos condiciones, cada grupo se compone a su vez de dos requisitos; el punto debe satisfacer alguna de para ser considerado perteneciente al polígono, en otro caso estará fuera.

Condiciones A:

$$(point_x < vertex1_x)\ AND\ (point_x > vertex2_x)$$

$$(point_y < vertex1_y)\ AND\ (point_y > vertex2_y)$$

Condiciones B:

$$(point_x > vertex1_x)\ AND\ (point_x < vertex2_x)$$

$$(point_y > vertex1_y)\ AND\ (point_y < vertex2_y)$$

Este método se puede utilizar para comprobar si un punto pertenece a un polígono en 2-D, por eso en principio es insuficiente para localizar un punto en un volumen. Sin embargo, una vez se comprueba que la proyección del punto central de la celda sobre el plano X-Y pertenece al polígono, únicamente se debe comprobar si la coordenada Z del punto se encuentra entre z_o y (z_o + Δz). Si se cumplen ambas condiciones el punto estará alojado en el interior del volumen.

El código completo que comprueba la existencia de un punto en un volumen puede verse en el APÉNICE III.4 (*Check_inside_cell.H*).

4.3.3. Simulación de zonas arboladas.

Un archivo ".map" se ha creado para poder llevar a cabo la simulación de la aplicación "leafIndexToFoam". Se desea comprobar si el programa carga correctamente los datos y características de los bosques definidos en el archivo ".map" al dominio en estudio. A continuación se muestran las características de algunas especies de árboles más comunes en Europa (Tabla 4.1).

	Picea Mariana	**Pinus Banksiana**	**Aspen forests**
Δz(m)	10	15	10
LAI	9,19	2,22	3,57
C_D	0,15	0,45	0,4
α	0,919	0,148	0,357

Tabla 4.1 Características de la picea mariana, el pinus banksiana y aspen forest. [8]

$$LAI = \int_0^h \alpha \, . \, dz \qquad \textbf{Eqn 4.2}$$

Donde LAI es el índice de hojas por área (LAI = Leaf Area Index) y α es la densidad de hojas por área (m^2/m^3).

La altura a la cual el follaje comienza es (z_o), la cual se asume de 3 m para estas especies árboles. Como la superficie de la colina Bolund está a 12 m sobre el nivel del mar, z_o = 15 m. Se

van a simular dos grupos de árboles distintos; el primero será un bosque de piceas marianas, el cual abarcará una superficie de 1600 m^2, el segundo grupo se compondrá de pinus banksiana, y se extenderá sobre 400 m^2 (Ver APÉNDICE III. Map file).

La aplicación grabará los valores de C_D y α en los archivos *Cd.H* y *alphaL.H* respectivamente, los cuales se encuentran en el directorio "0". Estos archivos deben ser reinicializados para volver a ejecutar la aplicación (Ver APÉNDICE III.6).

En las siguientes imágenes se puede observar el valor α asignado a cada celda de la malla, tras correr la aplicación "leafIndexToFoam". Las celdas en rojo (α = 0,919) representan el bosque de piceas marianas, y aquellas en verde (α = 0,148) se corresponden con el grupo de árboles de pinus banksiana.

Dado que el código solo comprueba el punto central de cada celda, si este está dentro del volumen, se le asignará el valor de α a la celda completa; y lo no mismo ocurre en caso contrario. Por lo que la aplicación puede declarar como follaje parte de la celda que en realidad debería quedar fuera del volumen; o viceversa, asignar un valor nulo de α a una zona donde en realidad hay follaje. La magnitud de este error será menor si se define una malla mas fina, o el bosque a insertar es más grande.

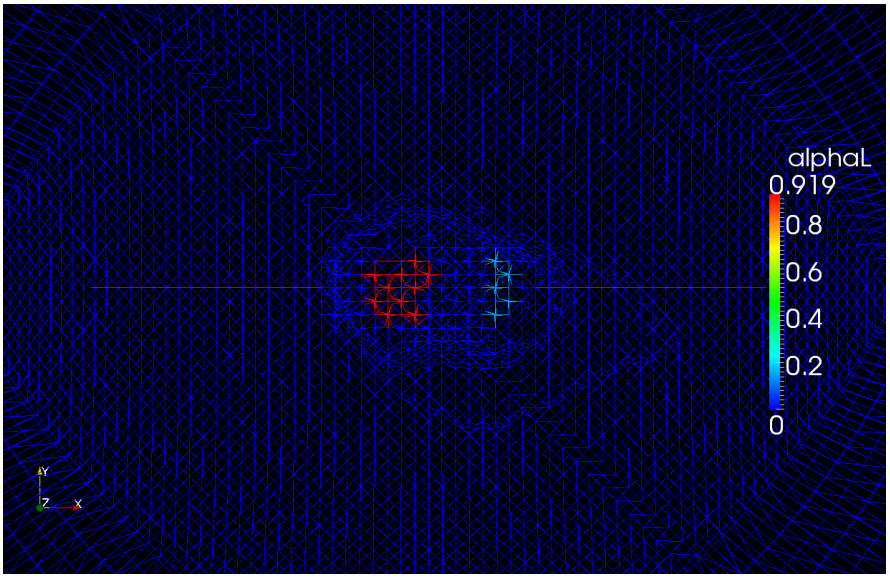


Fig 4.5 Representación de "alphal" en "Paraview" vista normal al eje Z.

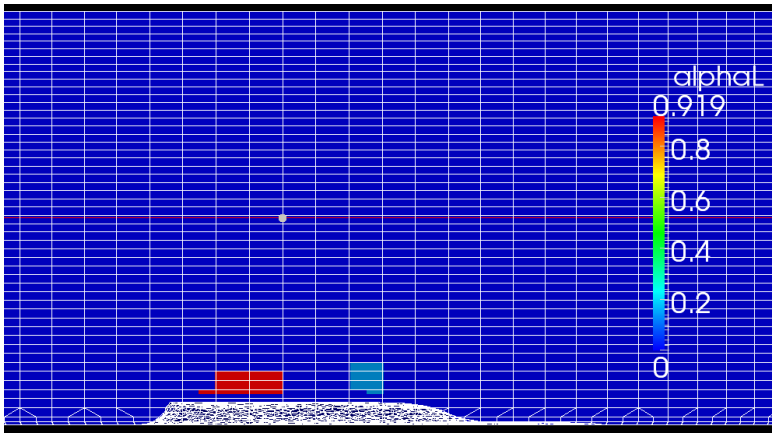


Fig 4.6 Representación de "alphal" en "Paraview" vista normal al eje Y.

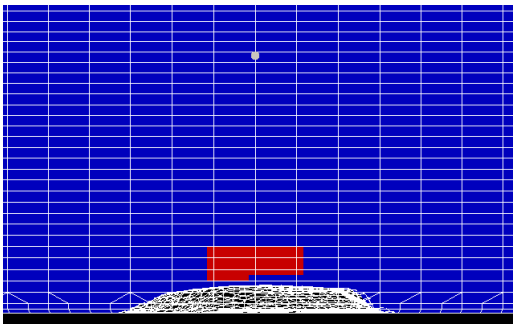


Fig 4.7 Representación de "alphal" en "Paraview" vista normal al eje X a -30 m.

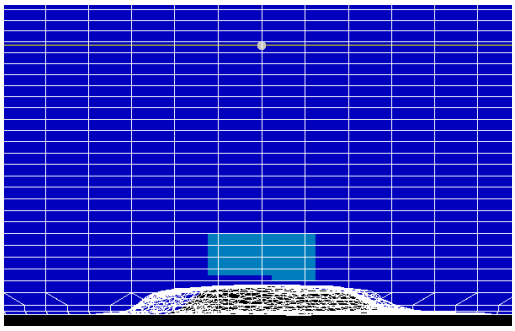


Fig 4.8 Representación de "alphal" en "Paraview" vista normal al eje X a 45.

4.4. TreeFoam "solver" test.

Una vez las nuevas librerías han sido compiladas con los cambios necesarios para incorporar el efecto de los árboles sobre el flujo de aire, y la herramienta de pre-procesado que permite incorporar los bosques al dominio desde los archivos ".map", el último paso es comprobar si el solver ("treeFoam") funciona correctamente.

La primera simulación no se pudo llevar a cabo, ya que el solver no alcanzaba la convergencia. Para detectar si el problema se hallaba en las librerías o en el programa, se recodificó para que utilizará las librerías estándar de OpenFOAM. Las librerías estándar no incorporan los términos adicionales, pero desde el punto de vista informático sirven para detectar el fallo. Se comparó el nuevo solver "treeFoam" con uno estándar "simpleFoam", ambos usaron las mismas librerías y se pudo determinar que el fallo estaba en las nuevas librerías turbulentas. Por este motivo y como se mencionó con anterioridad, se decidió modificar los archivos *kEpsilon.C* y the

kEpsilon.H dados por OpenFOAM y así usar las librerías estándar añadiendo ahí las fuerzas externas debidas al follaje de los bosques.

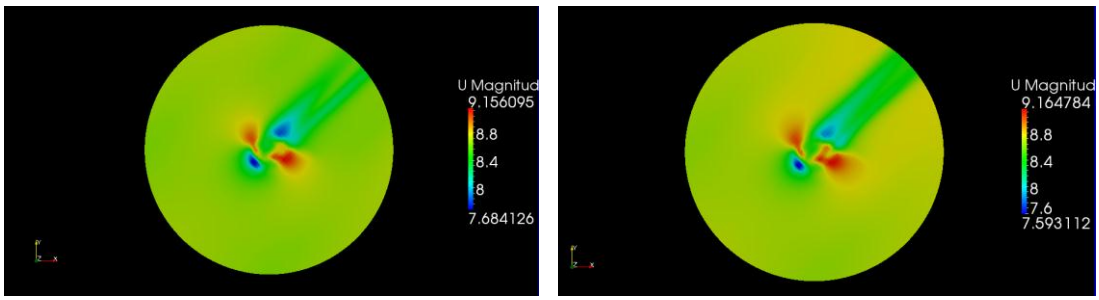
Para comprobar el funcionamiento del nuevo solver, se realizarán 4 simulaciones:

1. Simulación del caso Bolund con el estándar solver "simpleFoam".
2. Simulación del caso Bolund con el solver "treeFoam" sin añadir bosques sobre la colina.
3. Simulación del caso Bolund con "treeFoam" donde se inserta un bosque sobre la colina con un valor de α de 0,919 (m^2/m^3).
4. Simulación del caso Bolund con "treeFoam" donde se inserta un bosque sobre la colina con un valor de α de 1,838 (m^2/m^3).

En este informe se muestran cuatro imágenes de las simulaciones. Tres de ellas son del perfil de velocidades sobre el plano X-Y a tres alturas diferentes (20,10 y 40 metros); y la cuarta corresponde al perfil de velocidades del viento a lo largo de la dirección (1,1,0).

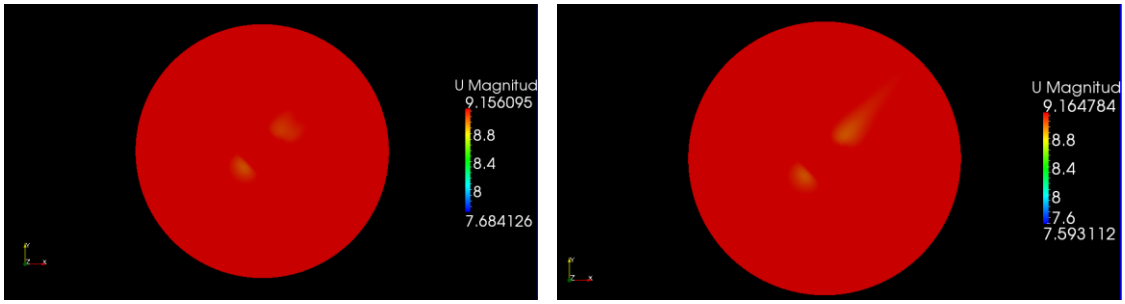
4.4.1. Sin bosques o zonas arboladas definidas en el dominio.

Ya que "simpleFoam", un solver dado por OpenFOAM, es el programa del que nace "treeFoam"; se utiliza como herramienta de referencia para comprobar el funcionamiento del nuevo solver. Las dos primeras simulaciones mencionadas anteriormente, se compararán, y si el programa funciona correctamente, el perfil de velocidades al que converjan ambos "solvers" serán iguales. En las siguientes imágenes se pueden observar los resultados obtenidos por los dos.



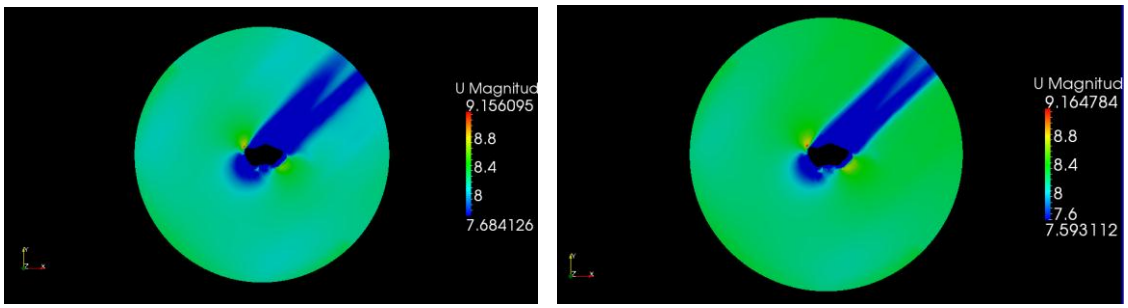
"simpleFoam" solution "treeFoam" solution

Fig 4.9 Perfil de velocidades (m/s) normal al eje Z a 20 m.



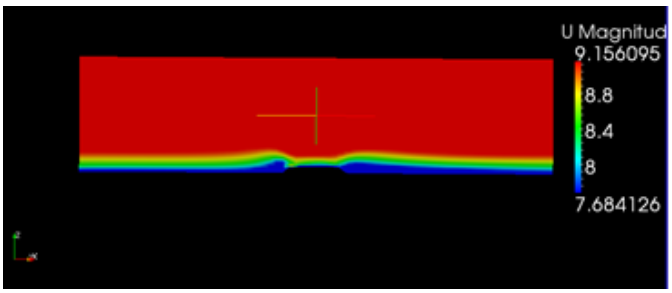
"simpleFoam" solution "treeFoam" solution

Fig 4.10 Perfil de velocidades (m/s) normal al eje Z a 40 m.

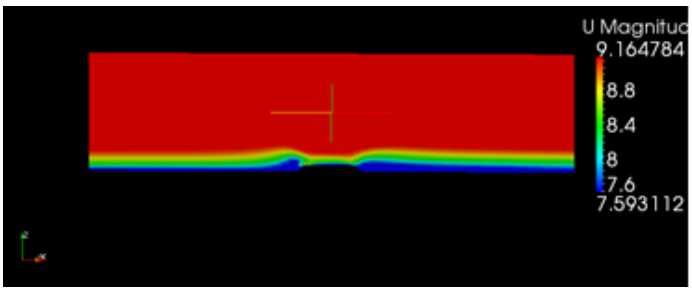


"simpleFoam" solution "treeFoam" solution

Fig 4.11 Perfil de velocidades (m/s) normal al eje Z a 10 m.



"simpleFoam" solution



"treeFoam" solution

Fig 4.12 Perfil de velocidades (m/s) a lo largo de la dirección del viento.

Como se puede apreciar en las imágenes (Fig 4.11, 4.12, 4.13 y4.14), ambos solvers muestran prácticamente el mismo perfil de velocidades, las pequeñas diferencias se deben a la diferencia en el error de convergencia que alcanzan cada uno. Después de comprobar que "treeFoam" funciona correctamente, el siguiente paso es asegurarse de que tiene en cuenta los bosques de la manera esperada.

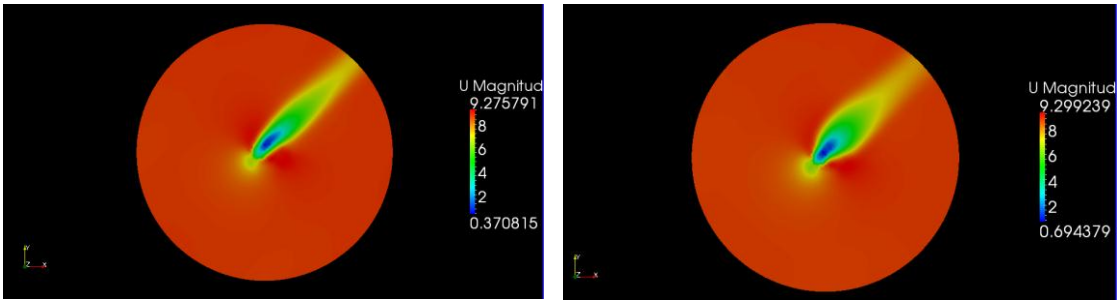
4.4.2. Simulación con "treeFoam" añadiendo bosques al dominio.

Las simulaciones tercera y cuarta, mostrarán el perfil de velocidades obtenido por el solver "treeFoam" una vez se ha utilizado "leafIndexToFoam" para insertar el bosque en el dominio. Se comprobará si los términos de fuerzas externas incluidos en las ecuaciones del momento funcionan de una manera racional. El archivo ".map" a utilizar se puede observar en el APÉNDICE III.5.

Características del bosque	
Δz(m)	50
z_o(m)	15
C_D	0,15
α(m^2/m^3)	0,919 – 1,838
Area(m^2)	1600

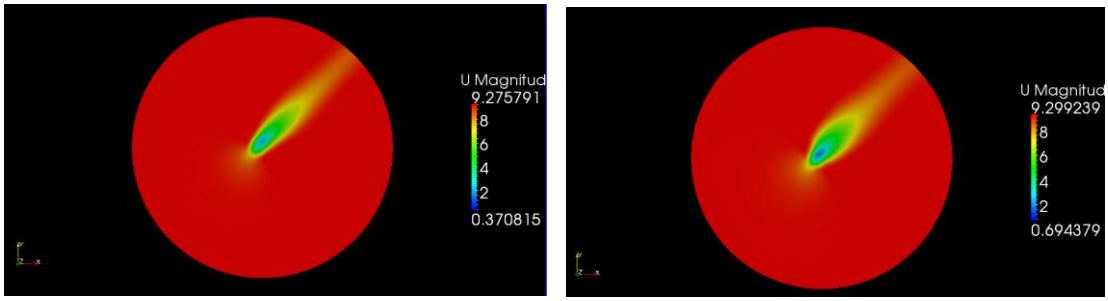
Tabla 4.2 Características del bosque utilizado en la simulación. [8]

Se simularán un grupo de picea mariana, de 53 m de alto de los cuales 3 m pertenecen al tronco y no será tenido en cuenta como follaje. El valor de α se dobla para comparar ambas simulaciones. Solo se modifica α, ya que tanto α como C_D están multiplicando en las ecuaciones que incluyen el efecto de los árboles, por lo que modifican el sistema de ecuaciones de la misma manera.



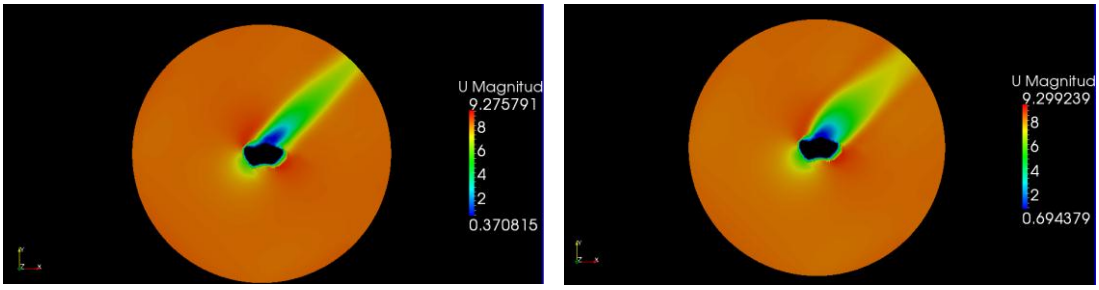
α =0,919 α =1,838

Fig 4.13 Perfil de velocidades (m/s) normal al eje Z a 20 m.



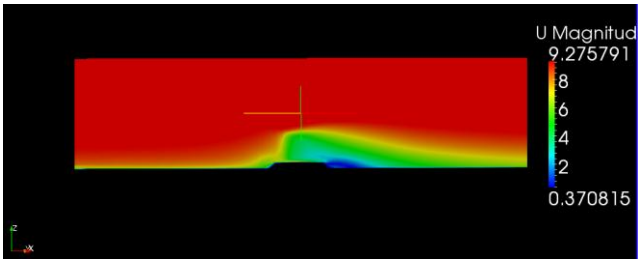
α =0,919 α =1,838

Fig 4.14 Perfil de velocidades (m/s) normal al eje Z a 40 m.

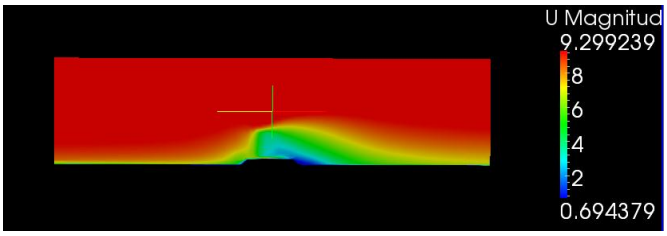


α =0,919 α =1,838

Fig 4.15 Perfil de velocidades (m/s) normal al eje Z a 10 m.



α =0,919



α =1,838

Fig 4.16 Perfil de velocidades (m/s) a lo largo de la dirección del viento.

Primero de todo, se puede apreciar una gran diferencia entre la solución obtenida por "simpleFoam" o por "treeFoam" sin zonas boscosas, frente a cualquiera de los dos perfiles de velocidades que ofrecen las simulaciones con "treeFoam" añadiendo bosques. Por lo que el nuevo solver tiene en cuenta el efecto que produce el follaje cuando el viento lo atraviesa.

La segunda conclusión que se puede obtener de las dos últimas simulaciones, es que si se observan las figuras anteriores (Fig 4.15, 4.16, 4.17, 4.18), se puede afirmar que para valores mayores de α, el flujo de aire se ve distorsionado en mayor medida cerca del bosque. Por eso se declarar que se ha testeado "treeFoam" y funciona correctamente. No obstante para que pueda ser 100 % comprobado, se deben enfrentar simulaciones informáticas con CFD frente a los datos empíricos obtenidos en un experimento.

5. Conclusión.

Para empezar, este proyecto tenía una dificultad añadida, el software empleado, OpenFOAM, no es uno de los programas habituales que se estudian en las aulas, principalmente porque su manejo no es tan sencillo como el de otros programas CFD y trabaja sobre el sistema operativo Linux. Por lo que este proyecto y los que de él deriven, requieren de conocimientos básicos con Linux, con el lenguaje de programación C++ y especialmente con OpenFOAM.

Sin embargo, como OpenFOAM es un software libre, es un programa CFD perfecto para desarrollar nuevos "solvers", sin incurrir en los gastos de adquirir un software comercial. Por otro lado ya que el código fuente está abierto, los usuarios pueden modificar las aplicaciones existentes para añadir mejoras o simplemente adaptarlos a sus casos particulares más específicos. Por último, el trabajo de cada usuario se puede compartir con la comunidad online para que lo prueben e incluso aconsejen o sugieran algunas mejoras o correcciones.

Se han desarrollado dos aplicaciones en este proyecto. La primera de ellas ("leafIndexToFoam") es una aplicación de pre procesado la cual carga las zonas arboladas al domino de estudio. Los valores de α y C_D se leen del archivo ".map" y se escriben satisfactoriamente en los archivos *alphaL.H* y *Cd.H*; gracias a la herramienta *paraView* se pueden observar estos valores de manera gráfica para cada celda de la malla, como se muestra en las figuras (Fig 4.7, 4.8, 4.9 y 4.10). Con este visualizador se puede ver como el bosque cargado al dominio es fiel al descrito en el archivo ".map".

La segunda aplicación es "treeFoam", este es un "solver" el cual resuelve el sistema de ecuaciones de Reynolds, aplicando un modelo turbulento. En este caso se añadió al "solver" las ecuaciones pertinentes para simular el efecto producido en el flujo de aire cuando atraviesa el follaje de los árboles. Las imágenes (Fig 4.15, 4.16, 4.17 y 4.18) muestran este efecto, el perfil de velocidades varía de manera apreciable cuando el viento se topa con un conjunto de árboles. Se puede observar también que cuanto mas frondoso es el bosque (el coeficiente α es mayor) el perfil de velocidades se distorsiona en mayor medida.

Para poder presentar "treeFoam" como un "solver" 100% fiable, se debería realizar un ensayo experimental y comparar los resultados obtenidos con la simulación del ensayo. Si los resultados son semejantes se podría confirmar como un "solver" CFD totalmente válido. Sin embargo esta tarea no es sencilla porque incluso la obtención del coeficiente de arrastre (C_D)

no esta claramente definida por el momento para cada especie de árbol, por lo que suele ser estimada.

6. Líneas futuras.

El objetivo mas importante que se propone para seguir mejorando este “solver”, es automatizar la ejecución de una simulación. De esta manera se permitirá al usuario realizar varias simulaciones con diferentes perfiles de flujo de aire, cambiando su velocidad y dirección, sin tener que preparar todos los archivos necesarios para simular un caso en OpenFOAM. El software incorpora una aplicación “*python*” facilita esta automatización, este sería un buen punto de partida.

En segundo lugar se sugiere añadir también el efecto que produce un aerogenerador sobre el flujo de aire, ya que normalmente en una granja eólica se hallan varios de estos instalados y cada uno distorsiona el flujo de aire.

Por último se debería cambiar el proceso del estado estacionario al modelo transitorio, para poder realizar simulaciones más fieles a las situaciones reales.

7. Bibliografía.

[1] International Energy Agency (IEA).

[2] H.K. Versteeg and W. Malalasekera. *An introduction to Computational Fluid Dynamics. The Finite Volume Method.* Longman Group Ltd, 1995.

[3] Lars Davidson. *An introduction to turbulence Models Göteborg*, Sweden: Chalmers University of Technology, 2011.

[4] D. Poggi, A. Porporato, C. Ridolfi, J.D. Albertson and G.G. Katul. *The effect of vegetation density on canopy sub-layer turbulence.* Kluwer Academic Publishers, 2004.

[5] B. Marcolla, A.Pitacco and A.Cescatti. *Canopy architecture and turbulence structure in a coniferous forest.* Kluwer Academic Publishers, 2003.

[6] A.Cescatti and B.Marcolla. "Drag coefficient and turbulence intensity in conifer canopies." *Agricultural and Forest Meteorology*, vol. 121, pp. 197-206, 2004.

[7] B.Dalpé and C.Masson. "Numerical simulation of wind flow near a forest edge." *Journal of Wind Engineering and Industrial Aerodynamics,* vol. 97, pp. 228-241, 2009.

[8] B.Dalpé and C.Masson. "Numerical Study of Fully Developed Turbulent Flow Within and Above a Dense Forest." *Wind Energy,* vol.11, pp. 503-515, 2008.

[9] Xabier Pedruelo. *Modeling of Wind Flow over Complex Terrain Using OpenFOAM.* Vattenfal Research and Development AB, 2009.

[10] Eric Furbo. *Evaluation of RANS turbulence models for flow problems with significant impact of boundary layers.* Uppsala University, 2010.

[11] Benjamin Martinez. Wind resource in complex terrain with OpenFOAM. Risø DTU, National Laboratory for Sustainable Energy, 2011.

[12] Håkan Nilsson. *Basic of C++ and object orientation in OpenFOAM.* Chalmers University.

[13] Jan Skansholm *C++ From the Beguinning*. Addison Wesley Longman, 1977.

[14] *OpenFOAM User´s Guide*. OpenFOAM Foundation, 2011.

[15] *OpenFOAM Programmer´s Guide*. OpenFOAM Foundation, 2011

[16] El Ideal. "Reserva 'ricotí ', un territorio protegido para la Alondra de Dupont". Internet: http://www.ideal.es/granada/20120708/local/granada/reserva-ricoti-territorio-protegido-201207081726.html

[17] Godfrey Boyle. *Renewable Energy: Power for a Sustainable Future.*Oxford.

[18] Wikipedia. "Wind rose". Internet: http://en.wikipedia.org/wiki/Wind_rose

[19] Paul Bourke. "Determining if a point lies on the interior of a polygon". Internet: http://local.wasp.uwa.edu.au/~pbourke/geometry/insidepoly/

[20] *OpenFOAM programming tutorial.* Politecnico de Milano, Chalmers.

[21] Wind Energy THE FACTS. "The Annual Variability of Wind Speed". Internet: http://www.wind-energy-the-facts.org/en/part-i-technology/chapter-2-wind-resource-estimation/local-wind-resource-assessment-and-energy-analysis/the-annual-variability-of-wind-speed.html

[22] WIND POWER PROGRAM. "Wind Turbine power output variation with steady wind speed." Internet: http://www.wind-power-program.com/turbine_characteristics.htm