UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DEL DISEÑO

Máster Universitario en Ingeniería Aeronáutica

Trabajo de Fin de Máster

Desarrollo e implementación de un algoritmo paralelo de identificación de vórtices en simulaciones DNS

Autor:

Jose Javier Aguilar Fuertes

Tutor:

D. Sergio Hoyas Calvo

Cotutor:

D. Lluis García Raffi

Valencia, Septiembre 2019

Resumen

En este trabajo se ha desarrollado e implementado en Fortran un algoritmo en paralelo para detectar y seguir la evolución temporal de los vórtices presentes en un flujo turbulento calculado mediante simulaciones DNS.

Para agregar los vórtices, que son estructuras coherentes tridimensionales, el algoritmo desarrollado primero agrupa regiones coherentes bidimensionales en planos para convertirlas en nodos de un grafo. Las aristas de este grafo se sitúan entre nodos de planos consecutivos que coincidan en al menos un punto en dos dimensiones. Una vez construido el grafo el programa obtiene sus componentes conexos, y cada uno de ellos será un vórtice.

Tras agregar los vórtices para una serie de pasos temporales, el programa compara las estructuras presentes en pasos temporales consecutivos, usando sus características geométricas en primer lugar y comparando punto por punto después, para así estudiar la evolución temporal de los torbellinos presentes en el campo fluido.

La evolución temporal de cada vórtice y sus interacciones con otros se guardan en una base de datos, desde la que se pueden representar los vórtices mediante grafos representativos de su vida e interacciones, imágenes y animaciones.

Palabras Clave: Turbulencia, Vórtices, Algoritmo, Grafo

Resum

En aquest treball s'ha desenvolupat i implementat en Fortran un algoritme en paral·lel per detectar i seguir l'evolució temporal dels vòrtices presents en fluxos turbulents obtinguts mitjançant simulacions DNS.

Per agregar els vòrtices, que són estructures coherents tridimensionals, el algoritme desenvolupat primer agrupa regions coherents bidimensionals per convertir-les en els nodes d'un grafs. Les aristes d'aquest graf es situen entre els nodes de plans consecutius que coincidisquen en almenys un punt en dos dimensions. Després de construir el graf, el programa agrupa els seus components conexos, i cadascú serà un vòrtex.

Després d'agregar els vòrtices per una sèrie de passos temporals, el programa després compara les estructures presents en passos temporals consecutius, utilitzant les seues característiques geomètriques primer i comparat punt a punt després, per així estudiar l'evolució temporal dels vòrtices presents al camp fluït.

L'evolució temporal de cadascun dels vòrtices i les seues interaccions amb els altres es guarden en una base de dades, des de la qual es poden representar els vòrtices mitjançant grafs representatius de les vides i interaccions, imatges i animacions.

Paraules Clau: Turbulència, Vòrtices, Algoritme, Graf

Abstract

In this project a parallel vortex identification and tracking algorithm has been developed and implemented in Fortran to detect and follow the evolution of vortices present in fluid fields obtained as a result of DNS simulations.

To aggregate vortices, which are coherent three-dimensional structures, the developed algorithm first groups coherent regions in two-dimensions in every plane which are the nodes of a graph. This graph will has its edges between every two nodes in consecutive planes which share at least one point in two dimensions. Once the graph is built, the program groups its connected components, each being representative of a vortex.

The program performs this operations for a series of time steps and then compares the structures present in pairs of consecutive time steps, first using the structures' geometric features and then performing a point to point comparison, obtaining then the evolution of the vortices present in the fluid field.

The evolution of the vortices and their interactions with others are saved in a database, from which vortices can be represented by means of graphs representing their lives and interactions, images and animations.

Keywords: Turbulence, Vortices, Algorithm, Graph

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Los flujos turbulentos presentan como característica definitoria la presencia de significantes variaciones en sus campos de velocidades, y estas variaciones son además irregulares en el tiempo y en el espacio. Pese a la presencia de estas variaciones, es posible extraer información útil de los flujos turbulentos tanto en estudios experimentales como en simulaciones computacionales.

En líneas generales, un flujo se puede determinar como turbulento a partir de su número de Reynolds, que compara las fuerzas inerciales (en el numerador) con las viscosas (en el denominador) en el fluido. Cuando las fuerzas viscosas dominan sobre las inerciales el fluido se comporta de manera ordenada originando los flujos conocidos como laminares, en los que las distintas capas del fluido no se mezclan y pueden tener una solución analítica en geometrías simples. Sin embargo, a números de Reynolds elevados, las fuerzas inerciales dominan sobre las viscosas y producen inestabilidades en el flujo y torbellinos caóticos: la turbulencia. Estas inestabilidades, con un carácter fuertemente no lineal, difuminan la clara distinción entre capas presente en los flujos laminares, e impiden su resolución analítica. Por lo tanto, las herramientas principales para el desarrollo científico en el conocimiento de la turbulencia son los estudios experimentales y computacionales.

Estas distinción entre los dos tipos de flujo fue formalizada por Osborne Reynolds [1], quien mediante estudios experimentales determinó que las transiciones de laminar a turbulento ocurrían a $Re \approx 2 \cdot 10^3$ (con el diámetro como dimensión característica) para flujos en tuberías y $Re \approx 1 \cdot 10^5$ (con la longitud de la placa como dimensión característica) para flujos sobre una placa plana, aunque la transición no es inmediata y existe una región de transición alrededor de esos valores.

El número de Reynolds está definido como:

$$Re = \frac{UL}{\nu} \tag{1.1}$$

Donde:

- U es la velocidad relativa entre el fluido y el cuerpo
- L es la longitud característica

- ν es la viscosidad cinemática del fluido

Debido a los bajos valores de viscosidad cinemática del aire y del agua, a longitudes o velocidades moderadas el número de Reynolds es lo suficientemente alto como para que el flujo sea turbulento, presentándose así la turbulencia tanto en situaciones cotidianas (el flujo de agua en un río o el humo de una chimenea) como en la gran mayoría de problemas de ingeniería: desde los flujos alrededor de embarcaciones, aeronaves y automóviles hasta la generación de energía eléctrica mediante turbinas eólicas, hidráulicas o de vapor. Tal es la importancia de los flujos turbulentos en ingeniería que se calcula que el 5 % de las emisiones de CO_2 anuales producidas por la humanidad son consecuencia de las pérdidas en flujos turbulentos cerca de la pared [2]. Como flujo laminar importante en problemas de ingeniería destaca la lubricación.

Es por esto que los avances en el estudio de la turbulencia son tan importantes en el mundo de la ingeniería, al transportarse sus avances aguas abajo a una gran variedad de campos de la ingeniería basados en la mecánica de fluidos.

En el campo de la mecánica de fluidos destacan las ecuaciones de Navier-Stokes, desarrolladas a principios del siglo XIX con el objetivo de describir los campos de velocidades y las características del flujo en un volumen fluido. Estas ecuaciones surgen de la aplicación de la segunda ley de Newton al movimiento del fluido. Expresadas en notación de Einstein:

$$\frac{\partial U_i}{\partial t} + U_j \frac{\partial U_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial P}{\partial x_i} + \nu \frac{\partial^2 U_i}{\partial x_j^2} \tag{1.2}$$

donde U_i son las componentes del vector velocidad, P es la presión, ρ la densidad y el subíndice i toma valores de 1 a 3, uno para cada una de las dimensiones cartesianas.

Como demostró Poincaré, sistemas no-lineales simples pueden mostrar comportamiento caótico incluso si están gobernados por ecuaciones diferenciales deterministas. Lorenz, posteriormente, aplicó esta idea a la turbulencia a través de sus trabajos en predicción meteorológica [3], demostrando su caracter caótico.

Pese al desconocimiento sobre la existencia de soluciones analíticas y al caracter caótico de las ecuaciones, sí que existen conceptos fundamentales para la descripción de la turbulencia obtenidos a través de derivaciones analíticas. Entre ellos destaca la cascada de energía de Kolmogorov.

1.1 Cascada de energía de Kolmogorov

La teoría de la cascada de energía desarrollada por Kolmogorov es uno de los desarrollos más importantes en el estudio de la turbulencia debido a su capacidad de hacer predicciones simples y fundamentales.

Suponiendo un flujo a alto número de Reynolds, esta teoría afirma que la energía se introduce en el fluido en las escalas más grandes, a números de Reynolds elevados, del orden del número de Reynolds del flujo, donde la viscosidad no tiene un papel importante,

por lo que no pueden disipar energía. Estos vórtices son inestables y se rompen en vórtices más pequeños, transfiriendo completamente su energía a éstos. Este proceso se repite hasta las escalas más pequeñas donde la viscosidad actúa y disipa estas estructuras más pequeñas. Esta transferencia de energía a través de las distintas escalas de la turbulencia es conocida como la *cascada de energía turbulenta*.

Partiendo de un torbellino, definido como una región coherente en la que existe un movimiento turbulento del fluido, con una longitud l, en un estado del flujo estadísticamente estacionario y a un número de Reynolds local $Re_l = l \cdot U_l/\nu$ suficientemente grande como para que el efecto de la viscosidad sea despreciable, este tendrá una energía cinética turbulenta del orden del cuadrado de su velocidad U_l y un tiempo característico t_l marcado por su longitud y su velocidad.

Sabiendo que $k \sim U_l^2$ y que $t_l \sim l/U_l$, el cociente de ambos será su disipación de energía cinética turbulenta, ε, independiente de la viscosidad:

$$\varepsilon \sim U_l^3/l \rightarrow U_l \sim (l\ \varepsilon)^{1/3} \tag{1.3}$$

Conforme la longitud característica se hace más pequeña, se llega a un tamaño en el que el tiempo característico del torbellino (l/u_l) es igual a su tiempo de disipación (l^2/ν): la escala de Kolmogorov, η. Sustituyendo la velocidad con la expresión obtenida en la Ecuación (1.3) e igualando estos dos tiempos característicos, se obtiene una expresión para dicha escala:

$$\frac{\eta^2}{\nu} = \frac{\eta}{(\eta\ \varepsilon)^{1/3}} \rightarrow \eta = \left(\frac{\nu^3}{\varepsilon}\right)^{\frac{1}{4}} \tag{1.4}$$

Basándose en esto, la teoría distingue tres regiones asociadas a las diferentes escalas de longitud:

- Escalas integrales: escalas comparables a la dimensión característica del flujo, donde ocurre la transferencia de energía desde el exterior hacia el flujo. El límite inferior se sitúa en $l_\varepsilon \sim L/6$, siendo L la dimensión característica del flujo.

- Escalas inerciales: escalas no influenciadas ni por la inyección de energía externa ni por la disipación viscosa. Estas escalas son isótropas y universales, por lo que la información respecto al dominio y la geometría no llega a estas escalas. Estas son las escalas en las qe se cumple la Ecuación (1.3) respecto a la disipación de energía cinética turbulenta. Su límite inferior es la longitud de Kolmogorov.

- Escalas disipativas: todas aquellas donde $l \leq \eta$, dominadas por efectos viscosos. En estas escalas la viscosidad disipa la energía.

Las tres regiones en las que separa las distintas escalas de turbulencia están representadas esquemáticamente en la Figura 1.1.

Para las dos regiones con escalas más pequeñas, o lo que es lo mismo, para $l < l_\varepsilon$, Kolmogorov formuló su primera hipótesis de similaridad, en la que afirma que su comportamiento solo depende de ε y ν. Para las escalas inerciales, además, formuló su segunda hipótesis de similaridad, afirmando que su comportamiento sólo está determinado por su disipación de energía cinética turbulenta [4].

Pese a que Kolmogorov derivó las consecuencias estadísticas de la cascada de energía, esta teoría no explica los fenómenos físicos que ocurren en cada uno de los vórtices durante la cascada de energía.

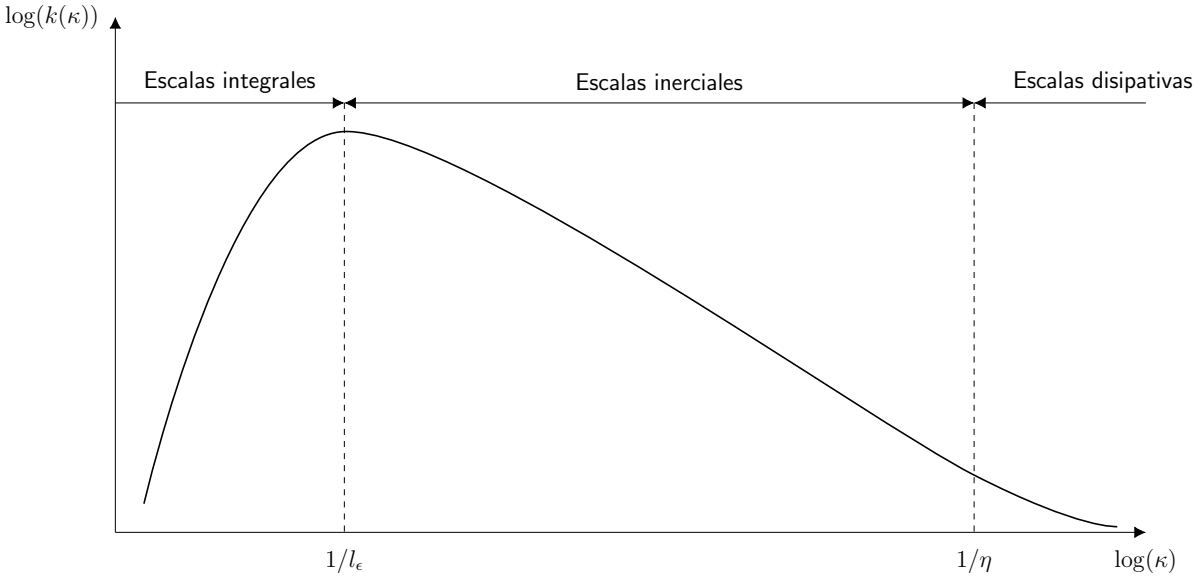


Figura 1.1: Representación esquemática de la cascada de energía de Kolmogorov, con el logaritmo del número de onda en el eje horizontal y el logaritmo de la energía cinética turbulenta en el eje vertical.

1.2 Flujos turbulentos de pared

La mayoría de los flujos turbulentos relevantes por sus aplicaciones ingenieriles están definidos por la interacción con una pared. Esta interacción se basa en la condición de no deslizamiento, que establece que el vector velocidad del fluido en la frontera debe ser igual al de la pared con la que hace contacto. La existencia de esta condición de contorno choca con la homogeneidad e isotropía necesarias de las que parte la teoría de Kolmogorov, ya que el comportamiento del flujo varía según la dirección en la que se estudie.

La condición de no deslizamiento, además, da lugar a la aparición de esfuerzos viscosos cerca de la pared τ_w que pueden expresarse como:

$$\tau_w = \nu \left.\frac{\partial U}{\partial y}\right|_w \tag{1.5}$$

Donde y es la dirección perpendicular a la pared. Estos esfuerzos viscosos tienen unidades de velocidad al cuadrado, por lo que se puede definir una magnitud u_τ conocida como velocidad de fricción:

$$u_\tau^2 = \nu \left.\frac{\partial U}{\partial y}\right|_w \tag{1.6}$$

Esta velocidad de fricción permite definir las velocidades y distancias a la pared de forma adimensional, tal que:

$$U^+ = \frac{U}{u_\tau},\ y^+ = \frac{y\ u_\tau}{\nu} \tag{1.7}$$

Por último, la velocidad de fricción permite definir también un número de Reynolds de fricción basado en ésta y en la longitud característica del problema:

$$Re_\tau = \frac{u_\tau\ h}{\nu} \tag{1.8}$$

Una vez adimensionalizadas las variables de trabajo, el estudio del flujo medio cerca de la pared se vuelve más intuitivo. A valores de y^+ bajos, muy cerca de la pared, $\partial U/\partial y$ es lineal dando lugar a la conocida como subcapa viscosa.

A valores de $y^+ > 100$ el comportamiento de la capa límite se rige mediante la ley logarítmica de von Kármán [4]:

$$U^+ = \frac{1}{\kappa}\log y^+ + A \tag{1.9}$$

donde κ y A son constantes que pueden ser determinadas mediante estudios experimentales o simulaciones DNS. La constante de von Kármán, κ, ronda valores cercanos a 0,384 y $A \approx 4{,}2$ para paredes lisas. El desarrollo que se ha de seguir para obtener esta expresión está presente en la primera sección del séptimo capítulo de [4]. Cabe destacar que entre estas dos subcapas existe una región *buffer* en la que el flujo se adapta de una a otra, sin regirse por ninguna de las dos leyes presentadas.

En estas regiones existe un gradiente de velocidades que, según la teoría clásica de turbulencia de pared, provoca una transferencia de energía en sentido contrario a la cascada de energía cinética turbulenta: la energía se produce cerca de la pared mediante el gradiente de velocidades, y se transfiere a escalas más grandes más lejos de la pared donde se disipa siguiendo la cascada de energía de Kolmogorov.

Los flujos turbulentos de pared pueden ocurrir de varias formas:

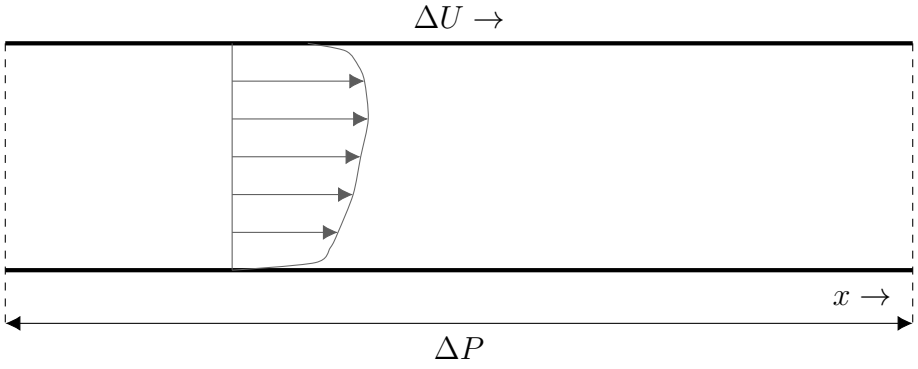
- **Canales**: las capas límite en cada una de las paredes se desarrollan hasta entrar en contacto en un flujo completamente desarrollado, considerando las dimensiones paralelas a la pared infinitas. Todo el dominio fluido está influenciado por las dos paredes. El fluido puede moverse mediante una diferencia de velocidades entre las placas (Couette), un gradiente de presiones (Poiseuille) o una combinación de ambas (Couette-Poiselle). La Figura 1.2 (a) contiene una representación esquemática de un flujo de Couette-Poiselle con gradiente de presiones favorable.

  El carácter infinito en las dimensiones del plano de la pared permite el uso de esquemas numéricos periódicos en estas dos direcciones, más baratos computacionalmente que los basados en diferencias finitas o diferencias finitas compactas.

- **Tuberías**: análogas a los canales pero en coordenadas cilíndricas, donde solo una de las direcciones en infinita, siendo la otra periódica. Los estudios experimentales a mayor número de Reynolds disponibles en la actualidad se realizan en este tipo de flujo, destacando los experimentos de Marusic *et al* [5], con $Re_\tau \sim 10^5$.

- **Capas límite**: el flujo presenta una pared inferior pero no superior, zona en la que se establece una condición de esfuerzo cortante nulo. Al igual que en los canales, estas capas límite también pueden estar movidas por una diferencia de velocidad o presión. Sin embargo, a diferencia del estudio de flujo en canales, donde se estudia generalmente el flujo completamente desarrollado, el estudio de capas límite busca estudiar su evolución. Este hecho dificulta su estudio computacional, por lo que existen menos estudios y a menor Re_τ que en canales. La Figura 1.2 (b) contiene una representación esquemática del desarrollo de una capa límite.

(a)



(b)

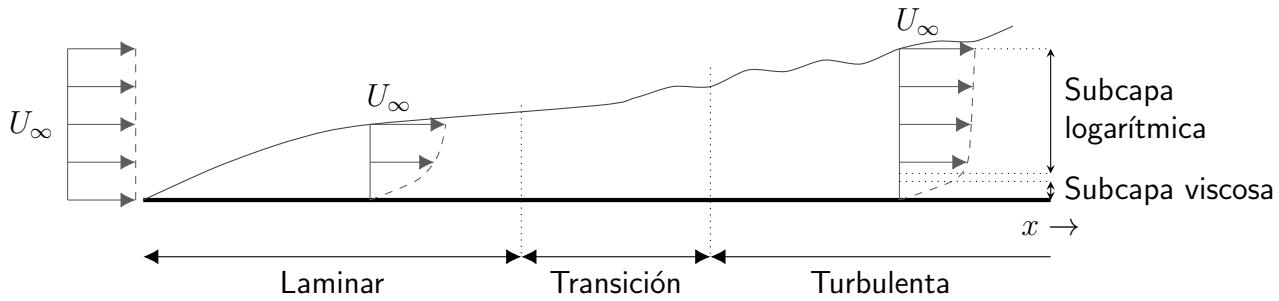


Figura 1.2: (a) Representación esquemática del flujo en un canal con gradiente de presión favorable y diferencia de velocidades entre las placas.(b) Representación esquemática del desarrollo de una capa límite.

1.3 Simulación numérica de flujos turbulentos

La simulación numérica de flujos turbulentos o mecánica de fluidos computacional (CFD) es actualmente uno de los problemas abiertos más difíciles de la física. Para la resolución de flujos turbulentos existen tres métodos principales de simulación de fluidos, cada uno con un nivel de detalle, cuya selección depende del objetivo de la simulación: RANS, LES y DNS.

1.3.1 *Reynolds Averaged Navier-Stokes*

El método RANS (Reynolds Averaged Navier-Stokes) es el menos detallado de los tres, siendo además el único de ellos que permite cálculos estacionarios. Esto es debido a que no resuelve ninguna escala de turbulencia (por definición no estacionaria), sino que las modela todas y resuelve la *geometría* del flujo mediante el uso de estos modelos. Debido a su bajo coste computacional, la gran mayoría de simulaciones CFD industriales se realizan en RANS.

Este método está basado en la descomposición del vector velocidad $\vec{U}(\vec{x},t)$ en su media $\langle\vec{U}(\vec{x})\rangle$ y su fluctuación instantánea $u(\vec{x},t)$ tal que:

$$\vec{U}(\vec{x},t) = \langle\vec{U}(\vec{x})\rangle + \vec{u}(\vec{x},t) \tag{1.10}$$

Aplicando esta descomposición a las ecuaciones de Navier-Stokes incompresibles, tal como se detalla en el cuarto capítulo de [4], se pueden obtener las *Ecuaciones de Reynolds*:

$$\frac{\overline{D}\langle U_j\rangle}{\overline{D}t} = -\frac{1}{\rho}\frac{\partial\langle P\rangle}{\partial x_j} + \nu\nabla^2\langle U_j\rangle - \frac{\partial\langle u_i u_j\rangle}{\partial x_i} \tag{1.11}$$

donde $\overline{D}\langle U_j\rangle/\overline{D}t$ representa la tasa de variación de la velocidad en un punto moviéndose según la velocidad media del fluido en ese punto.

Estas ecuaciones son análogas a las originales de Navier-Stokes en todos los términos menos el último, que aparece como consecuencia de las derivadas cruzadas. Este término, conocido como el *tensor de esfuerzos de Reynolds*, representa la transferencia de momento producida por el campo fluctuante de velocidades.

La aparición de este término introduce seis nuevas incógnitas sin introducir nuevas ecuaciones, originado el denominado *problema de cierre*. Para obtener estas seis incógnitas, $\langle u^2\rangle$, $\langle v^2\rangle$, $\langle w^2\rangle$, $\langle uv\rangle$, $\langle uw\rangle$, $\langle vw\rangle$, es necesaria la incorporación de nuevas ecuaciones basadas en modelos de turbulencia empíricos o teóricos. Estas nuevas ecuaciones resultan en que todas las escalas de turbulencia se modelan en vez de resolverse, requiriendo una menor resolución. Por lo tanto, los esquemas RANS solo resuelven los aspectos geométricos del flujo medio, incorporando cualquier inestabilidad en el modelo de turbulencia.

Una forma de *cerrar* el problema es mediante la introducción de la viscosidad turbulenta ν_t de Boussinesq, según la cual los términos del tensor de esfuerzos Reynolds son proporcionales a los esfuerzos cortantes en cada punto del fluido tal que:

$$-\rho\langle u_i u_j\rangle = \nu_t\left(\frac{\partial\langle U_i\rangle}{\partial x_j} + \frac{\partial\langle U_j\rangle}{\partial x_i}\right) - \frac{2}{3}k\delta_{ij} \tag{1.12}$$

donde k es la energía cinética turbulenta en dicho punto del fluido y δ_{ij} es la Delta de Kronecker. Los modelos que usan esta aproximación, conocidos como EVM (*Eddy Viscosity Models*) son los de uso más común en la industria por su menor coste computacional. Comúnmente, emplean una o dos ecuaciones de transporte para obtener los valores de ν_t y k en cada punto, con constantes calibradas experimentalmente o computacionalmente

mediante simulaciones LES o DNS. Como ejemplos de estos modelos, destacan los modelos Spalart-Allmaras, k-epsilon, k-omega y sus variantes [4].

Otra forma de resolver el tensor de esfuerzos de Reynolds es la usada por los modelos RSE (*Reynolds Stress Equation*), en la que se transportan cada uno de los seis componentes del tensor de esfuerzos de Reynolds de forma independiente, evitando así la viscosidad turbulenta de Boussinesq y pudiendo producir resultados más precisos. Este aumento del número de ecuaciones conlleva un mayor coste computacional respecto a los modelos EVM, aunque sea este aún muy inferior al de simulaciones LES o DNS, por lo que su uso industrial está menos extendido que los EVM.

Sin embargo, ambas formas de calcular el tensor de esfuerzos de Reynolds presentan el mismo problema: las escalas más grandes de la turbulencia no son universales sino que dependen de las características de cada problema. Esto impide a los modelos de turbulencia usados en esquemas RANS ser universales y les obliga a ser adaptados para cada caso para producir resultados fiables.

1.3.2 *Large Eddy Simulation*

En un nivel superior de complejidad se sitúa el modelo LES, en el que se resuelven directamente las escalas grandes pero se modelan las más pequeñas. Una de las formas de construir un esquema numérico para llegar ahí consiste en filtrar las ecuaciones de Navier-Stokes mediante un filtro de convolución de tipo paso-bajo, explícito o implícito, aplicado mediante la discretización espacial del problema.

El objetivo de este filtro es eliminar las frecuencias y números de onda más grandes, correspondientes a las escalas más pequeñas. El filtrado de las ecuaciones produce un término de advección no lineal en el que se introducen los modelos de las escalas más pequeñas [4].

Estos modelos son generalmente de viscosidad turbulenta, análogos a los introducidos anteriormente pero solo funcionando en las escalas más pequeñas. Debido a su mayor complejidad y a que no son usados en este trabajo, no se comentará en este trabajo detalladamente el desarrollo matemático de estos modelos.

Los modelos LES tienen un coste computacional órdenes de magnitud superiores a los RANS, por lo que su uso ingenieril es más limitado. Sin embargo, son imprescindibles en los casos en los que se buscan las dinámicas de las escalas más grandes de la turbulencia, destacando como ejemplo la meteorología.

1.3.3 *Direct Numerical Simulation*

Por último, las simulaciones de mayor complejidad computacional son las realizadas mediante DNS, que asume como única hipótesis las ecuaciones de Navier-Stokes y las resuelve en todas sus escalas, desde las integrales hasta las disipativas, sin emplear ningún mo-

delo adicional. Debido a esta resolución completa, el coste computacional es altísimo y dependiente del número de Reynolds: el número de operaciones depende de $Re_\tau^{15/4}$ [6].

Además, la precisión en materia de escalas requerida requiere el uso de mallas estructuradas con unos requisitos de posicionamiento estrictos, limitando su uso a geometrías simples. Estas dos características de las simulaciones DNS hacen que no resulten prácticas a nivel industrial, siendo su uso fundamentalmente científico.

Para reducir su coste computacional, las simulaciones DNS utilizan generalmente dominios periódicos en todas las direcciones en las que es posible en el caso a resolver, al ser los métodos basados en la transformada discreta de Fourier muy eficientes computacionalmente. Si resuelven las ecuaciones de Navier-Stokes completamente en el dominio de Fourier, estos métodos son conocidos como espectrales y si usan transformaciones a dominio físico para ciertos términos, pseudo-espectrales [4].

El número de dimensiones periódicas usadas divide en dos grandes grupos las simulaciones DNS: por un lado, la turbulencia homogénea, en la que las tres direcciones espaciales se pueden asumir periódicas, y por otro, la tubulencia inhomogénea, en la que una o más direcciones presentan anisotropía y por lo tanto no se pueden asumir como periódicas.

Dentro de la turbulencia inhomogénea destacan los flujos con esfuerzos cortantes como los chorros y los flujos de pared. En estos casos, dos direcciones se consideran periódicas para aprovechar la eficiencia de los métodos de Fourier y otro método, como por ejemplo polinomios de Chebyshev o diferencias finitas compactas, es usado en la dirección anisotrópica.

En este trabajo se ha partido de simulaciones DNS al ser éstas las que más información proporcionan de todas las escalas turbulentas, y el esquema numérico de DNS usado está descrito en detalle en §2.1. Sin embargo, el algoritmo y programa desarrollados podrían ser también aplicados a simulaciones LES en mallas estructuradas prismáticas.

1.4 Estado del arte en simulaciones DNS

La evolución de la informática en la segunda mitad del siglo XX, que implicó una tendencia de aumento de capacidad de cálculo en tres órdenes de magnitud cada 15 años [6], ha permitido una evolución paralela del tamaño de los casos resueltos mediante simulación numérica directa. Las primeras simulaciones directas como la de Kim *et al* [7], que supusieron un gran esfuerzo computacional en el momento de su realización, ahora pueden ser realizadas en un equipo personal en tiempos del orden de horas.

En el momento de la redacción de este trabajo, las simulaciones de flujos turbulentos en canales alcanzan un número de Reynolds de fricción de $Re_\tau = 10000$, en una simulación con $8 \cdot 10^{10}$ puntos y una base de datos generada de 75TB [8]. Estos casos ya sitúan el número de Reynolds simulado en el mismo orden de magnitud que los estudios experimentales en laboratorio de flujos en tuberías [5], pero aún quedan lejos de los estudios de turbulencia atmosférica como el de Hutchins *et al* [9]. La principal ventaja de los estudios

DNS frente a los experimentales es que permiten la extracción de datos mucho más completos debido a las mallas que necesitan para su correcta resolución, causa de su altísimo coste computacional.

Esta gran cantidad de datos generada por las simulaciones DNS, que puede llegar al orden de Petabytes [6], supone un además un reto añadido debido a las necesidades de almacenaje y mantenimiento y a la dificultad de transferencia de datos entre grupos de investigación.

En el otro extremo de las simulaciones, el avance de la potencia de cálculo también ha abierto la posibilidad de realizar simulaciones DNS a bajo número de Reynolds en cuestión de minutos, permitiendo así la aplicación de herramientas computacionalmente caras, como los análisis Monte Carlo . Estas herramientas pueden ser útiles en el avance de la investigación científica en turbulencia al reducir el impacto de los sesgos del equipo investigador en la selección de los casos de estudio [6].

1.5 Estructuras coherentes en flujos turbulentos

El rápido desarrollo de las técnicas experimentales a partir de la década de 1960 desencadenó el avance de un nuevo paradigma para explicar la cascada de energía turbulenta. Experimentos de visualización de flujo como los de Kline *et al* [10] empezaron a explicar la turbulencia desde un punto de vista físico, torbellino por torbellino, en vez de un punto de vista estadístico agregado a todo el flujo. Estos experimentos identificaron regiones coherentes espacial y temporalmente, y cómo la cinemática de estas estructuras participa en la producción y transporte de energía turbulenta.

Posteriormente, los avances tecnológicos en el área de la computación han permitido el desarrollo de simulaciones DNS, por el elevado coste computacional que presentan. Estas simulaciones, al contrario que los métodos experimentales, permiten obtener información completa de todo el campo de velocidades en el dominio resuelto. De esta forma, los resultados de estas simulaciones pueden ser usados para identificar regiones vorticiales mediante métodos más sofisticados que los que pueden ser usados en medidas experimentales. Esta mayor resolución que proporcionan las simulaciones DNS ha permitido el desarrollo de modelos que describen la turbulencia a partir de sus componentes, especialmente los denominados *hairpins*, vórtices en forma de U cuyos extremos nacen en la pared [11].

En esta década, destacan los estudios de dinámica vorticial de Cardesa *et al* [12] en turbulencia isótropa y Lozano-Durán y Jiménez [13] en turbulencia de pared, en los que se estudia el movimiento de las estructuras turbulentas a partir de los resultados de simulaciones DNS.

En este último artículo, Lozano-Durán y Jiménez han obtenido que la mayoría de los vórtices que nacen en flujos en canales mantienen un tamaño pequeño durante sus vidas, en su mayoría cortas. Sin embargo, algunos de los torbellinos que se crean crecen hasta llegar desde la pared hasta la subcapa logarítmica. Estos últimos son autosemejantes

espacial y temporalmente y su tiempo de vida es proporcional a su tamaño y distancia a la pared.

1.6 Objetivo del trabajo

El objetivo de este trabajo es desarrollar un programa que identifique, agregue y caracterice las estructuras coherentes presentes en un canal turbulento en todo su rango de escalas y estudie su evolución temporal a partir de resultados de simulaciones DNS.

Este trabajo parte ya de un código DNS funcional, verificado y validado, de los resultados del cual se puede recuperar información del gradiente de velocidades en cada punto del dominio fluido. A partir de éstos, los objetivos propuestos para el presente trabajo son:

- **Desarrollo conceptual de un algoritmo** de fácil paralelización que identifique estructuras coherentes y estudie su evolución a través del tiempo.
- **Programación del algoritmo en paralelo** en Fortran con OpenMP, probado y documentado para facilitar su uso en el futuro para el desarrollo de otros estudios.
- **Desarrollo de códigos auxiliares** en MATLAB, incluyendo herramientas de estudio de los campos fluidos, de calibración de las constantes del código en Fortran y de visualización de archivos de salida.
- **Extracción de una serie de resultados** para representar diversos aspectos de la dinámica de los vórtices en simulaciones a bajo número de Reynolds.

Por último, se pretende también que el código desarrollado sea la base de estudio de los fenómenos físicos que ocurren en las estructuras coherentes turbulentas en flujos de pared y capas límite en el futuro.

Capítulo 2

Simulación mediante DNS

El uso de simulaciones DNS presenta problemas, en especial el coste computacional de las simulaciones y la consiguiente limitación en el número de Reynolds. Sin embargo, es el método disponible actualmente que más información proporciona sobre el campo fluido, al resolver directamente las ecuaciones de Navier-Stokes sin utilizar ningún modelo adicional y no presentar los problemas de medición presentes en experimentos.

Por lo tanto, en este trabajo se han empleado simulaciones DNS de flujo en canales, siendo el dominio simulado un canal en el que se consideran las dimensiones paralelas a la pared infinitas, implementadas como periódicas en el método numérico permitiendo así el uso de métodos basados en la transformada discreta de Fourier, aprovechando sus ventajas en eficiencia computacional. En la dirección perpendicular a la pared el dominio es finito, y se imponen condiciones de contorno de no deslizamiento en las paredes.

Debido a que las simulaciones DNS de flujos movidos por un gradiente de presión necesitan dominios más pequeños que los flujos con diferencia de velocidades para producir resultados físicamente relevantes [14], en este trabajo todos las simulaciones empleadas han sido de flujos de Poiseuille.

El código DNS usado en este trabajo ha sido empleado y validado en diversos trabajos [15][16][17] por lo que no se considera necesaria la validación de sus resultados en el presente trabajo.

2.1 Método de simulación DNS

El esquema numérico empleado, introducido por Kim, Moin y Moser [7], parte de las ecuaciones de Navier-Stokes y de continuidad para un flujo incompresible:

$$\nabla \cdot \vec{U} = 0 \tag{2.1}$$

$$\frac{\partial \vec{U}}{\partial t} + \vec{U} \cdot \nabla \vec{U} = -\frac{1}{\rho} \nabla P + \nu \nabla^2 \vec{U} \tag{2.2}$$

Para la resolución numérica más eficiente de estas ecuaciones, el esquema numérico las transforma en un sistema de dos ecuaciones: una para el laplaciano de la velocidad normal a la pared $\phi = \nabla^2 v$ y otra para el componente de la vorticidad perpendicular a la pared ω_y. Para ello, se introduce el parámetro $\vec{H}$, la *helicidad*, igual al producto vectorial de la velocidad $\vec{U}$ y la vorticidad $\vec{\Omega}$:

$$\vec{H} = \vec{U} \times \vec{\Omega} \tag{2.3}$$

Aplicando la definición de *helicidad* (Ecuación 2.3) al término convectivo $(\vec{U} \cdot \nabla \vec{U})$ y adimensionalizando (sin cambiar los símbolos) la Ecuación 2.2 se obtiene:

$$\frac{\partial \vec{U}}{\partial t} + \frac{1}{2}\nabla(\vec{U} \cdot \vec{U}) - \vec{H} = -\frac{1}{\rho}\nabla P + \frac{1}{Re_b}\nabla^2 \vec{U} \tag{2.4}$$

donde Re_b es el número de Reynolds relativo a la velocidad media del canal u_b: $Re_b = u_b h/\nu$. La velocidad media del canal viene definida como:

$$u_b = \frac{1}{2h}\int_{-h}^{h} \langle \vec{U} \rangle \, \mathrm{d}y \tag{2.5}$$

El siguiente paso para obtener las ecuaciones ϕ y ω_y consiste en obtener la divergencia de la Ecuación 2.4 para formular las ecuaciones de transporte de la divergencia de la helicidad $\nabla \cdot \vec{H}$, aplicando la propiedad conmutativa de la divergencia con laplacianos y con derivadas temporales:

$$\frac{\partial(\nabla \cdot \vec{U})}{\partial t} + \frac{1}{2}\nabla^2(\vec{U} \cdot \vec{U}) - \nabla \cdot \vec{H} = -\frac{1}{\rho}\nabla^2 P + \frac{1}{Re_b}\nabla^2(\nabla \cdot \vec{U}) \tag{2.6}$$

que puede ser fácilmente simplificada aplicando continuidad, $\nabla \cdot \vec{U} = 0$:

$$\nabla \cdot H = \frac{1}{2}\nabla^2(\vec{U} \cdot \vec{U}) + \frac{1}{\rho}\nabla^2 P \tag{2.7}$$

El laplaciano de la Ecuación 2.4 viene dado por:

$$\frac{\partial(\nabla^2 \vec{U})}{\partial t} + \nabla\left(\frac{1}{2}\nabla^2(\vec{U} \cdot \vec{U})\right) - \nabla^2 H = -\nabla\left(\frac{1}{\rho}\nabla^2 P\right) + \frac{1}{Re_b}\nabla^4 \vec{U} \tag{2.8}$$

incluyendo los términos de la derecha de la Ecuación 2.7 queda:

$$\frac{\partial(\nabla^2 \vec{U})}{\partial t} + \nabla\left(\nabla \cdot H\right) - \nabla^2 H = +\frac{1}{Re_b}\nabla^4 \vec{U} \tag{2.9}$$

Y el rotacional de la Ecuación 2.4 viene dado por:

$$\frac{\partial \vec{\Omega}}{\partial t} + \frac{1}{2}\nabla \times \nabla(\vec{U} \cdot \vec{U}) - \nabla \times \vec{H} = -\frac{1}{\rho}\nabla \times \nabla P + \frac{1}{Re_b}\nabla^2 \vec{\Omega} \tag{2.10}$$

sabiendo que el rotacional de un gradiente es siempre nulo:

$$\frac{\partial \vec{\Omega}}{\partial t} - \nabla \times \vec{H} = \frac{1}{Re_b}\nabla^2 \vec{\Omega} \tag{2.11}$$

Las Ecuaciones 2.9 y 2.11 son ecuaciones vectoriales, pero solo los términos en la dirección perpendicular a la pared son necesarios para resolver el problema:

$$\frac{\partial \phi}{\partial t} = h_v + \frac{1}{Re_b}\nabla^2 \phi \tag{2.12}$$

$$\frac{\partial \omega_y}{\partial t} = h_g + \frac{1}{Re_b}\nabla^2 \omega_y \tag{2.13}$$

donde:

$$h_v = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right) H_2 - \frac{\partial}{\partial y}\left(\frac{\partial H_1}{\partial x} + \frac{\partial H_3}{\partial z}\right) \tag{2.14}$$

$$h_g = \frac{\partial H_1}{\partial z} - \frac{\partial H_3}{\partial x} \tag{2.15}$$

siendo H_1, H_2 y H_3 los tres componentes del vector helicidad introducido en la Ecuación (2.3). Las ecuaciones restantes para cerrar el problema son:

$$\nabla^2 v = \phi \tag{2.16}$$

$$\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} = -\frac{\partial v}{\partial y} \tag{2.17}$$

$$\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} = \omega_y \tag{2.18}$$

$$\frac{\partial \omega_x}{\partial x} + \frac{\partial \omega_z}{\partial z} = -\frac{\partial \omega_y}{\partial y} \tag{2.19}$$

$$\frac{\partial \omega_z}{\partial x} - \frac{\partial \omega_x}{\partial z} = \phi \tag{2.20}$$

donde $\vec{U} = (u, v, w)$ y $\vec{\Omega} = (\omega_x, \omega_y, \omega_z)$. A este sistema de ecuaciones se le imponen como condiciones de contorno la condición de no deslizamiento $v(\pm 1) = 0$, la diferencia de velocidad en las paredes $u(+1)$ y el gradiente de presión en la dirección del flujo $\partial P/\partial x$. Esta transformación de las ecuaciones de Navier-Stokes presenta dos ventajas que solo están presentes mediante la selección de estas dos variables:

- No es necesario resolver la presión para resolver las ecuaciones
- Solo dos ecuaciones de transporte son necesarias en vez de una para cada componente de la velocidad.

Para la resolución numérica el método empleado utiliza series de Fourier de-aliaseadas en las direcciones paralelas a la pared, y diferencias finitas compactas de siete puntos con resolución quasiespectral [18] en la dirección y. Este método presenta ventajas en precisión y robustez frente a los métodos de diferencias finitas habituales, manteniendo la posibilidad de posicionar los puntos en la dirección perpendicular a la pared de forma arbitratia frente a otros métodos que imponen sus posiciones, como los basados en polinomios de Chebyshev. La discretización temporal usada es un método Runge-Kutta semi-implicito de tercer orden. Este código de simulación DNS está desarrollado para funcionar de forma eficiente en clusters de computadores, y por lo tanto está programado usando un esquema paralelo MPI.

Al ser un canal, los resultados con significado físico se obtienen una vez el sistema ha llegado a un estado estadísticamente estacionario, y todos los resultados deben ser extraídos una vez la simulación haya alcanzado este estado.

2.2 Criterios de identificación de vórtices

Pese a estar aceptado universalmente el vórtice como unidad definitoria de la turbulencia, no se ha aceptado una definición rigurosa de lo que delimita una región espacial constitutiva de un vórtice.

Numerosos métodos para la identificación de vórtices se han desarrollado en conjunto con la evolución en la tendencia del estudio de la turbulencia, existiendo tanto criterios para identificar estructuras coherentes en experimentos (lagrangianos) como en resultados de simulaciones DNS (eulerianos).

Los criterios eulerianos locales, usados típicamente en simulaciones DNS, utilizan la información de la velocidad y sus derivadas espaciales en cada instante del tiempo para distinguir qué puntos del dominio podrían formar parte de estas estructuras coherentes. Estos criterios pueden ser tan sencillos como la comparación de la presión o el valor absoluto de la vorticidad ($|\nabla \times \vec{U}|$) con un valor límite establecido para cada caso, o tan complejos como el propuesto por Jeong *et al* [19], que además permite obtener los núcleos de los vórtices que identifica. Cabe destacar que, al tener que aplicar la función definida por el criterio sobre todos los puntos del dominio, el incremento de complejidad de ésta conlleva un incremento no despreciable en el coste computacional que supone.

Todos los criterios bien planteados y con un valor límite bien seleccionado, independientemente de su complejidad, identifican regiones similares dentro del dominio fluido, con discrepancias dependientes de las particularidades del criterio escogido.

Los que se han considerado para el desarrollo de este trabajo basan su definición en la matriz jacobiana del vector velocidad, para transformarla posteriormente a un valor booleano en cada punto:

$$\begin{array}{rccl} f: & M_{3\times3}(\mathbb{R}) & \rightarrow & \mathbb{B} \\ & \mathbf{J}[\vec{U}] & \mapsto & b = f(\mathbf{J}[\vec{U}]) \end{array} \tag{2.21}$$

donde $M_{3\times3}(\mathbb{R})$ es el conjunto de todas las matrices de números reales de dimensión 3×3, $\mathbf{J}[\ ]$ el operador jacobiano y $\mathbb{B}$ el conjunto formado por verdadero y falso, en el que está definida el álgebra booleana.

Este proceso se hacen en dos pasos: primero se obtiene un valor del criterio de vorticidad para cada punto realizando

$$\begin{array}{rccl} g: & M_{3\times3}(\mathbb{R}) & \rightarrow & \mathbb{R} \\ & \mathbf{J}[\vec{U}] & \mapsto & a = g(\mathbf{J}[\vec{U}]) \end{array} \tag{2.22}$$

donde $\mathbb{R}$ es el conjunto de todos los números reales. El valor obtenido es luego compararlo con un valor límite previamente definido obteniendo así el valor booleano final en cada

punto del campo:

$$\begin{array}{rccc} h: & \mathbb{R} & \rightarrow & \mathbb{B} \\ & a & \mapsto & b = h(a) \end{array} \tag{2.23}$$

La función criterio de vorticidad f queda definida como la composición $h \circ g$. En este trabajo se han considerado tres criterios diferentes que serán introducidos a continuación, todos ellos partiendo de la descomposición de la matriz jacobiana del vector velocidad en sus partes simétrica y antisimétrica [20]:

$$\mathbf{J}[\vec{U}] = \mathbf{G} = \mathbf{S} + \mathbf{\Omega} \tag{2.24}$$

$$\mathbf{S} = \frac{1}{2}\left[\mathbf{G} + \mathbf{G}^T\right] \tag{2.25}$$

$$\mathbf{\Omega} = \frac{1}{2}\left[\mathbf{G} - \mathbf{G}^T\right] \tag{2.26}$$

donde:

$$G_{ij} = \frac{\partial U_i}{\partial x_j} \tag{2.27}$$

$$S_{ij} = \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) \tag{2.28}$$

$$\Omega_{ij} = \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} - \frac{\partial U_j}{\partial x_i}\right) \tag{2.29}$$

La parte simétrica de la matriz jacobiana, $\mathbf{S}$, representa la tasa de deformaciones en cada punto del fluido, y la antisimétrica, $\mathbf{\Omega}$, su tasa de rotación.

De entre todos los criterios disponibles en la literatura, en este trabajo se han considerado los criterios de Hunt, de Chong con límite homogéneo y de Chong con límite no homogéneo y por tanto solamente estos están explicados en detalle a continuación.

2.2.1 Criterio de Hunt

Este criterio, introducido por Hunt, Wray y Moin en [21], considera como pertenecientes a un vórtice los puntos en los que se cumplan dos condiciones:

1. Los esfuerzos de rotación predominen sobre las deformaciones irrotacionales
2. Exista un mínimo en la presión en la zona definida como vórtice

El primer punto puede ser descrito fácilmente mediante las matrices introducidas previamente:

$$Q = \frac{1}{2}\left(||\mathbf{\Omega}||^2 - ||\mathbf{S}||^2\right) \tag{2.30}$$

donde el operador $||\cdot||$ representa la norma de Frobenius de una matriz $||\mathbf{A}|| = (\sum_i \sum_j A_{ij}^2)^{\frac{1}{2}}$. Para identificar las regiones vorticiales, este valor de Q es comparado con un valor límite Q_{thresh}. El valor B, representativo de si el punto evaluado cumple o no la primera condición del criterio, puede ser expresado como:

$$B = (Q > Q_{thresh}) \tag{2.31}$$

Cuando presentaron este criterio, Hunt *et al* [21] escogieron como valor límite la media cuadrática del valor de Q en todo el dominio:

$$Q_{thresh} = \sqrt{\overline{Q^2}} \tag{2.32}$$

Existen otras definiciones para el criterio, como la definida por Dubief y Delcayre [22], que busca maximizar la media temporal de las fluctuaciones de vorticidad en las regiones vorticiales.

Respecto a la segunda condición, como se ha descrito anteriormente, el esquema numérico DNS utilizado no requiere información del campo de presiones y por tanto éste habría de ser calculado y posteriormente derivado para extraer sus extremos. Esto produciría un incremento considerable en el coste computacional respecto a otros criterios solamente basados en una función del gradiente de velocidades.

2.2.2 Criterio de Chong

Chong y Perry [23] consideran un sistema de referencia móvil en cada punto a la velocidad del fluido para aplicar en cada uno de ellos análisis de puntos críticos (también conocido como *teoría de espacio de fases*) [20].

Considerando cada punto resuelto como un punto crítico, la trayectoria de una partícula fluida en sus proximidades con vector posición $\vec{y}(t)$ puede aproximarse mediante el primer término de una expansión de Taylor, dado por la matriz jacobiana:

$$\frac{\mathrm{d}\vec{y}}{\mathrm{d}t} = \mathbf{G}\vec{y} \tag{2.33}$$

Partiendo de una condición inicial $\vec{y}(0) = \vec{y}_0$, este es un sistema de ecuaciones diferenciales lineales con solución analítica mediante la descomposición en autovalores y autovectores de la matriz $\mathbf{G}$:

$$\vec{y}(t) = \vec{\chi}_1 e^{\lambda_1 t} c_1 + \vec{\chi}_2 e^{\lambda_2 t} c_2 + \vec{\chi}_3 e^{\lambda_3 t} c_3 \tag{2.34}$$

donde $\vec{\chi}_i$ son los autovectores de la matriz $\mathbf{G}$, λ_i sus autovalores y c_i constantes dependientes de $\vec{y}_0$.

Si los tres autovalores son reales, el punto se considera una *fuente*, *sumidero* o *punto de remanso*. Sin embargo, si entre los autovalores se encuentra una pareja de conjugados complejos, el punto se considera un *foco* e indica que las trayectorias a su alrededor son cerradas o en espiral, indicando que el punto analizado pertenece a un vórtice.

Los autovalores de $\mathbf{G}$ se obtienen resolviendo la ecuación característica de la matriz:

$$\det\left(\mathbf{G} - \lambda\mathbf{I}\right) = 0 \tag{2.35}$$

$$\lambda^3 - \mathrm{tr}\left(\mathbf{G}\right)\lambda^2 + \frac{1}{2}\left(\mathrm{tr}\left(\mathbf{G}\right)^2 - \mathrm{tr}\left(\mathbf{G}^2\right)\right)\lambda - \det\left(\mathbf{G}\right) = 0 \tag{2.36}$$

donde el operador $\mathrm{tr}\left(\cdot\right)$ representa la traza de una matriz, la suma de los términos de su diagonal principal. Al ser éste un caso incompresible, aplicando la Ecuación 2.1:

$$\mathrm{tr}\left(\mathbf{G}\right) = \nabla \cdot \vec{U} = 0 \tag{2.37}$$

lo que lleva a la simplificación de la Ecuación 2.36 hasta:

$$\lambda^3 + \frac{1}{2}\left(||\mathbf{\Omega}||^2 - ||\mathbf{S}||^2\right)\lambda - \det\left(\mathbf{G}\right) = 0 \tag{2.38}$$

Siguiendo el método de Cardano para la resolución de ecuaciones cúbicas, se llega a que no es necesario terminar la resolución para obtener si los autovalores son reales o complejos, basta con conocer el valor del discriminante Δ:

$$\Delta = \frac{27}{4}\det\left(\mathbf{G}\right)^2 + \left(\frac{||\mathbf{\Omega}||^2 - ||\mathbf{S}||^2}{2}\right)^3 \tag{2.39}$$

Por lo tanto, el criterio de Chong define como puntos pertenecientes a un vórtice todos aquellos en los que se cumpla que:

$$B = \Delta > \Delta_{thresh} \tag{2.40}$$

Según Chong y Perry en el momento de su propuesta, $\Delta_{thresh} = 0$, ya que a partir de este valor se cumple estrictamente que los autovalores tengan parte imaginaria [23]. A efectos prácticos, para conseguir resultados relevantes físicamente, Δ_{thresh} tiene que adoptar valores más altos [20].

2.2.3 Criterio de Chong de límite no homogéneo

Sin embargo, el criterio de Chong con límite homogéneo, aplicado a turbulencia de pared, presenta un inconveniente: para un valor constante de Δ_{thresh} en todo el dominio, el número de puntos que identifica como vórtices depende fuertemente de la distancia a cada una de las paredes [24]. Esto provoca que, al seleccionar un valor de Δ_{thresh} que permita estudiar correctamente la turbulencia en el centro del canal, las regiones más próximas a las paredes estarán abarrotadas de puntos que cumplan el criterio, y si se selecciona un valor más pequeño para estudiar las estructuras cercanas a la pared con mayor claridad, en la parte central del canal no se apreciarán apenas estructuras.

Como solución a este problema, Del Álamo *et al* [24] proponen escalar el criterio en cada plano del dominio con la desviación estándar del valor del discriminante en cada plano, modificando la Ecuación (2.40):

$$B = \Delta > \alpha \cdot \left(\overline{(\Delta')^2}\right)^{1/2}\ (y) \tag{2.41}$$

Donde el parámetro α resultaría análogo a los valores límite expuestos en los otros criterios. Los valores de la desviación estándar del discriminante adimensionalizados obtenidos por Del Álamo *et al* en [24] para valores de Re_τ entre 180 y 1900 están presentes en la Figura 2.1.

Al igual que con los otros criterios, un valor excesivamente pequeño causa que los vórtices se condensen en unos pocos clúster de gran tamaño, y un valor excesivamente grande provoca que solo una serie de estructuras coherentes pequeñas y aisladas puedan ser identificadas. Del Álamo *et al*, en sus estudios, usan un valor de $\alpha = 0{,}02$ [24].

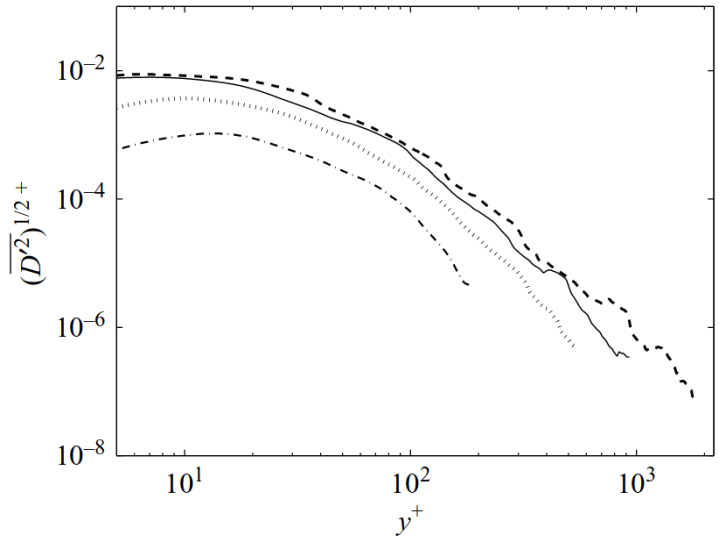


Figura 2.1: Valores de la desviación estándar del discriminante en función de la distancia a la pared, ambas adimensionalizadas, para distintos valores de Re_τ: $(-\cdot-)Re_\tau = 180,\ (\cdots)Re_\tau = 550,\ (-)Re_\tau = 950,\ (--)Re_\tau = 1900$. Extraído de [24].

2.3 Simulaciones empleadas

El algoritmo de agregado desarrollado en este trabajo se ha lanzado con los resultados de una simulación DNS de flujo en canal movido por un gradiente de presiones. Para minimizar el tiempo transcurrido de escritura en disco, solo se ha guardado de cada paso temporal la matriz booleana comprimida que contiene un valor verdadero en los puntos que cumplen el criterio de vorticidad establecido y falso en los que no. Las características geométricas y fluidas de las simulaciones empleadas están presentes en la Tabla 2.1.

2.3.1 Criterio de vorticidad y estudio de percolación

De los tres criterios presentados anteriormente, se ha optado por usar el de Chong con límite no homogéneo introducido por Del Álamo en [24], pero el programa implementado permite el uso de cualquiera de los tres criterios.

Se ha realizado un estudio de percolación para obtener el valor del parámetro α (Ecuación 2.41) que permita la agrupación de clústers de vórtices más representativa desde el punto de vista de la física del problema. Como primer paso, partiendo de una simulación ya estadísticamente estacionaria, se ha calculado el valor del discriminante Δ (Ecuación 2.40) para todos los puntos de 100 campos, separados 10 pasos temporales entre cada uno de éstos.

Una vez obtenido el valor del discriminante, se ha obtenido su media y desviación estándar en función de la distancia a la pared, para posteriormente usar la información de la

Características Geométricas		
Número de puntos en x	N_x	192
Número de puntos en y	N_y	251
Número de puntos en z	N_z	192
Longitud del dominio en x	L_x	2π
Longitud del dominio en z	L_z	π
Altura del canal		2

Características del fluido		
Número de Reynolds	Re	2500
Número de Reynolds de fricción	Re_τ	180
Viscosidad cinemática	ν	0.000308

Características numéricas		
Número CFL		0.9
Paso temporal	Δt	0.112

Tabla 2.1: Características principales de las simulaciones numéricas empleadas.

desviación estándar para aplicar el criterio de vorticidad.

Aplicando el criterio para los distintos campos obtenidos anteriormente, se ha obtenido la probabilidad de que un punto del dominio fluido cumpla el criterio en función de su distancia a la pared, para varios valores del parámetro α. Esta distribución de probabilidad está representada en la Figura 2.2.

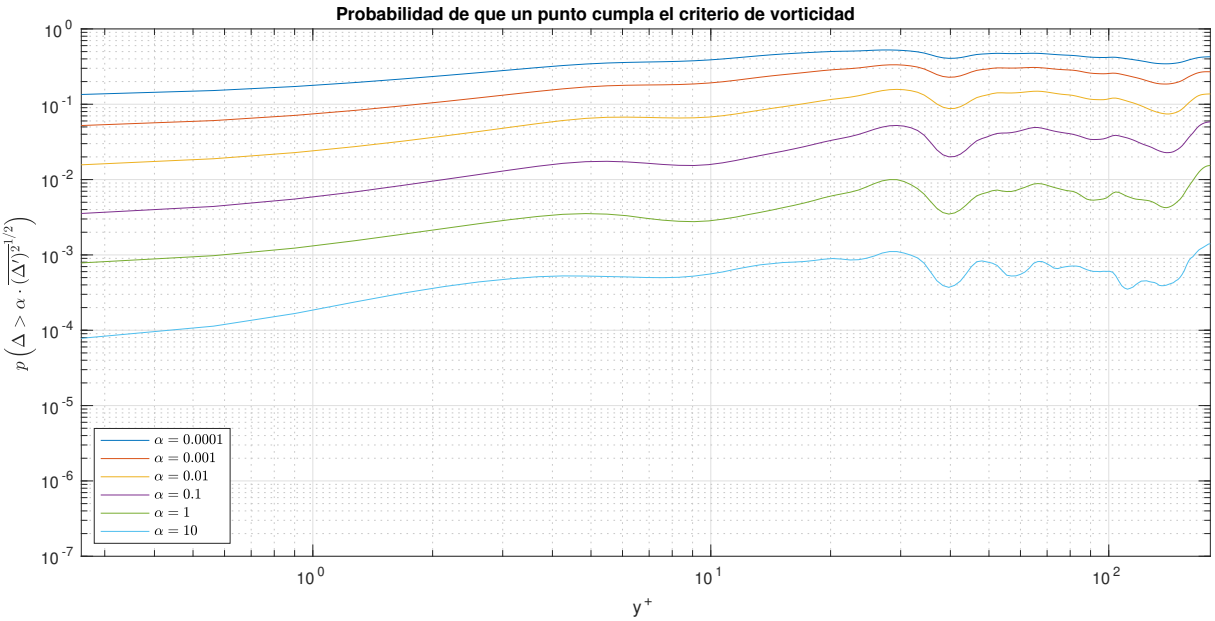


Figura 2.2: Probabilidad de que un punto cumpla el criterio de vorticidad para distintos valores del parámetro α, en función de su distancia a la pared.

Como se puede apreciar en la Figura 2.2, el rango de probabilidad aumenta en función del parámetro α seleccionado, cuanto más restrictivo es el criterio, más variabilidad hay entre los planos de mayor y menor vorticidad. Como comparación, la desviación estándar en un canal similar a mismo número de Reynolds en [24], presentada en la Figura 2.1, presenta un factor de $2 \cdot 10^2$ entre sus valores máximo y mínimo. Una vez aplicada la corrección, este factor entre máximo y mínimo se sitúa 13,5 para el valor del parámetro α con mayor variabilidad de los presentados en la Figura 2.2.

Una vez determinada la distribución de puntos pertenecientes a un vórtice en función de su distancia a la pared, se ha procedido a la comparación visual de las estructuras en función del parámetro α seleccionado. Para ello, se han representado isosuperficies del criterio booleano para tres valores de α, presentes en la Figura 2.3.

De los tres casos representados en la Figura 2.3, se puede apreciar en sus diferencias la importancia de ajustar correctamente el criterio:

- Para criterios muy permisivos (valores bajos de α), el campo resultante toma forma de *esponja*, al fusionarse todos los hilos de vorticidad en una gran estructura coherente que ocupa la mayor parte del dominio, con estructuras pequeñas en los huecos que deja la estructura más grande. Esta situación es visible en el caso de arriba de la Figura 2.3, con $\alpha = 0{,}0001$.

- Para criterios muy estrictos (valores altos de α), solo los puntos del dominio más turbulentos, como los hilos centrales de los torbellinos, estarán catalogados como vórtices. Por lo tanto, solo se podrá estudiar una pequeña parte del dominio, perdiendo parte de la información disponible. Esta situación es visible en el caso de abajo de la Figura 2.3, con $\alpha = 1$.

- Entre los dos casos presentados anteriormente, existe un rango de valores del parámetro límite en los que no sucede ninguno de los problemas descritos para los casos descritos. Dentro de este rango, el criterio es lo suficientemente permisivo para permitir que las estructuras coherentes crezcan más allá de los núcleos de torbellinos, sin juntarse entre ellas para formar estructuras de tipo *esponja*. Un ejemplo de este rango es el caso central de la Figura 2.3, en el que $\alpha = 0{,}01$.

Es en este último rango donde se puede estudiar de manera más efectiva la dinámica de las estructuras coherentes, utilizando Del Álamo *et al* un valor de $\alpha = 0{,}02$ para sus estudios de paquetes de vórtices *hairpin* autosemejantes [24]. Al estar dentro del rango considerado como aconsejable para los campos calculados, en este trabajo se ha elegido también $\alpha = 0{,}02$.

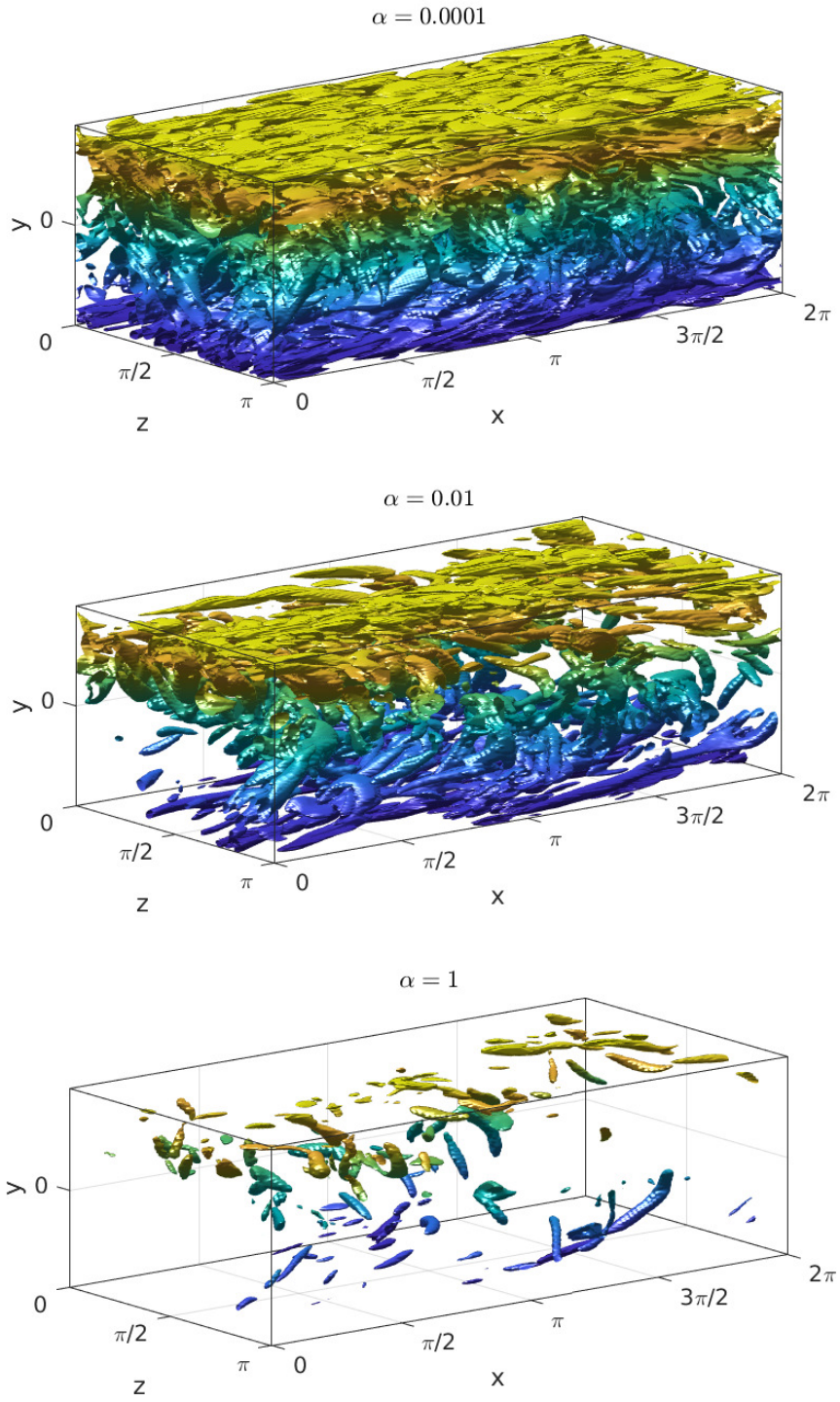


Figura 2.3: Isosuperficies del criterio de vorticidad de Chong no homogéneo para tres valores del parámetro α

Capítulo 3

Identificación y seguimiento de estructuras coherentes

Una estructura coherente se puede describir como un conjunto de puntos en los que se cumple un criterio de vorticidad establecido y que forma una región conectada del espacio tal que existe al menos un camino que conecta cada una de todas sus posibles parejas de puntos recorriendo solamente otros puntos de la estructura y avanzando solo entre puntos contiguos en cualquiera de las 3 direcciones cartesianas rectas, en cualquiera de los dos sentidos de cada una de éstas. Dicho de otra forma, todo punto perteneciente a una estructura tiene que contar con al menos un punto vecino en cualquiera de las direcciones rectas cartesianas perteneciente a esa misma estructura.

Para identificar estas estructuras vorticiales, también denominadas de forma intercambiable en este trabajo como vórtices (o clústers de vórtices) o torbellinos, se ha ideado e implementado un algoritmo que será descrito a lo largo de este capítulo.

3.1 Motivación del algoritmo desarrollado

Seguir los torbellinos en un flujo turbulento constituye un problema que puede ser dividido en dos partes principales. En primer lugar, estas estructuras coherentes deben ser identificadas y clasificadas para cada paso temporal, para poder, en segundo lugar, compararlas con las obtenidas previamente y extraer información sobre su evolución.

La primera parte, consistente en agregar las estructuras vorticiales coherentes ha sido tradicionalmente [12][13] realizada mediante la expansión en de un punto inicial en las direcciones a expandir. Sin embargo, este proceso presenta un problema desde el punto de vista computacional.

Una forma muy común de guardar la información del dominio cuando se trabaja con mallas estructuradas es en matrices tridimensionales, correspondiendo cada una de estas dimensiones a una dimensión física. Estas matrices, una vez el programa ha sido compila-

do, se guardan en memoria de forma vectorial como una lista de elementos, dependiendo la permutación hecha del lenguaje de programación. Esta reducción dimensional resulta en que solo una de las dimensiones se mantenga contigua en la memoria, manteniéndose solo el primer índice continuo en RAM en el caso de los programas en Fortran. Un ejemplo de este proceso está ilustrado en la Figura 3.1

$$A_{codigo} = \left[\begin{bmatrix} a_{1,1,1} & a_{1,1,2} \\ a_{1,2,1} & a_{1,2,2} \end{bmatrix} \begin{bmatrix} a_{2,1,1} & a_{2,1,2} \\ a_{2,2,1} & a_{2,2,2} \end{bmatrix} \right] \xrightarrow{compilado} A_{RAM} = [a_{1,1,1}\ a_{2,1,1}\ a_{1,2,1}\ a_{2,2,1}\ a_{1,1,2}\ a_{2,1,2}\ a_{1,2,2}\ a_{2,2,2}]$$

Figura 3.1: Representación de la transformación de matriz multidimensional a vector unidimensional durante la compilación del código fuente.

Para realizar cualquier operación, el procesador carga en la memoria caché solo la parte de este vector unidimensional en la que se encuentra el índice al que va a acceder en ese momento. Si opera secuencialmente en el primer índice, el procesador no tendrá que cargar en caché entre operaciones ya que los índices son contiguos. Si es en el segundo índice, el número de cargas en caché dependerá del tamaño de la matriz en la primera dimensión. En cambio, si itera sobre el tercer índice cada operación supondrá una nueva carga en caché en matrices del tamaño de las usadas en este trabajo, con el incremento de carga computacional que eso conlleva.

En este trabajo se ha desarrollado un procedimiento que evita iteraciones sobre el índice más externo, presentada detalladamente en §3.2. El concepto seguido es el de agregado de estructuras coherentes bidimensionales en cada plano para que estas sean los nodos de un grafo [25], cuyos componentes conexos serán representativas de las estructura tridimensionales. La agrupación de las regiones coherentes bidimensionales está representada mediante un ejemplo en la Figura 3.2 y la obtención de los componentes conexos en la Figura 3.3. Un campo con las estructuras tridimensionales agregadas está representado en la Figura 3.4

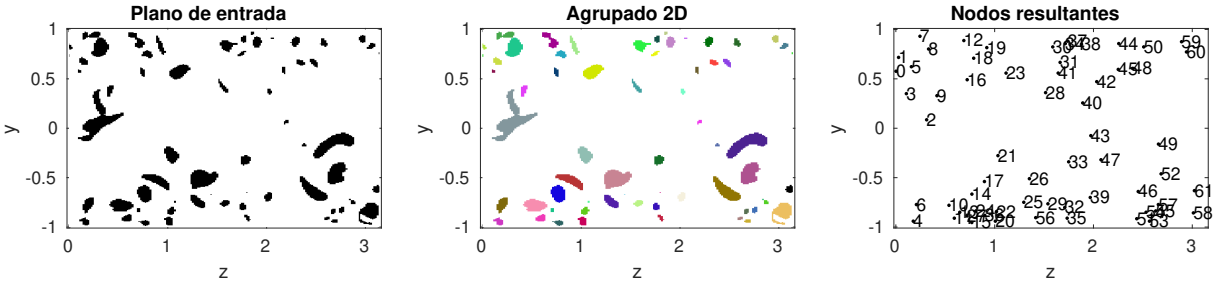


Figura 3.2: Agrupación de estructuras coherentes en un plano de entrada (izquierda), catalogación de cada estructura mediante un índice, representado por un color distinto en cada región coherente (centro) y representación de cada una de estas regiones como un nodo (derecha)

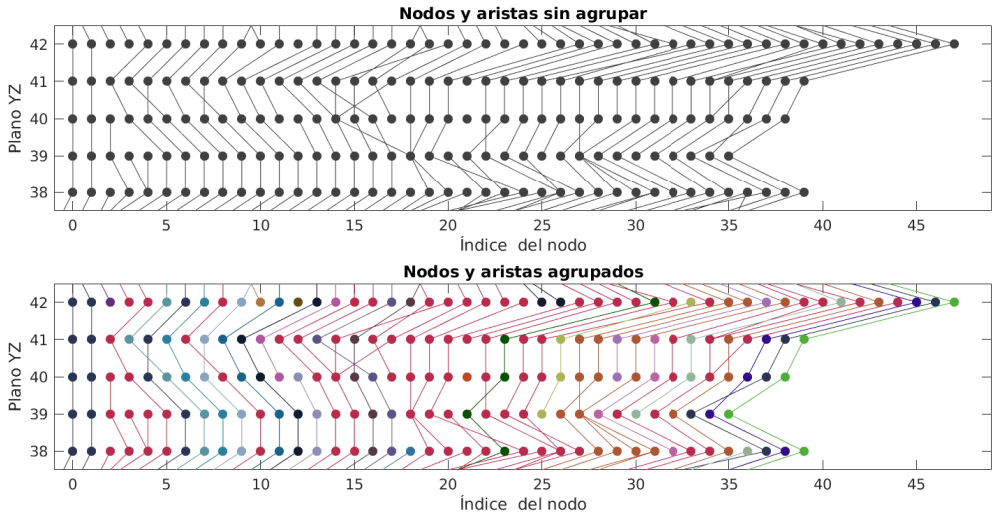


Figura 3.3: Nodos y aristas iniciales sin agrupar, coloreados según el plano al que pertenecen (arriba) y nodos y aristas agrupados, coloreados según el componente conexo al que pertenecen (abajo).

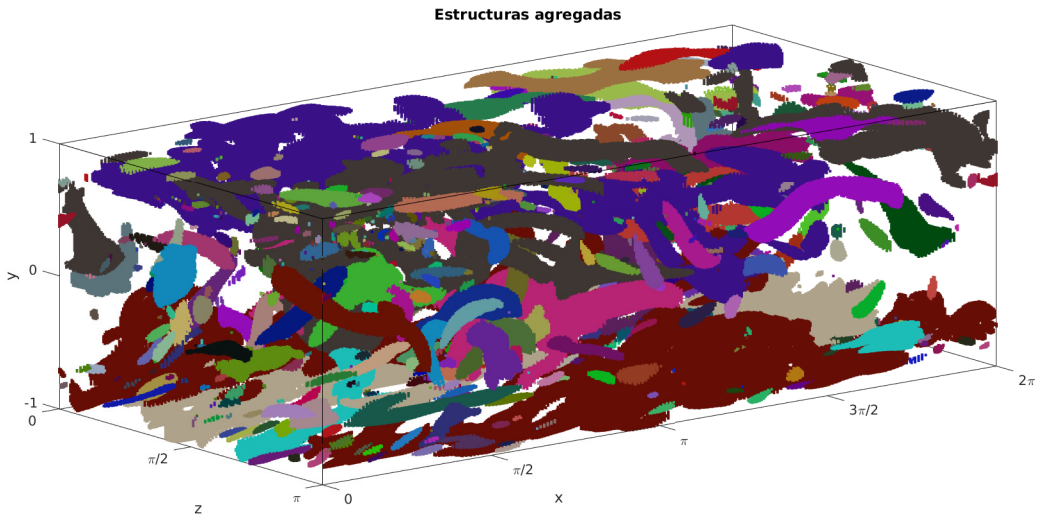


Figura 3.4: Estructuras tridimensionales en un campo fluido agregadas, con cada estructura coherente representada de un color.

En la segunda parte del problema también se ha tomado una aproximación diferente a la tradicional, consistente en calcular la intersección entre estructuras coherentes en campos separados por varios pasos temporales de distancia [12][13]. Por una parte, se ha decidido extraer información de todos los pasos temporales en los que se resuelve la simulación DNS, lo que incrementa drásticamente el número de veces a ejecutar el programa. Por otra parte, esto reduce la evolución de cada estructura en un paso temporal, permitiendo el uso de otras estrategias de comparación.

La aproximación tomada en este trabajo con tal de reducir la carga computacional es comparar una serie de características definitorias que pueden ser extraídas de los campos de entrada: centro de masas, caja envolvente y volumen. En el caso de que la comparación de las características no funcione, se recurre a una comparación punto por punto. Con la información de estas conexiones, se puede obtener la vida e interacciones de cada vórtice presente en el campo, generando una base de datos en la que se guarda esta información para todos los torbellinos encontrados. El proceso está explicado detalladamente en §3.3.

En los apartados siguientes de este capítulo se va a presentar detalladamente el algoritmo, incluyendo una serie de diagramas de flujo en pseudocódigo. Para hacerlos más fácilmente inteligibles, han sido construidos usando las funciones de Python 3, listas dinámicas e indización 0, por lo que son representativos del proceso seguido pero no del propio código fuente ni contienen información de las optimizaciones aplicadas. En §3.5 se han comentado varias de las diferencias entre el código fuente y los diagramas de flujo, además de varias de las estrategias de optimización seguidas en el código.

3.2 Algoritmo de agregado 3D

Para identificar estas estructuras de forma eficiente, se ha desarrollado un algoritmo en paralelo que, a partir de una matriz booleana que contiene cuáles de los puntos cumplen el criterio de vorticidad devuelve una lista de estructuras tridimensionales con los índices de los puntos que contiene cada una de ellas y sus características principales. El algoritmo se puede separar en tres partes:

3.2.1 Agrupación de nodos

Como primer paso para obtener las estructuras tridimensionales, el algoritmo agrupa primero paralelamente regiones vorticiales coherentes en cada plano YZ y les asigna un número identificador. Estas regiones serán los **nodos** del grafo que representa las estructuras tridimensionales.

El proceso seguido para obtener estos nodos es un algoritmo de Búsqueda en Profundidad (*Deep-First Search*) [26], y en este caso consiste en recorrer el plano punto a punto, comprobando en cada uno de ellos si cumple el criterio de vorticidad y no está asignado a otro nodo. En el caso de que esta condición se cumpla, el programa crea un nuevo nodo con el punto seleccionado como primer punto de la lista y marca el punto como asignado

a un identificador de nodos. Después, añade a esa lista los puntos que forman parte de ese nodo mediante el siguiente proceso:

1. Recorrer las dos direcciones cartesianas en sus dos sentidos desde el punto seleccionado en ese plano hasta encontrar algún punto que no cumple las condiciones de criterio o no-asignación previa, según se muestra en el diagrama de flujo presente en la Figura 3.6.

2. Añadir a la lista los puntos recorridos anteriormente, y marcarlos en la matriz como asignados al nodo.

3. Seleccionar el siguiente punto en la lista.

4. Repetir los dos pasos anteriores hasta llegar al último de la lista, donde no puede añadir ningún otro punto a la lista (en caso de poder, por definición, no sería el último).

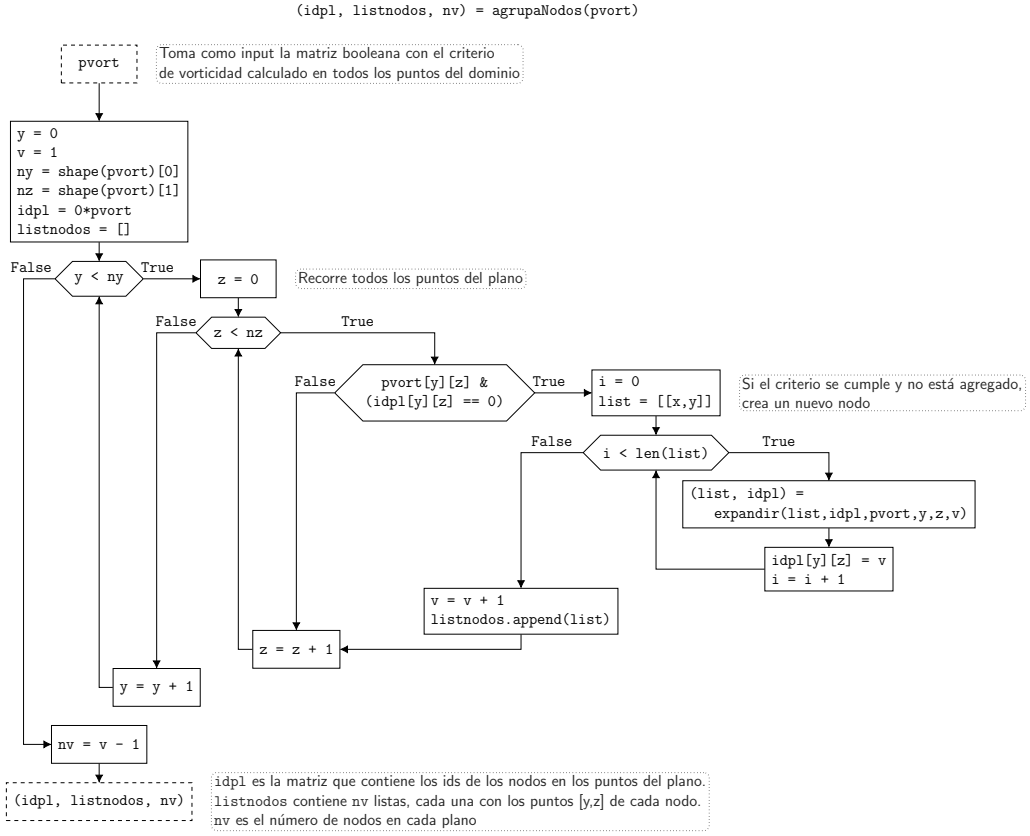


Figura 3.5: Diagrama de flujo del proceso usado para agrupar los nodos presentes en un determinado plano YZ.

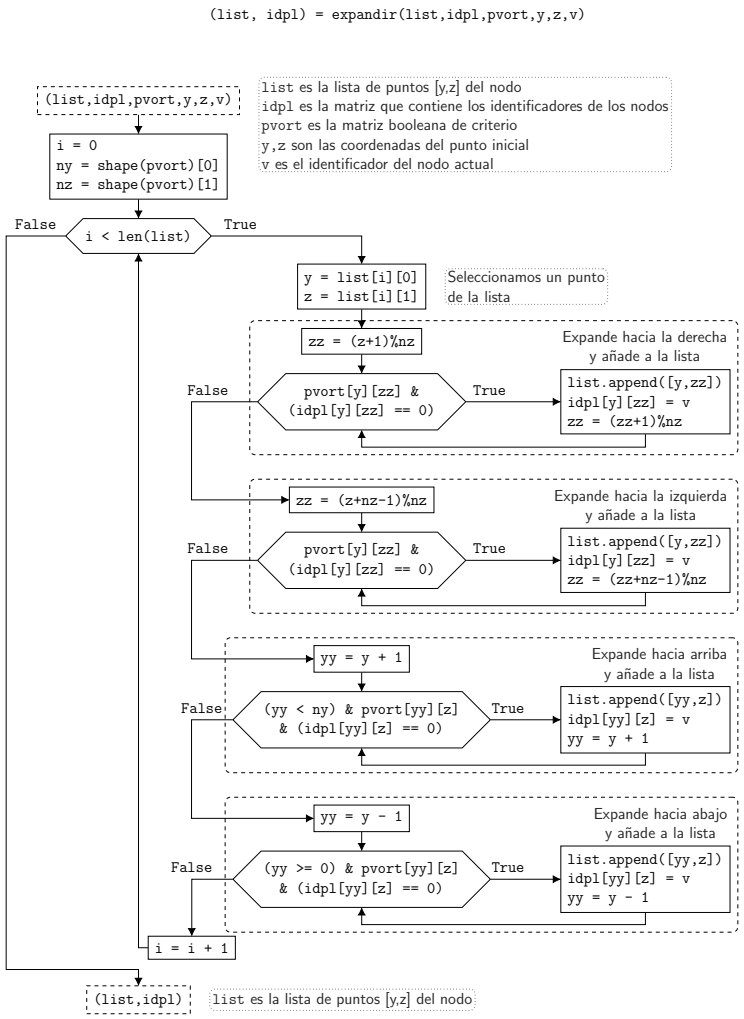


Figura 3.6: Diagrama de flujo del proceso usado para agrupar un nodo a partir de un punto inicial.

Este proceso se repite conforme el algoritmo recorre los puntos del plano, incrementando el valor del identificador para cada nodo obtenido. Una vez terminada la agrupación para un plano, se obtienen como resultados:

- Una matriz del tamaño del plano en la que marca el identificador del nodo al que pertenece cada punto (si cumple el criterio de vorticidad).
- Una lista de puntos contenidos en cada nodo.
- El número de nodos contenidos en dicho plano.

La paralelización de este proceso es directa asignando a cada procesador una serie de planos YZ, evitando de esta forma conflictos de memoria. El diagrama de flujo de este proceso de agrupación de nodos para un plano está presente en la Figura 3.5.

3.2.2 Conectividad de los nodos

Obtenidos los nodos, el algoritmo obtiene cómo se conectan entre ellos con los nodos de los planos YZ contiguos, o lo que es lo mismo, las listas de adyacencia de cada uno de los nodos. Para ello, el algoritmo recorre para cada plano YZ todos sus nodos, leyendo la lista de puntos de cada uno de ellos y comparándola con la matriz que contiene los identificadores de los nodos en cada punto del plano contiguo. El proceso a seguir, para cada punto del vórtice, es el siguiente:

1. Comprobar el valor del punto en el plano contiguo
2. Si contiene un identificador, comprobar si no está ya añadido a la lista de conectividad
3. Si no está añadido, añadirlo a la lista

Una vez hecho esto para todos los puntos de cada nodo, en todos los nodos de todos los planos YZ con sus planos contiguos anterior y posterior, teniendo en cuenta la periodicidad del dominio, se obtiene para cada nodo una lista de conectividad conteniendo los identificadores de los nodos con los que conecta en los planos anterior y posterior.

Este proceso necesita que cada procesador lea datos de planos que no corresponden a su bloque asignado, concretamente de los planos anterior y siguiente a este. Sin embargo, debido a que la paralelización se ha realizado con OpenMP y por tanto la memoria es compartida entre todos los procesadores, esto no supone un problema para la ejecución del programa. El diagrama de flujo del proceso seguido para obtener la conectividad entre un nodo y los del plano contiguo está presente en la Figura 3.7.

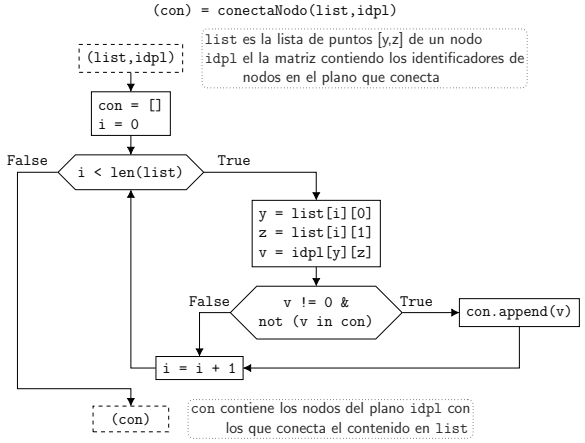


Figura 3.7: Diagrama de flujo del proceso usado para conectar un nodo con los presentes en los planos adyacentes.

3.2.3 Formación de estructuras tridimensionales

Una vez formados los nodos y obtenida su conectividad, podemos entender las estructuras coherentes tridimensionales como componentes conexos del grafo cuyos nodos son los agrupados previamente en cada plano y cuyas aristas están definidas por las listas de conectividad local. Para obtener los puntos que conforman estas estructuras y así sus propiedades estos componentes conexos deben ser reconstruidos a partir de la información disponible. Para reconstruirlas, el algoritmo sigue un proceso similar computacionalmente al presentado para la formación de nodos, también basado en un algoritmo de Búsqueda en Profundidad. El proceso a seguir es, partiendo de un nodo inicial no asignado a ninguna estructura tridimensional como único punto del componente:

1. Obtener los nodos adyacentes al seleccionado a partir de la conectividad local
2. Si dichos nodos no han sido ya añadidos al componente, añadirlos
3. Seleccionar el siguiente nodo
4. Repetir hasta llegar al último nodo, cuyos nodos adyacentes han sido ya añadidos.

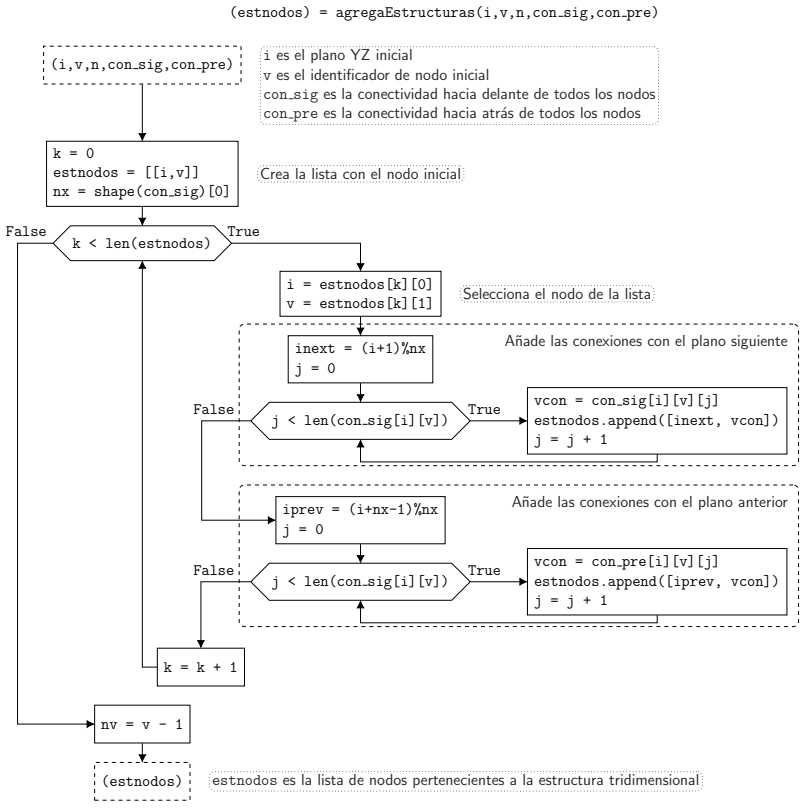


Figura 3.8: Diagrama de flujo del proceso usado para reconstruir los componentes conexos del grafo a partir de sus nodos y aristas.

Una vez agregado el componente correspondiente a una estructura tridimensional, el algoritmo saca como resultado una lista con el plano en el que está situado y el identificador

de nodo correspondiente a cada uno de los nodos que lo forman, que puede ser usada para obtener todos sus puntos y así obtener sus características principales. El diagrama de flujo del programa usado para agregar estos componentes está representado en la Figura 3.8.

La paralelización de este proceso deja de ser directa, al poder llegar dos procesadores al mismo componente reconstruyendo a partir de nodos iniciales muy lejanos entre sí. Para evitar duplicidades, el paso de guardado de cada uno de estos componentes se ha marcado como *crítico*, o lo que es lo mismo, solo un procesador lo realiza a la vez. El proceso que sigue el programa dentro de este paso crítico es:

1. Comprobar si el primer nodo de la estructura ha sido añadido ya a otra estructura.
2. En caso de que no haya sido añadido, marcar todos los nodos del componente como añadidos a una nueva estructura.
3. Guardar la lista de nodos dentro de una nueva estructura tridimensional
4. Incrementar el contador de estructuras tridimensionales con el número de procesadores, para evitar conflictos de memoria.

Existen algoritmos más eficientes computacionalmente para la realización de este proceso, destacando el de Chin *et al* [27], consistente en la formación en paralelo de *supernodos* conexos entre sí hasta obtener una serie de componentes conexos no conectados entre ellos.

3.2.4 Extracción de características de las estructuras

Con los componentes representativos de las estructuras tridimensionales ya obtenidos, el programa pasa a obtener las características geométricas más representativas, en paralelo. Para ello, reparte las estructuras entre los procesadores disponibles, y para cada una de ellas obtiene las listas de puntos que la forman, accediendo a los puntos que conforman cada uno de los nodos del componente. Con esta lista de puntos disponible, el proceso seguido para obtener cada una de las características principales escogidas es:

- **Volumen**: al solo ser variable la distancia entre nodos en el eje y, el volumen del elemento correspondiente a cada punto solo depende de su posición en y. Por lo tanto, para calcular el volumen de cada estructura basta con calcular el área de la estructura coherente en cada plano XZ para luego integrar mediante la Regla del Trapecio.
- **Caja envolvente**: para cada estructura tridimensional se puede definir un único prisma rectangular con aristas paralelas a las direcciones cartesianas de volumen mínimo que contenga todos los puntos de la estructura. En caso de que la estructura no se vea cortada por los planos periódico del dominio, las dimensiones de la caja se pueden calcular como la diferencia entre el punto máximo y mínimo en cada una de las dimensiones. En la dimensión perpendicular a la pared, al no ser periódica, se usará siempre este proceso.

En el caso de existir periodicidad, repetimos la operación pero desplazando el plano de periodicidad a la mitad del dominio sumándole a cada punto de la estructura la mitad de la longitud del dominio en la dirección en la que sucede la periodicidad y calculando el resto con la longitud del dominio en ese eje. Si en ese caso aún existe periodicidad, el algoritmo desplaza el plano periódico punto por punto de la misma forma hasta encontrar uno que no corte la estructura.

Una vez obtenida una estructura no cortada por el plano periódico del dominio, se pueden calcular las dimensiones de la caja envolvente obteniendo las diferencias entre los máximos y mínimos en cada una de las dimensiones.

- **Centro de masas**: en caso de que la dimensión considerada del vórtice no esté afectada por la periodicidad, el centro de masas se puede calcular como la media de las coordenadas en cada dirección cartesiana de todos los puntos de la estructura.

  En el caso de estar la estructura partida debido a la periodicidad, se realiza un proceso como en el punto anterior de desplazamiento del plano de periodicidad, se calcula la coordenada del centro de masas en la dirección en la que ocurre la periodicidad, se resta a ese centro de masas el avance producido por el desplazamiento del plano y se normaliza dentro de los límites del dominio. Por último, para mejorar la precisión de la comparación temporal, se le añade al valor del eje x el desplazamiento medio del fluido en la coordenada y de su centro de masas.

Todos estos procesos están detallados en el diagrama de flujo presente en la Figura 3.9.

3.2.5 Secuencia de procesos

El algoritmo no hace estos procesos de manera estrictamente secuencial, en parte debido al hecho de que está concebido para ejecutarse en paralelo. Esta paralelización, implementada mediante directivas de *work-sharing* `!$OMP DO`, consiste en dividir la variable sobre la que se itera en una serie de bloques, cada uno correspondiente a las tareas que va a realizar cada hilo.

En primer lugar, una vez hechas las inicializaciones pertinentes de las variables e iniciado el entorno paralelo `!$OMP PARALLEL`, cada procesador agrega los vórtices del último plano de los asignados al procesador anterior (en caso de ser el primero, agrega el último plano del dominio). Posteriormente, para cada uno de los planos del bloque asignado a cada procesador, agrega los nodos si no lo ha hecho otro hilo antes y obtiene la conectividad hacia delante de los nodos del plano anterior y la conectividad hacia detrás de los del plano actual. Tras agregar y conectar los nodos, el programa obtiene todos los componentes conexos del grafo usando como puntos de partida cada procesador los nodos presentes en los planos de su bloque, guardando resultados en un paso crítico para evitar conflictos.

Por último, estas estructuras se reparten entre los procesadores disponibles para que obtengan de cada una de ellas todos los puntos que pertenecen a cada estructura y calculen sus características geométricas. Todo el proceso está representado en el diagrama de flujo presente en la Figura 3.10.

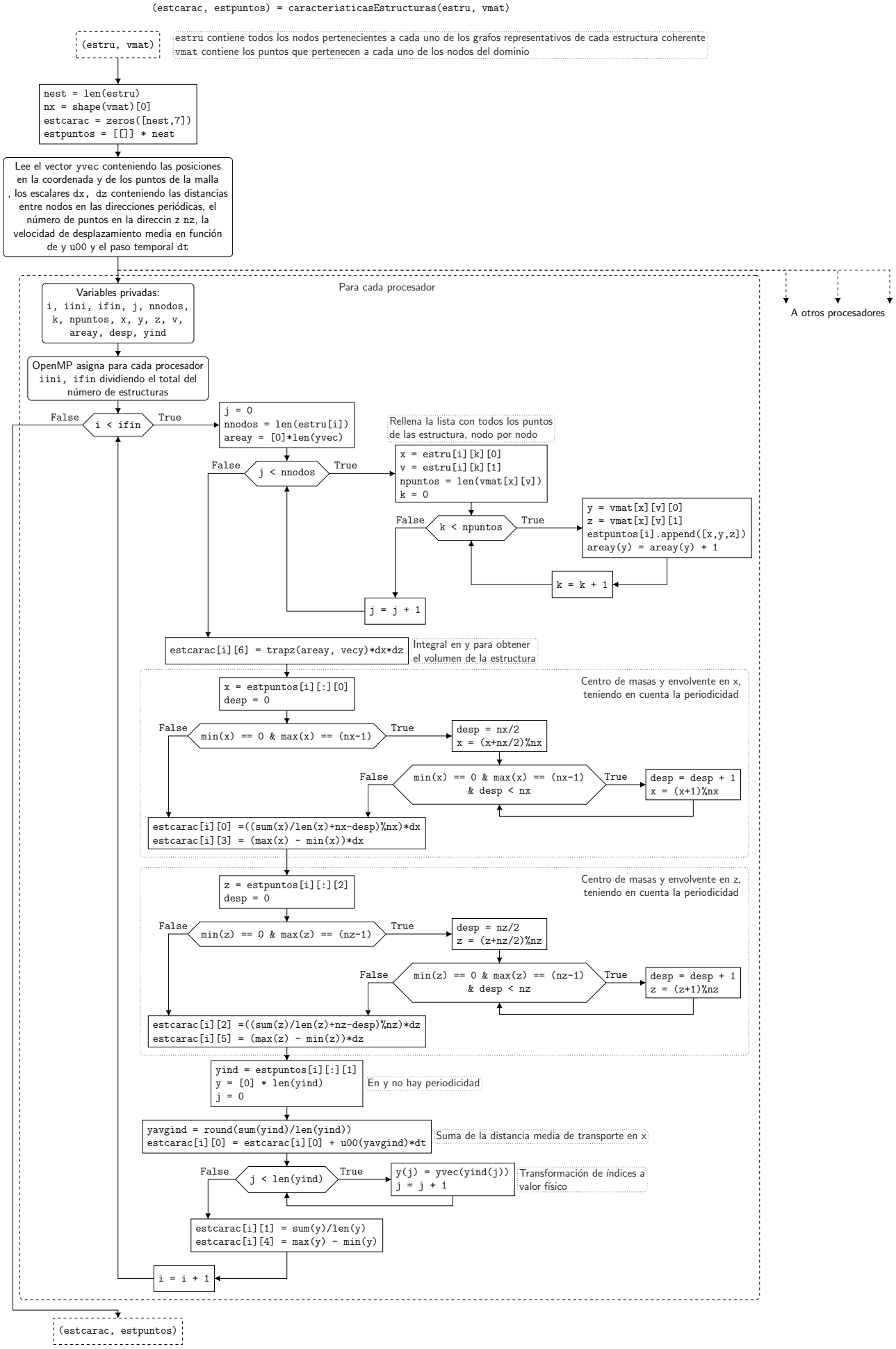


Figura 3.9: Diagrama de flujo del programa de extracción de las características geométricas de los vórtices ya agregados.

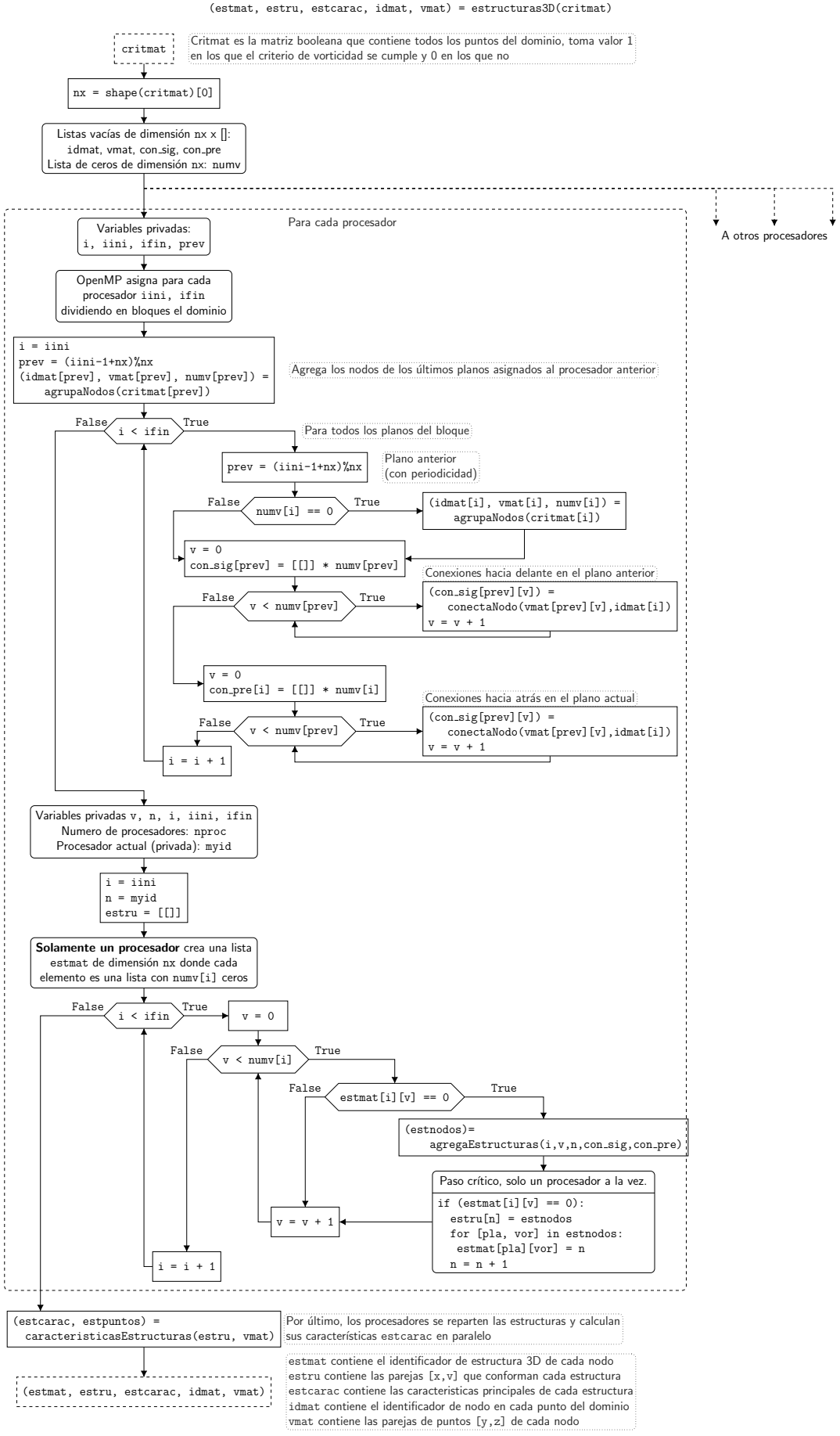


Figura 3.10: Diagrama de flujo del programa de agregado de estructuras tridimensionales y extracción de sus características geométricas a partir de la matriz booleana.

3.3 Algoritmo de evolución temporal

Una vez agregadas las estructuras y sacadas sus características principales, se ha desarrollado un algoritmo en paralelo para correlacionar las estructuras tridimensionales obtenidas en un paso con las del anterior. Este algoritmo genera una base de datos de torbellinos, marcando en cada uno de ellos:

- **Nacimiento**: tanto si se forma cerca de la pared como si procede de la división de otro torbellino, indicando en este caso la posición del torbellino *padre* en la base de datos.
- **Muerte**: sea por disipación viscosa o dividiéndose, referenciando si es así a las posiciones de los torbellinos *hijos*.
- **Evolución**: proporcionando una forma de reconstruir el vórtice indicando el identificador de la estructura que constituye ese vórtice en cada paso temporal.
- **Divisiones**: en ciertos casos, un torbellino puede dividirse de forma asimétrica, en tanto que uno de los torbellinos resultantes es muy similar al original y otro no. En este caso, se considera como una evolución del mismo torbellino y se marca el vórtice no similar como una división del primero.
- **Uniones**: análogamente, dos vórtices de tamaños muy dispares pueden unirse. Este caso se considera como una evolución del vórtice grande (que será el más similar al vórtice resultante de la unión), considerando muerto al más pequeño.

Para seguir la evolución de los vórtices, el algoritmo desarrollado realiza dos tipos de correlaciones entre las estructuras presentes en dos pasos temporales consecutivos:

3.3.1 Conexiones *fáciles*

Se ha definido en este trabajo como conexión *fácil* entre dos estructuras la que solamente utiliza los valores de sus características geométricas obtenidas mediante el proceso explicado en §3.2.4. Esta conexión está ideada para relacionar solamente los vórtices que siguen vivos en iteraciones sucesivas, pero al ser este el caso más común entre pasos temporales (la mayoría de torbellinos presentes no nacen o mueren en cada paso temporal), la eficiencia de clasificación de estos torbellinos determinará en gran medida el tiempo de ejecución del algoritmo.

El proceso seguido por el programa para buscar este tipo de conexiones en cada estructura tridimensional de un paso temporal es el siguiente:

1. Calcular el error entre dicha estructura y todas las obtenidas en el paso temporal anterior como la distancia en $\mathbb{R}^7$ de los vectores de características, teniendo en cuenta la periodicidad del dominio, mediante el proceso detallado en el diagrama de flujo representado en la Figura 3.11.

2. Obtener el mínimo del error y el índice de la estructura del paso temporal anterior en el que ocurre el mínimo, si el error está por debajo de un nivel marcado.

3. Si la estructura del paso anterior no tiene conexión marcada, marcar su conexión hacia delante y la conexión hacia atrás en el tiempo de la estructura seleccionada.

4. Si la estructura ya tiene una conexión marcada, comprobar cuál de las dos tiene un error menor, y si es la actual, marcarla y desmarcar la otra.

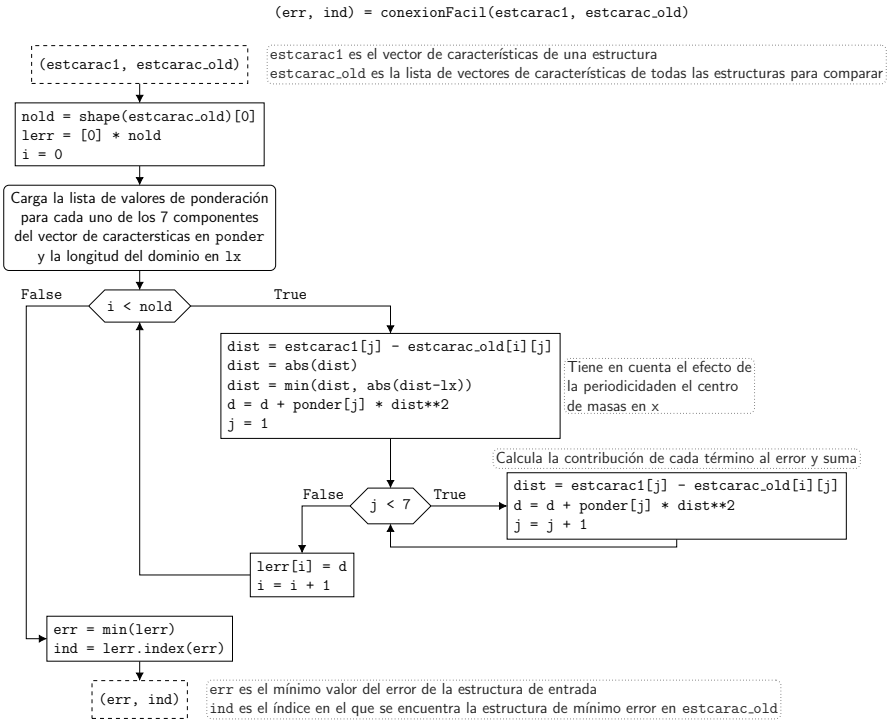


Figura 3.11: Diagrama de flujo del proceso usado para calcular la estructura más similar y su error a una estructura dada, usado para las conexiones *fáciles*.

El valor máximo del error y las constantes de ponderación de cada una de las características se pueden considerar la piedra angular del proceso, al depender la precisión del algoritmo fuertemente de la relevancia física de estos valores. Por lo tanto, se ha procedido a realizar un estudio de optimización no lineal para decidir el criterio escogido. Este estudio, detallado en §3.6, es solo válido para el número de Reynolds de pared iguales a los de las simulaciones empleadas, cuyas características están detalladas en la Tabla 2.1.

Al poder realizarse el proceso de comparación para cada una de las estructuras sin interferencia con las demás, y el acceso a la memoria donde se almacenan las características de cada estructura es compartido para todos los procesadores, el proceso descrito en la Figura 3.11 tendría una paralelización sencilla. Sin embargo, los pasos 3 y 4 de la lista anterior sí pueden requerir a un hilo entrar a los dominios de otro, por lo que el paso deberá ser crítico para evitar conflictos de memoria.

3.3.2 Conexiones *difíciles*

No todas las conexiones entre las estructuras tridimensionales de dos pasos temporales consecutivos pueden ser obtenidas mediante las comparaciones explicadas anteriormente, ya sea porque este se divida o fusione con otros o porque el criterio de error usado sea demasiado estricto para la evolución sufrida por un vórtice en concreto. Por lo tanto, aparece la necesidad de desarrollar un proceso para registrar todas las demás conexiones entre estructuras en dos pasos temporales, que han sido denominadas como conexiones *difíciles*.

Estas conexiones difíciles se basan en la búsqueda punto por punto de intersecciones entre estructuras tridimensionales en dos pasos temporales consecutivos. El proceso seguido, primero para cada una de las estructuras del paso temporal actual que no tienen ninguna conexión *fácil*:

1. Recorre la lista de nodos que pertenecen a dicha estructura tridimensional.
2. Aplicando el algoritmo descrito en §3.2.2, busca con qué nodos conecta del paso anterior, tanto en el mismo plano como en el anterior en la dirección del flujo.
3. Obtiene el identificador de la estructura 3D a la que pertenecen los nodos con los que ha detectado conexión.
4. Si dicha estructura no está en la lista de conexiones, la agrega.

Este proceso, cuyo diagrama de flujo está presente en la Figura 3.12, devuelve una lista de identificadores de estructuras tridimensionales del paso temporal anterior con las que conecta cada estructura evaluada del paso actual. Después, el proceso se repite para las estructuras coherentes del paso anterior que no tienen ninguna conexión *fácil*.

La obtención de estas conexiones es un proceso cuyos resultados solo se escriben para la estructura asignada, por lo que no puede llevar a conflictos de escritura en memoria. Por lo tanto, la inclusión de este proceso en un programa en paralelo OMP no supondría ningún problema ni sería necesaria la inclusión de ninguna directiva de control.

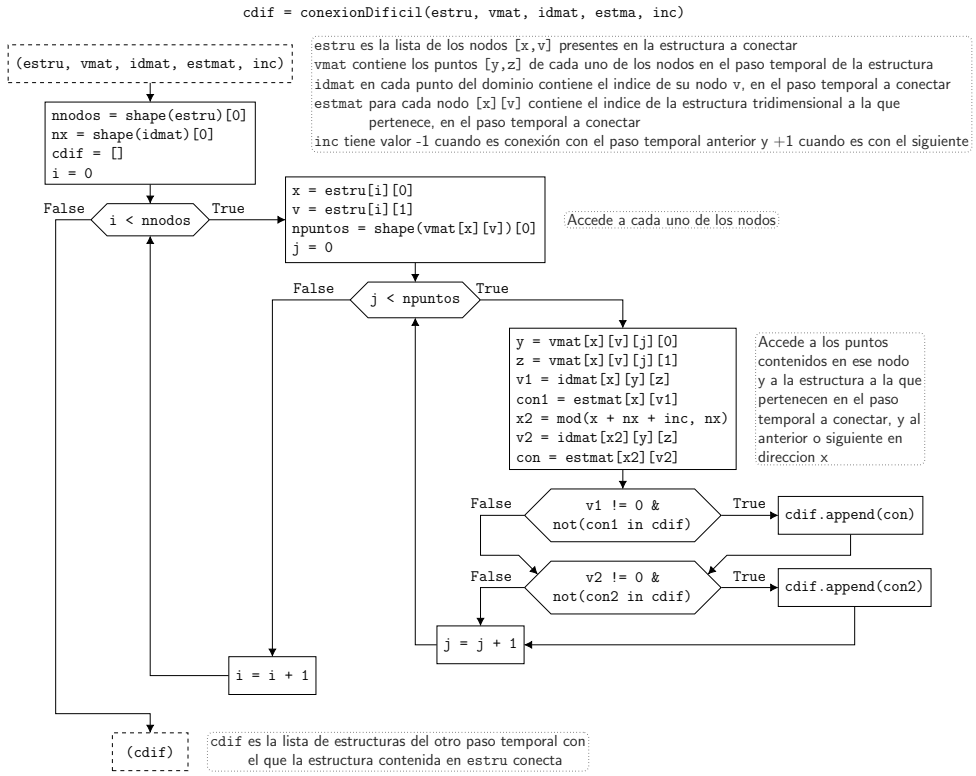


Figura 3.12: Diagrama de flujo del proceso usado para obtener todas las estructuras que coinciden en el espacio con otra de otro paso temporal, usado en las conexiones *difíciles*

3.3.3 Actualización de la base de datos

Con todas las conexiones entre estructuras realizadas, el siguiente paso en el proceso es actualizar la base de datos con la información obtenida en el paso temporal. Para ello, el algoritmo recorre todas las estructuras del paso temporal actual en paralelo y las clasifica de la siguiente forma:

- Si no tiene conexiones *fáciles* ni *difíciles*, se crea una nueva entrada en la base de datos.
- Si tiene conexión *fácil*, no se hace nada en esta primera comprobación.
- Si no tiene conexión *fácil* pero sí *difícil*, hay varias posibilidades:
  - Si la conexión *difícil* es única para las dos estructuras que participan, se establece que la conexión es equivalente a una conexión *fácil*.
  - Si la conexión es única solamente para la estructura del paso temporal actual, se comparan los volúmenes de las estructuras actual y la que conecta con ésta.
    - Si están dentro del margen de error, se marca como si la conexión fuese *fácil*, comprobando antes que en el periodo entre la primera comprobación y esta ningún otro hilo ha marcado una conexión como *fácil* en el torbellino

el paso anterior. Si es así, se comprueba cuál de las dos (la marcada por otro hilo y la del hilo actual) es más similar a la del paso anterior, se establece la conexión con la que lo sea y se suprime la otra conexión. Al estar este paso en conflicto con otro hilo, será crítico para evitar conflictos de memoria.

    * Si no, crea una nueva entrada y se marca como una división del torbellino al que pertenece la estructura del paso anterior.

- Si la conexión *difícil* no es única en la estructura del paso temporal actual, buscamos una estructura de volumen similar entre las conexas.

    * Si hay una dentro del margen de error, se asume que la conexión es *fácil* y se añade a la evolución del torbellino, realizando el mismo proceso dentro de un bloque crítico que en el caso anterior.

    * Si no, se crea una nueva entrada y se marca como el resultado de varios torbellinos uniéndose, para que estos sean posteriormente marcados como muertos.

Una vez hecha esta clasificación, el programa vuelve a recorrer todas las estructuras en paralelo, añadiendo la información de la evolución temporal de los vórtices en las estructuras con conexiones *fáciles* o en las marcadas como si las tuviesen.

Por último, el algoritmo recorre todas las estructuras del paso temporal anterior en paralelo, y marca las que no han sido clasificadas de ninguna forma en el proceso de decisión anterior como muertas. Si conectan de forma *difícil* con alguna estructura del paso temporal actual, incluye las uniones en la entrada del torbellino al que se unen.

La paralelización de este proceso no es directa, al necesitar la inclusión de las directivas `!$OMP CRITICAL` en los casos comentados anteriormente. El diagrama de flujo de todo el proceso de actualización de la base de datos está presente en la Figura 3.13

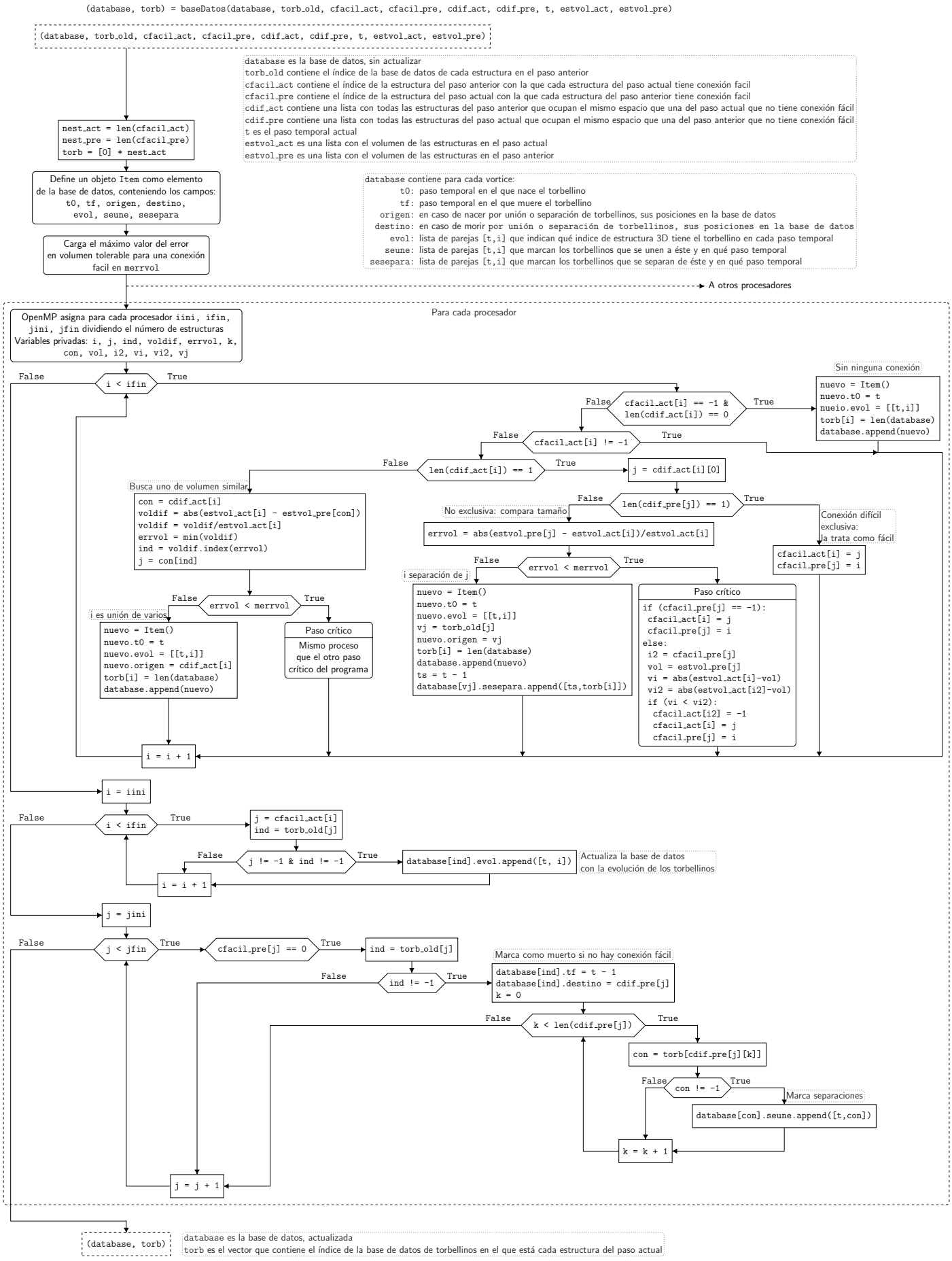


Figura 3.13: Diagrama de flujo del algoritmo de actualización de la base de datos de torbellinos.

3.4 Orden de operaciones en el programa

Primero, el código reserva la memoria necesaria para guardar todas las variables usadas de dimensión fija, lee la matriz booleana que contiene la información del criterio de vorticidad en cada punto en el primer paso temporal y procede a agregar sus estructuras tridimensionales. Posteriormente, crea la base de datos vacía y empieza el bucle principal.

Mientras el programa disponga de bases de datos para leer, ejecutará el bucle principal. Los pasos que incluye este bucle principal son:

1. Guardar la información de las estructuras tridimensionales obtenida en el paso anterior, para proceder después a la comparación con las obtenidas en el paso actual.

2. Leer la matriz booleana de la base de datos de este nuevo paso temporal, y el número de este paso.

3. Agregar las estructuras tridimensionales, en paralelo por bloques dentro del dominio, según se describe en §3.2.

4. Realizar las conexiones *fáciles* en paralelo para cada una de las estructuras obtenidas en el paso actual, como se describe en §3.3.1.

5. Realizar las conexiones *difíciles* en paralelo para cada una de las estructuras obtenidas en el paso actual que no tienen conexión *fácil* como está descrito en §3.3.2.

6. Actualizar la base de datos a partir de las conexiones obtenidas, en paralelo para cada una de las estructuras del paso actual primero y del anterior después, según el proceso de §3.3.3.

El proceso general del algoritmo está representado en el diagrama de flujo presente en la Figura 3.14.

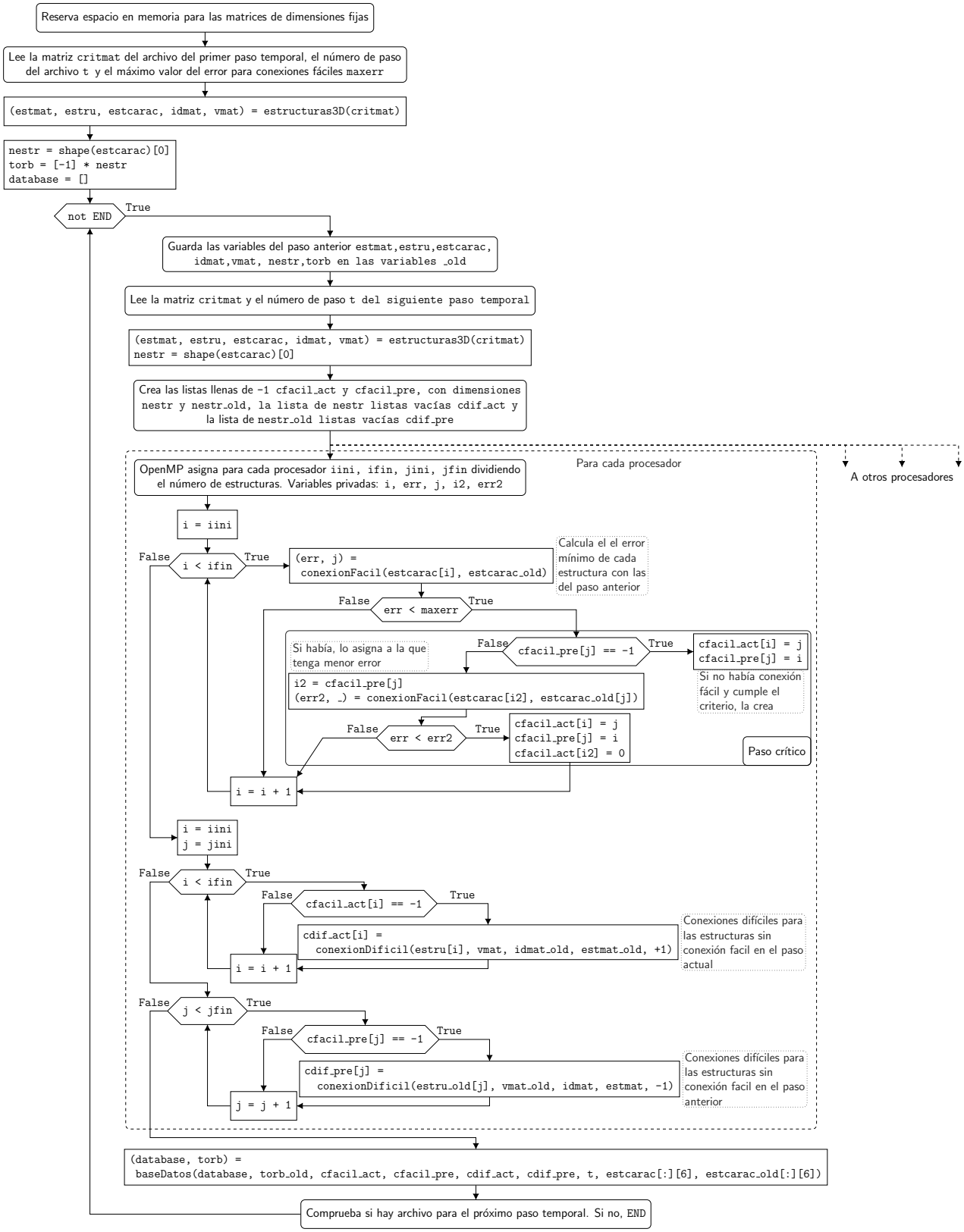


Figura 3.14: Diagrama de flujo de la función principal del algoritmo desarrollado.

3.5 Aspectos de la implementación en Fortran y paralelización

El algoritmo ha sido presentado en los diagramas de forma simplificada para facilitar su lectura, perdiendo así información concreta sobre los procesos seguidos para optimizar el tiempo de ejecución del programa. Los cambios más importantes son:

- **Listas dinámicas:** por definición, las listas dinámicas son ineficientes computacionalmente al tener que reasignar el espacio que ocupan en memoria cada vez que se les añade un elemento. En el código desarrollado se han tomado dos aproximaciones para evitarlas:
  - Cuando la variable contiene un número de elementos dentro de un orden de magnitud constante, se pre-asignan las dimensiones de las variables en un orden de magnitud superior. Esta aproximación se ha tomado en variables representativas de la conectividad entre nodos, los nodos en cada plano o las estructuras coherentes en todo el campo, cuyos números permanecen dentro de unos márgenes en todos los casos comprobados.
  - En los casos en los que los números de elementos de las matrices no permanecen dentro de unos márgenes, se ha optado por usar una interfaz a un tamaño fijo en la subrutina que realiza la lógica, para luego guardar solo los datos necesarios dentro de un tipo derivado, que puede ser asingnado en memoria de forma más flexible.
- **Interfaces explícitas:** en muchos de los casos, las variables con las que trabajan tanto la rutina principal como las diferentes subrutinas son las mismas. Para evitar que el compilador asigne espacio de memoria varias veces para lo mismo, lo que sucedería usando *dummy arguments* para todas las variables, éstas se guardan dentro de un módulo al que llaman las subrutinas que acceden a las variables.
- **Orden de las dimensiones:** en los diagramas de flujo presentados anteriormente se ha buscado la inteligibilidad y para ello se han situado los índices en un orden análogo a los bucles de operaciones. Sin embargo, esta práctica en Fortran es extremadamente ineficiente debido a las múltiples cargas en caché que supone, por lo que se ha usado un orden de índices que maximice las operaciones consecutivas en el primero de los índices.
- **Órdenes de OpenMP:** en los diagramas sólo se han presentado tres secciones críticas, pero no es la única orden de sincronización y coordinación del código, omitidas por no considerarse necesarias para la inteligibilidad de los diagramas al ser usadas en asignaciones de memoria, operaciones no consideradas en los diagramas. Dos ejemplos de estas órdenes son:
  - Las órdenes representadas como `.append( )` en el diagrama contienen una orden `!$OMP CRITICAL` en el código fuente, usada para reservar el punto de la matriz en el que el programa escribirá los datos.

  - En los casos en los que es necesario asignar memoria dentro de una sección en paralelo, solo un procesador realiza esa tarea. Para ello, si es una variable la orden se incluye dentro de un `!$OMP SINGLE` y, si es una asignación a un componente de un tipo derivado, `!$OMP CRITICAL`.

- **Compresión de datos:** el sistema HDF5 [28], usado para input-output en el programa en Fortran, no permite guardar datos en formato booleano, de crucial interés para este trabajo. Por lo tanto, para agilizar la lectura y escritura en disco, los campos booleanos se comprimen representando 32 puntos dentro de un número entero y se descomprimen una vez leído el archivo, reduciendo el espacio ocupado en disco y los tiempos globales de lectura/escritura.

La implementación en Fortran del algoritmo presentado a lo largo de este capítulo supone el programa principal de todos los desarrollados y se compone de 2967 líneas de código.

3.5.1 Proceso de desarrollo del algoritmo

Este algoritmo no se ha desarrollado directamente en Fortran en paralelo, sino que se ha seguido un proceso incrementando a cada paso la complejidad del resultado final, validando los resultados de cada uno de los niveles para asegurar el correcto funcionamiento de este. Los pasos en el desarrollo de este han sido:

1. Desarrollo conceptual de un algoritmo: en pseudocódigo y diagramas de flujo, para obtener así su estructura y las líneas generales de su funcionamiento. Esta parte, por su naturaleza, no puede ser validada.

2. Implementación en un lenguaje de programación interpretado de alto nivel: el algoritmo desarrollado se ha implementado en primer lugar en serie, en MATLAB, como lenguaje de programación de fácil depuración y producción de imágenes y resultados. Una vez estos resultados han sido generados, se pueden utilizar para validar cada sección del código comparando su salida con las salidas planteadas en el paso anterior.

3. Implementación del algoritmo en Fortran: una vez garantizado que las salidas producidas en el programa en MATLAB son las salidas deseadas, se programó el algoritmo en Fortran pensando en su paralelización, ejecutándose primero en serie sin optimizaciones para validar resultados frente a los producidos anteriormente, comprobando que son idénticos.

4. Paralelización y optimización: por último, se ha paralelizado el código mediante la incorporación de órdenes de OpenMP y se han realizado una serie de optimizaciones para mejorar el rendimiento del código. Estos cambios se han validado comprobando que todos los resultados siguen siendo los mismos que en los pasos anteriores.

3.5.2 Validación del programa

Una vez terminado el desarrollo del código, disponiendo ya de un programa funcional en paralelo que produce resultados, se procedió a realizar una serie de procesos de validación para comprobar que estos resultados producidos son certeros y que no existen duplicidades o contradicciones dentro entre ellos. Esta verificación se puede separar en dos partes:

- **Verificación del agregado 3D:** la verificación de esta sección del código ha consistido en, disponiendo de las listas de puntos de cada una de las estructuras agregadas para una serie de pasos temporales, comprobar que:
  - Cada punto solo pertenece a, como máximo, un vórtice.
  - Los vecinos de todos los puntos pertenecientes a un vórtice sólo pueden pertenecer a ese mismo vórtice, en el caso de pertenecer a alguno.
  - Reconstruyendo el campo fluido booleano a partir de todos los vórtices, este es exactamente igual al campo de entrada.

  Al cumplirse esto para todos los pasos temporales comprobados, esta sección del código se ha considerado validada.
- **Verificación de la evolución temporal:** la verificación de esta sección del código se ha hecho mediante la inspección de las matrices de conectividad y de la base de datos. Se ha comprobado que:
  - Cada estructura coherente sólo aparece como conexión *fácil* de, como máximo, una otra estructura del paso temporal anterior o siguiente, según corresponda.
  - La correspondencia entre entrada de la base de datos y vórtice en un paso temporal es recíproca y unívoca, o lo que es lo mismo, que en cada paso temporal solo un vórtice se corresponde a un índice de la base de datos y este índice además solo contiene al vórtice en ese paso temporal.

  Además, para verificar que el seguimiento es correcto se han producido imágenes y videos siguiendo a torbellinos de la base de datos, comprobando en ellos que la evolución de éstos tiene sentido físico. Como todas estas pruebas han dado resultados favorables en los casos comprobados, se ha considerado esta sección del programa validada también.

3.6 Estudio de calibración del criterio de error

Para realizar las conexiones *fáciles* correctamente, es necesario calibrar el modelo para la simulación empleada. Al considerar en el algoritmo 7 variables geométricas de las estructuras coherentes, se ha procedido a ajustar una constante para cada una de estas variables geométricas y otra para valor máximo de error aceptable para considerar una conexión como *fácil*.

Para ajustar estas 8 constantes, se ha optado por realizar una optimización no lineal con restricciones, buscando maximizar el número de conexiones realizadas manteniendo la probabilidad de falso positivo dentro de un límite lo más pequeño posible.

En primer lugar, para tener una base de datos de calibración, se ha lanzado el algoritmo estableciendo el límite del error para las conexiones *fáciles* a 0, para forzar que todas las conexiones sean difíciles, guardando en cada paso temporal la matriz que contiene la información de las características geométricas de todas las estructuras y las listas que contienen los índices de las estructuras con las que conectan en el paso anterior y siguiente.

Disponiendo de esta información, se ha lanzado una optimización no lineal con restricciones con las siguientes características:

- **Variables de estado:** el algoritmo ajustará los valores de las 8 constantes del algoritmo de conexión *fácil*:
  - Las 3 constantes de ponderación del centro de masas de la estructura coherente: c_x, c_y, c_z
  - Las 3 constantes correspondientes a las dimensiones de la caja envolvente de la estructura coherente: b_x, b_y, b_z
  - La constante relativa al volumen de la estructura coherente: v
  - El límite del valor del error para el cual se considera una conexión *fácil*: m_{err}

  Todas ellas estarán comprendidas en el rango $[0, 1]$. Las constantes de las variables geométricas y la del error no son independientes, por lo que el óptimo será una linea de óptimos, en la que el algoritmo de optimización parará una vez la encuentre. Se ha decidido usar esta aproximación en vez de eliminar una de las variables para asegurar que todos los óptimos están dentro del rango deseado, al ser su interdependencia no lineal. Sus valores iniciales y rangos están presentes en la Tabla 3.1, junto con sus valores en el óptimo local encontrado.

$$\vec{x} = [c_x, c_y, c_z, b_x, b_y, b_z, v, m_{err}]^T \tag{3.1}$$

- **Función Objetivo:** se ha establecido la probabilidad de acierto de una conexión fácil $p_a = f(\vec{x})$ como función a maximizar. Para obtener este valor, se corre el algoritmo desarrollado para las conexiones fáciles con las constantes introducidas por el optimizador sobre las matrices guardadas y se comparan los resultados guardados, obteniendo los aciertos y los falsos positivos.

$$\max_{\vec{x} \in [0,1]} f(\vec{x}) \tag{3.2}$$

- **Restricciones:** como única restricción de desigualdad se ha impuesto que la probabilidad de falso positivo $p_f = g(\vec{x})$ sea menor que 10^{-6}.

$$g(\vec{x}) \leq 10^{-6} \tag{3.3}$$

- **Algoritmo de Optimización:** se ha utilizado el algoritmo de Punto Interior [29], ya que permite resolver problemas de optimización con restricciones. El gradiente se ha aproximado mediante diferencias finitas de primer orden. Otros métodos más eficientes, como el algoritmo SQP (programación cuadrática secuencial) también han sido probados, pero el algoritmo de Punto Interior ha producido resultados más óptimos y ha presentado mayor estabilidad de cálculo. Como, por la naturaleza de este problema, no podemos asumir que sea convexo, la presente optimización es un problema no polinomial y el óptimo encontrado será un óptimo local, permitiendo la existencia de otros puntos más óptimos.

  Las características de la optimización lanzada son:

  - Paso de diferencias finitas comprendido en el rango $[0{,}001, 0{,}5] \cdot x_{i0}$ para cada una de las variables de estado, donde x_{i0} es su valor inicial.
  - Mínima norma del paso de avance de $10^{-9} \cdot \vec{x}_0$
  - Tolerancia de las restricciones de 10^{-12}.

Una vez lanzada la optimización con los datos de características geométricas y conexiones *difíciles* de 70 pasos temporales consecutivos, se ha obtenido un óptimo local en el punto presente en la Tabla 3.1. En este punto se ha obtenido, para los casos de la optimización:

$$p_a = f(\vec{x}_{opt}) = 0{,}71$$
$$p_f = g(\vec{x}_{opt}) = 0$$

Para comprobar la validez de esta optimización, se ha comprobado la eficacia de estas constantes en otros 50 pasos temporales consecutivos entre ellos pero no con los usados para la optimización, obteniendo:

$$p_a = f(\vec{x}_{opt}) = 0{,}062$$
$$p_f = g(\vec{x}_{opt}) = 0$$

Estos valores, pese a no ser tan beneficiosos como los anteriores (al no estar la optimización hecha específicamente para estos campos) se pueden seguir considerando como satisfactorios, por lo que estos valores serán usados para los casos lanzados.

Cabe destacar también que, al ser este un estudio de optimización no lineal por descenso del gradiente, los valores resultantes no tienen por qué presentar una correlación directa con la física detrás del movimiento vorticial, y que existen otros métodos para realizar las conexiones *fáciles* que podrían resultar también interesantes por poder adaptarse mejor al problema, como los basados en redes neuronales.

Variables de estado		**Valor inicial**	**Rango**		**Resultado**
Constantes del centro de masas	c_x	0.5	0.0	1.0	0.53
	c_y	0.5	0.0	1.0	0.51
	c_z	0.5	0.0	1.0	0.52
Constantes de la caja envolvente	b_x	0.5	0.0	1.0	0.50
	b_y	0.5	0.0	1.0	0.50
	b_z	0.5	0.0	1.0	0.48
Constante de volumen	v	0.5	0.0	1.0	0.50
Límite del error	m_{err}	0.02	0.0	1.0	$3.3{\cdot}10^{-5}$

Tabla 3.1: Variables de estado de la optimización, valores iniciales, límites inferior y superior de su rango, valores iniciales y valores del óptimo local encontrado

Capítulo 4

Resultados

Una vez ejecutado el algoritmo, se disponen de datos sobre evolución de vórtices e interacciones entre estos durante sus periodos de vida y procesos de nacimiento y muerte. Una forma sencilla y usada en la bibliografía de representar estos procesos es mediante el uso de grafos, como Lozano-Durán y Jiménez en [13], en el que separan las interacciones que en este trabajo se consideran como evolución del vórtice como ramas primarias del grafo de su vida, y las interacciones con otros como ramas secundarias.

El proceso seguido para la generación de estos grafos es sencilla una vez se dispone de la base de datos, pero puede ser computacionalmente costoso para vórtices con numerosas interacciones con otros. En este trabajo se ha optado por un algoritmo recursivo que, empezando por una entrada de la base de datos, se llama a sí mismo en cada una de las interacciones que la entrada tenga con otros torbellinos, siguiendo también la vida de éstos hasta que se acabe la información disponible o se sobrepase el límite de tiempo marcado.

Un ejemplo de grafo generado mediante este proceso está representado en la Figura 4.1. Disponiendo de grafos y de la información de las estructuras tridimensionales en cada punto, es posible también generar imágenes para seguir visualmente la evolución de estos vórtices a lo largo del tiempo.

A partir de estos grafos, se pueden evaluar visualmente las vidas de los vórtices en el fluido, y seleccionar cuáles de ellos pueden resultar interesantes o dignos de estudio. Este proceso puede ser automatizado para buscar ciertas características, pero eso queda fuera del alcance de este trabajo.

Se ha ejecutado el programa desarrollado sobre 200 pasos temporales consecutivos de la simulación descrita en la Tabla 2.1 y se han extraído una serie de casos que se han considerado interesantes de la base de datos, generando grafos e imágenes representativas de sus vidas (o de las secciones de ésta que se han considerado relevantes) para ser presentados a lo largo de este capítulo. Sin embargo, ninguno de estos resultados pueden ser descritos como universales ni se puede extraer conclusiones generales de ninguno de ellos, solo se presentan como ejemplos de casos que pueden ser observados mediante el programa desarrollado.

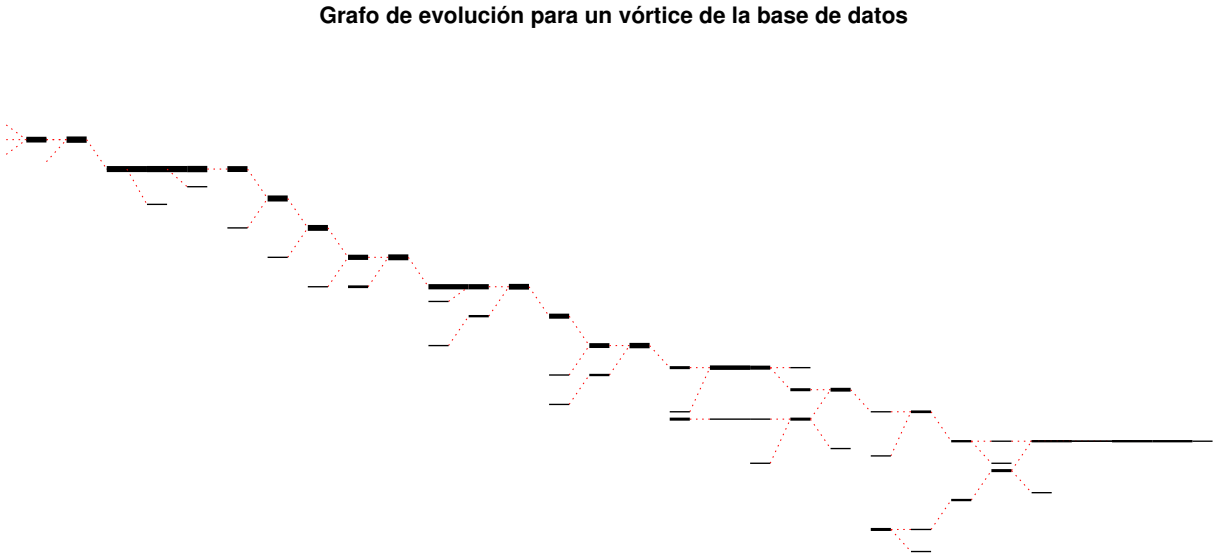


Figura 4.1: Representación de la vida de un torbellino mediante un grafo reconstruido desde la base de datos. Las líneas negras indican evolución de un mismo vórtice de la base de datos, su anchura representando el volumen que ocupa, su longitud su duración en el tiempo, y las líneas rojas las referencias a otros índices, sea mediante uniones o separaciones. El tiempo avanza de izquierda a derecha.

4.1 Interacciones entre dos torbellinos perdiendo energía

Como ejemplo de resultado a destacar, se ha obtenido de entre los resultados un ejemplo de interacción entre dos vórtices en su proceso de descender la cascada de energía. El grafo de su vida está representado en la Figura 4.2 y las capturas de su vida que se han considerado más relevantes están el la la Figura 4.3.

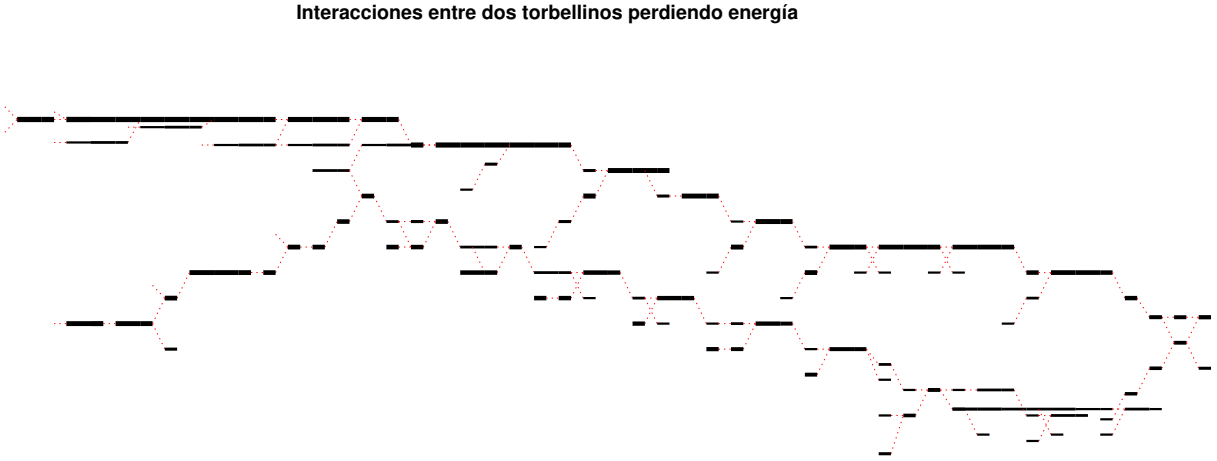


Figura 4.2: Grafo representativo de las interacciones de dos torbellinos perdiendo energía hasta disiparse.

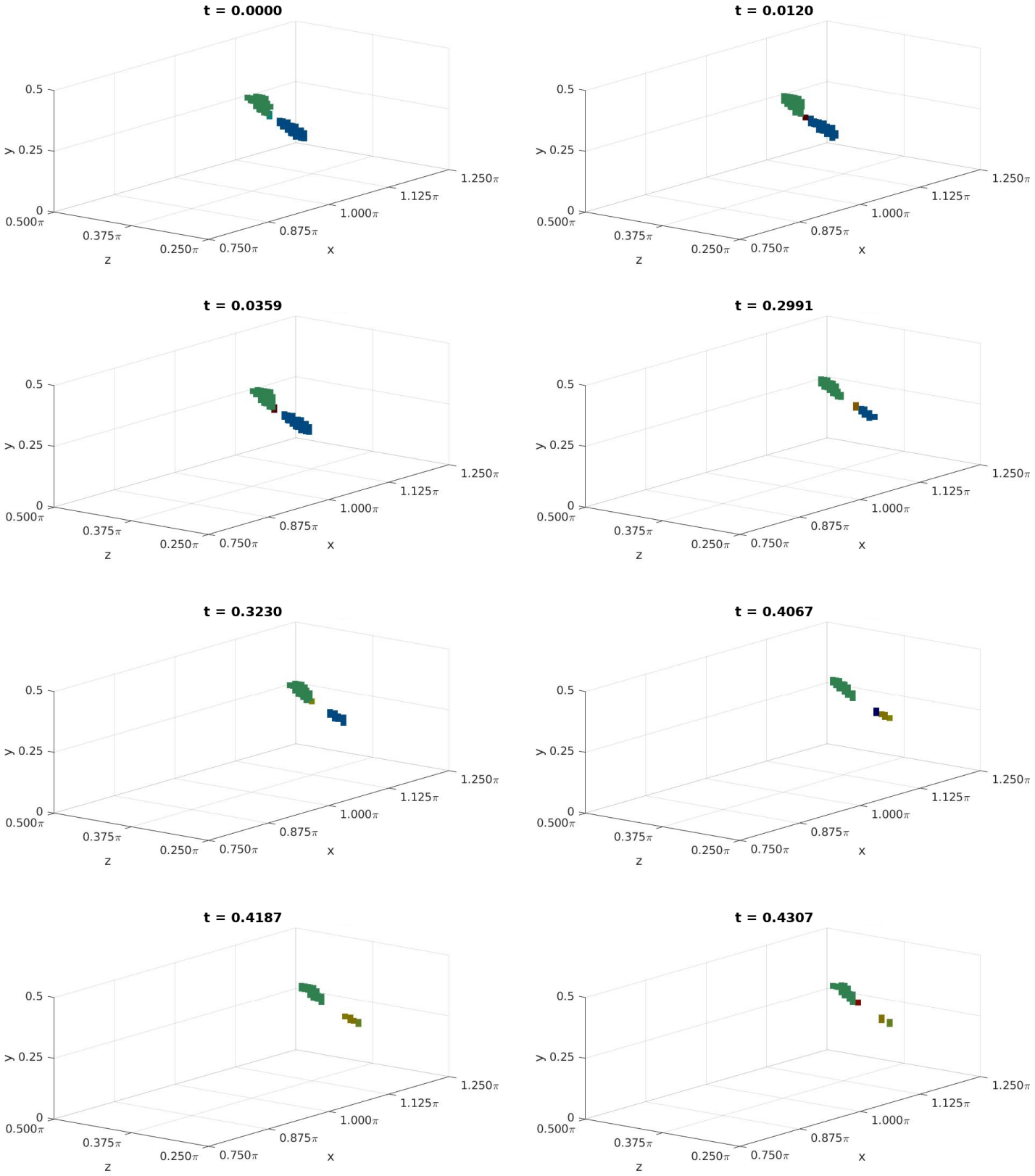


Figura 4.3: Representación de dos torbellinos perdiendo energía en una serie de pasos temporales representativos de este proceso.

Tanto en el grafo como en las imágenes se puede ver cómo los dos vórtices van perdiendo energía y, por tanto, reduciendo su tamaño. Este proceso se hace tanto por disipación dentro del propio torbellino como por separación de torbellinos más pequeños y división del torbellino en otros más pequeños. Durante este proceso estos torbellinos interaccionan entre ellos mediante los vórtices más pequeños en los que se separan y, en menor medida, se juntan, hasta disiparse completamente.

4.2 Formación de vórtices

Otro resultado que se ha extraído y que se considera muy interesante es el proceso de formación de vórtices a base de colisiones, uniones y crecimiento de los torbellinos. La Figura 4.4 muestra el grafo que describe este proceso y la Figura 4.5 contiene una serie de instantáneas de varios pasos temporales del proceso.

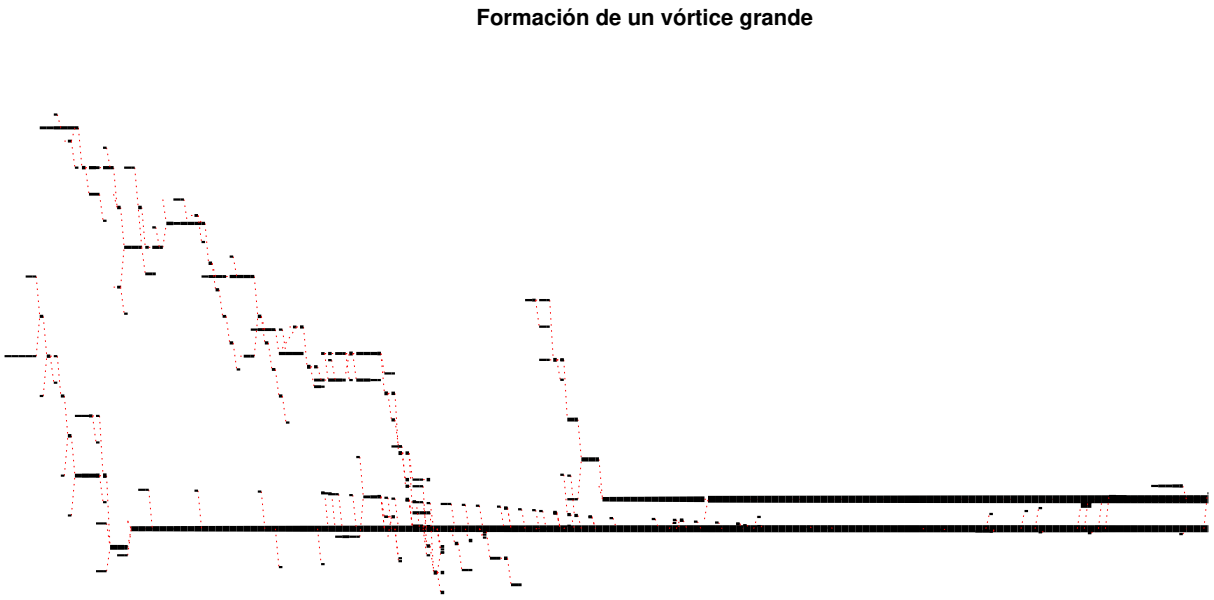


Figura 4.4: Grafo representativo de las interacciones de dos torbellinos formándose hasta fusionarse en uno más grande

En este proceso de formación, el efecto conjunto de la colisión entre vórtices pequeños y el crecimiento de los resultantes de las colisiones va formando torbellinos de mayor tamaño de manera sucesiva, ralentizándose este proceso de crecimiento conforme aumenta el tamaño de las estructuras implicadas. En este caso, en los pasos temporales en los que se ha ejecutado el programa no se ha llegado a apreciar toda la vida de dicho vórtice, pero sí que se ha visto la secuencia de colisiones que se han dado a lo largo de su formación.

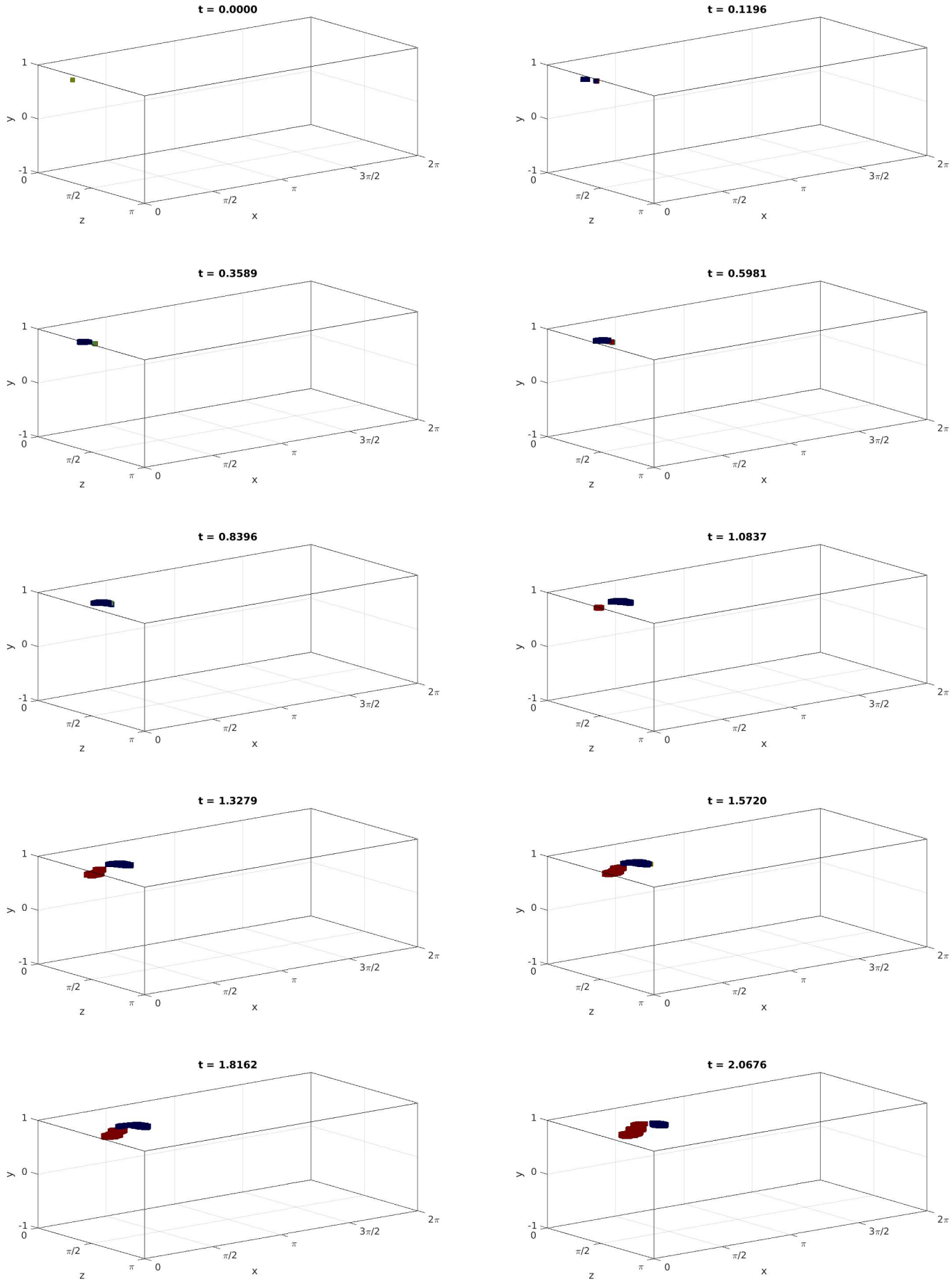


Figura 4.5: Representación del proceso de interacción de dos vórtices hasta su fusión.

4.3 Vida completa de un vórtice pequeño

Un tercer resultado que se ha considerado digno de mención es la vida completa de un vórtice pequeño, desde su nacimiento hasta su separación. La Figura 4.6 muestra el grafo que describe este proceso y la Figura 4.7 contiene la evolución de la distancia a la pared del centro de masas del vórtice y de su volumen.

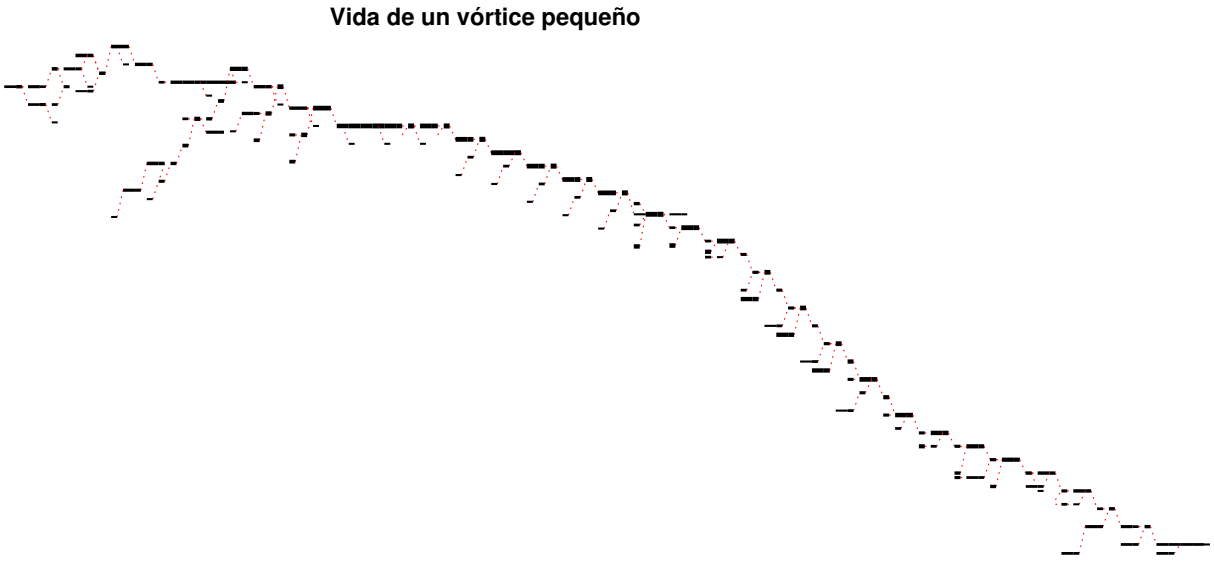


Figura 4.6: Grafo representativo de la vida completa de un vórtice pequeño.

Debido al pequeño número de pasos temporales calculados, solo se ha podido extraer la vida completa de los torbellinos con un proceso de crecimiento y disipación más rápido. Como obtuvieron Lozano-Durán y Jiménez [13], estos serán los más pequeños y cercanos a la pared.

Como se comprueba en la Figura 4.7, el vórtice se aleja progresivamente de la pared conforme va aumentando su volumen hasta que llega a su máximo tamaño, cuando empieza a disiparse también alejándose de la pared.

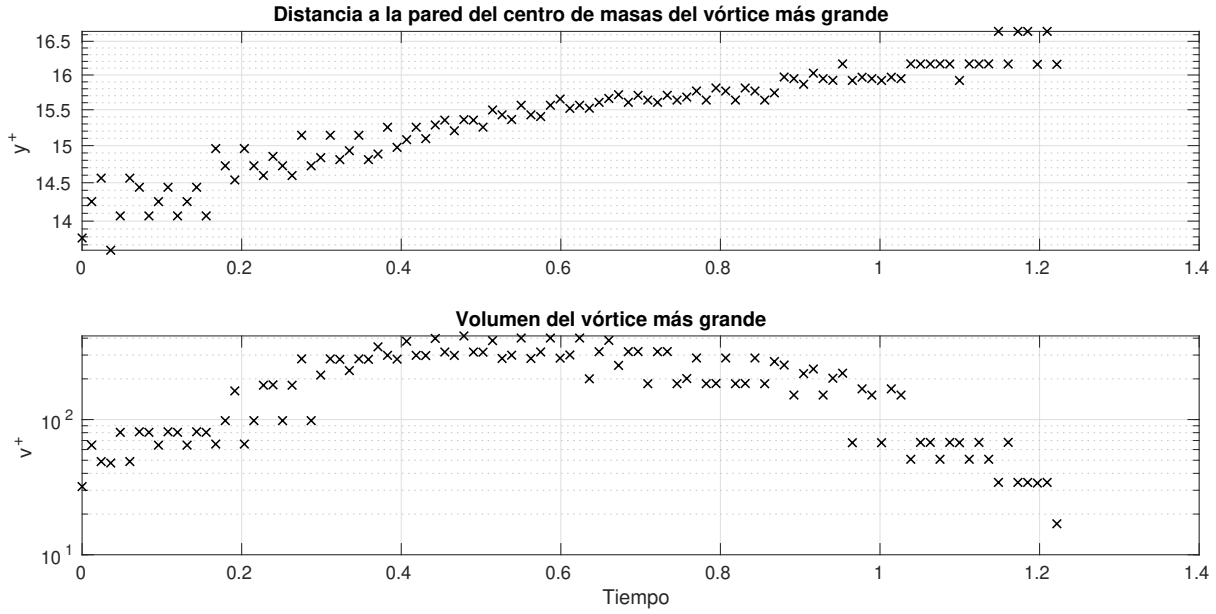


Figura 4.7: Distancia a la pared del centro de masas de la rama más grande del grafo de vida del vórtice (arriba) y su volumen (abajo).

4.4 Medidas de la simulación

Por último, se va a comentar brevemente una serie de datos agregados de los resultados extraídos de forma agregada. En la Figura 4.8 está representado el número total de vórtices en cada caso temporal, la fracción del volumen total que ocupan y la distribución de probabilidad de su número en los casos calculados.

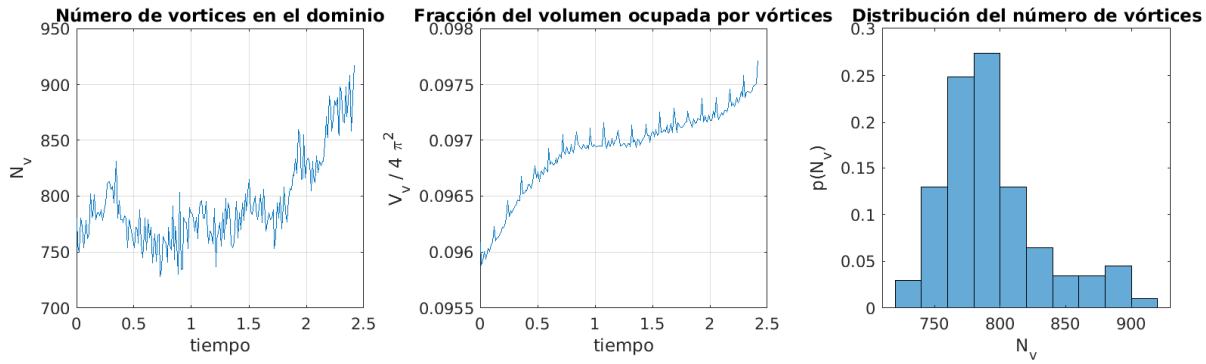


Figura 4.8: Número de vórtices en el dominio a lo largo del tiempo de simulación (izquierda), fracción del volumen total ocupada por éstos (centro) y distribución de probabilidad del número de vórtices en los casos calculados(derecha).

En estos pasos temporales lanzados se produce un aumento tanto de vórtices como de la fracción del volumen total que ocupan 4.8. Sin embargo, el análisis estadístico de las simulaciones empleadas indica que están en un estado estacionario, por lo que este

crecimiento puede resultar indicativo de fenómenos de frecuencia características más bajas que la duración de la base de datos obtenida. Se observa también un fenómeno de alta frecuencia claramente en la gráfica del centro, con picos de volumen a una frecuencia constante a lo largo de todo el tiempo estudiado. Sabiendo esto, los datos obtenidos son insuficientes para afirmar que la distribución del número de estructuras presentada en la gráfica de la derecha de la Figura 4.8 sea representativa del comportamiento estacionario del problema.

Capítulo 5

Conclusiones y trabajos futuros

5.1 Conclusiones

Partiendo de los objetivos planteados en §1.6, se pueden extraer una serie de conclusiones del desarrollo de este trabajo.

- Se ha demostrado que es posible aplicar una aproximación novedosa al problema de agregado tridimensional de vórtices que permite evitar uno de los problemas principales de los algoritmos tradicionales.
- Se ha desarrollado y validado un programa que agrupa con éxito las estructuras tridimensionales y sigue su evolución, generando satisfactoriamente una base de datos que contiene información sobre los vórtices presentes en el flujo estudiado, permitiendo realizar un seguimiento de nacimientos, muertes, colisiones y separaciones de vórtices a partir del cual estudiar la física de las estructuras turbulentas.
- Se ha desarrollado satisfactoriamente una serie de códigos con los que postprocesar y visualizar los datos obtenidos con el programa en Fortran, representando los resultados tanto mediante imágenes del campo como mediante grafos que representan las interacciones entre vórtices.
- Se ha estudiado un pequeño número de pasos temporales a bajo número de Reynolds, destacando una serie de casos por haberse encontrado relevantes.

Teniendo estos dos puntos en cuenta, podemos considerar por tanto que el presente trabajo ha cumplido los objetivos planteados.

5.2 Trabajos Futuros

Se ha desarrollado un código pensado para ser usado en el futuro para investigaciones de dinámica vorticial, extrayendo la física que hay detrás de los vórtices mediante los resultados generados por el programa. Dentro de este campo, se pueden destacar varios campos en los que el uso del programa desarrollado podría resultar interesante para el desarrollo científico del conocimiento de la turbulencia:

- El estudio de las estructuras turbulentas en flujos a números de Reynolds de pared altos, ya que la bibliografía actual está limitada a $Re_\tau = 4200$ en dominios pequeños y $Re_\tau = 550$ en dominios grandes.
- Las interacciones entre las estructuras térmicas presentes en canales con diferencia de temperatura, como las descritas por Gandia *et al* [14] y las estructuras turbulentas del fluido.
- Las diferencias en la dinámica vorticial de flujos de Couette y Poiseuille turbulento, debido a la diferencia del tamaño de las estructuras en estos dos casos.

Además, el código se ha desarrollado para ser lo más modular posible y poder ser expandido y adaptado al máximo número de problemas posible. Varias de las extensiones y mejoras posibles al código desarrollado son:

- Mayor optimización del código, para rebajar su coste computacional. Implementación de un algoritmo óptimo de agregado de componentes conexos de grafos como el de Chin *et al* [27].
- Aplicación de nuevos algoritmos para la comparación de estructuras entre pasos temporales, basados por ejemplo en redes neuronales.
- Adaptación del código para su uso en simulaciones LES.
- Paralelización del programa en MPI, para poder ejecutarlo en clusters de computación de memoria distribuida.
- Implementación de una interfaz DNS-Seguimiento para eliminar la necesidad de escritura de campos en disco duro.

En caso de que alguna de estas extensiones se lleve a cabo, esta memoria constituye un punto de partida para cualquiera de los trabajos descritos en la lista.

Capítulo 6

Pliego de condiciones y presupuesto

6.1 Pliego de condiciones

La normativa que regula el tipo de actividad realizada para minimizar los riesgos para el trabajador en este caso viene dada por el Real Decreto 488/1997 del 14 de abril, sobre disposiciones mínimas de seguridad y salud relativas al trabajo con equipos que incluyen pantallas de visualización (el constituido por un equipo con pantalla de visualización provisto, en su caso, de un teclado o dispositivo de adquisición de datos, de un programa para la interconexión persona-máquina, de accesorios ofimáticos y de un asiento y mesa o superficie de trabajo). Las cuatro variables atendidas para prever el tipo de riesgos a los que se enfrenta el trabajador, según esta normativa:

- Tiempo de trabajo con la pantalla de visualización.
- Tiempo de atención requerida ante la pantalla, que a su vez puede ser continua o discontinua.
- Exigencia y grado de complejidad de la tarea realizada ante la pantalla
- Necesidad de obtener una información de manera muy rápida.

Los riesgos presentes, según esta normativa, son:

- Seguridad (por contactos eléctricos)
- Higiene industrial (iluminación, ruido y condiciones termohigrométricas)
- Ergonomía (fatiga visual, física y mental)

Los entornos en los que se ha llevado a cabo la actividad desarrollada en este trabajo cumplen lo dispuesto en el Real Decreto 488/1997 del 14 de abril, específico para las condiciones de trabajo dadas, y en el Real Decreto 486/1997 del 14 de abril, sobre condiciones mínimas de seguridad y salud aplicables a los lugares de trabajo.

6.1.1 Recursos informáticos utilizados

Para el desarrollo de este trabajo ha sido necesario el uso de diversos recursos informáticos, tanto de software como de hardware.

Hardware

El grueso de este trabajo ha sido desarrollado en dos máquinas:

- Estación de trabajo: para implementar el algoritmo y para las tareas computacionalmente intensivas se ha usado una estación de trabajo con un procesador Intel Xeon E3.5-1220, 16GB de memoria RAM y almacenamiento en disco de estado sólido y en disco duro convencional.
- Ordenador portátil: para las tareas de soporte, documentación y redacción de la memoria se ha usado un ordenador portátil con un procesador Intel i7-6700HQ, 16GB de RAM y almacenamiento en disco duro convencional.

Software

En el desarrollo de este proyecto se ha optado preferentemente por el software libre en los casos en los que ha sido posible, contando las dos máquinas usadas con un sistema operativo GNU-Linux. Para la ejecución de los programas desarrollados en este trabajo se requiere del siguiente software:

- `gfortran` (libre) o `ifort` (privativo, comercial), como compiladores del código desarrollado en Fortran. Estos incluyen el estándar `OpenMP` de paralelización. El coste de adquisición de licencia de Intel Fortran es de 999USD, 901€ al cambio en el momento de redacción del presente trabajo.
- `HDF5` (libre), como interfaz de lectura/escritura de archivos de grandes dimensiones en disco.
- `FFTW3` (libre), necesario para las operaciones en transformada de Fourier en las simulaciones DNS.
- `openMPI` (libre), para la parelización en memoria distribuida de las simulaciones DNS.

Además del software necesario para la ejecución del código, para el desarrollo de este proyecto han sido necesarias las siguientes aplicaciones:

- `VScode` (privativo, gratuito) como IDE para la programación más fácil e intuitiva del código y como interfaz con el *debugger* de cada compilador.
- `MATLAB` (privativo, comercial) para el proceso de los datos y la generación de gráficos. El coste de una licencia anual es de 400€ en el momento de redacción.

6.2 Presupuesto

En esta sección se procederá a valorar económicamente el trabajo invertido en el desarrollo de este proyecto, teniendo en cuenta tanto el número de horas dedicadas al trabajo como los recursos materiales y licencias de software necesarias para su realización.

6.2.1 Relación de actividades: horas de trabajo

Las horas empleadas se pueden dividir en una serie de tareas principales para cuantificar y establecer un coste a cada una de las fases principales del proyecto.

1. **Revisión Bibliográfica y familiarización con las herramientas:** en esta primera fase se engloban el trabajo de familiarización con el estado del arte del estudio de la turbulencia y la familiarización con las herramientas ya existentes que servirán de base para el trabajo.

2. **Desarrollo conceptual del algoritmo:** en esta tarea se incluyen el desarrollo del algoritmo se de forma conceptual y su programación en serie en MATLAB.

3. **Programación en Fortran del algoritmo:** en esta fase se ha programado el algoritmo en Fortran, primero en serie y posteriormente incorporandole las directivas de paralelización. Por último se ha validado el programa desarrollado.

4. **Estudios de calibración:** esta etapa incluye los estudios realizados una vez desarrollado el grueso algoritmo, incluyendo los estudios de percolación y similaridad, cuyos resultados son necesarios para el correcto funcionamiento del programa.

5. **Generación de resultados:** esta tarea consiste en el lanzamiento de una simulación DNS y el lanzamiento del algoritmo sobre los resultados de ésta.

6. **Redacción de esta memoria:** este último punto incluye las horas dedicadas a la redacción de la presente memoria y la generación de las gráficas y diagramas que contiene.

Las horas dedicadas a cada una de estas tareas están presentadas en la Tabla 6.1, junto con su coste económico estimando suponiendo un precio de hora de ingeniero de 15 €/h. También se han incluido las horas aportadas por los tutores en el proyecto, doctores, asumiendo un coste horario de 30 €/h.

Tarea	Concepto	Número	Coste Unitario	Subtotal
Tarea 1	Horas de Ingeniero	50	15 €/h	750.00 €
	Horas de Doctor	10	30 €/h	300.00 €
Tarea 2	Horas de Ingeniero	200	15 €/h	3000.00 €
	Horas de Doctor	25	30 €/h	750.00 €
Tarea 3	Horas de Ingeniero	250	15 €/h	3750.00 €
	Horas de Doctor	30	30 €/h	900.00 €
Tarea 4	Horas de Ingeniero	70	15 €/h	1050.00 €
	Horas de Doctor	10	30 €/h	300.00 €
Tarea 5	Horas de Ingeniero	60	15 €/h	900.00 €
	Horas de Doctor	6	30 €/h	180.00 €
Tarea 6	Horas de Ingeniero	80	15 €/h	1200.00 €
	Horas de Doctor	10	30 €/h	300.00 €
Total	Horas de Ingeniero	710	15 €/h	10650.00 €
	Horas de Doctor	91	30 €/h	2730.00 €
			Total	**13380.00 €**

Tabla 6.1: Desglose de costes de las horas de trabajo empleadas en el desarrollo del presente proyecto.

6.2.2 Recursos materiales y licencias de software

El desglose de costes provenientes de recursos materiales y licencias de software viene dado en la Tabla 6.2. El precio especificado de las licencias de Software está actualizado en el momento de redacción de este trabajo, para las licencias anuales de cada uno de los productos descritos.

Categoría	Concepto	Número	Coste Unitario	Subtotal
Hardware	Estación de trabajo	1	1200 €	1200.00 €
	Ordenador Portátil	1	950 €	950.00 €
Licencias de software	MATLAB	2	400 €	800.00 €
	Intel Fortran	1	901 €	901.00 €
			Total	**3851.00 €**

Tabla 6.2: Desglose de costes de los recursos materiales y licencias de software empleados en el presente trabajo.

6.2.3 Coste total del proyecto

Sumando los costes por horas de trabajo y los costes materiales, añadiendo un 25 % de costes indirectos, un 6 % de beneficio industrial y el 21 % de IVA:

Categoría	**Subtotal**
Horas de trabajo	13380.00 €
Materiales y Software	3851.00 €
Gastos Indirectos	4307.75 €
Beneficio Industrial	1033.86 €
IVA	3618.51 €
Total	**26191.12 €**

Tabla 6.3: Cómputo del coste total del trabajo incluyendo gastos indirectos, beneficio industrial e IVA.

El coste total estimado de este trabajo es de **26191.12 €**.

Bibliografía

[1] O Reynolds. On the dynamical theory of incompressible viscous fluids and the determination of the criterion. In *Proceedings of the Royal Society-Mathematical and Physical Sciences*, volume 56, pages 40–45, 1894.

[2] Javier Jiménez. Near-wall turbulence. *Physics of Fluids*, 25(10):101302, 2013.

[3] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141, 1963.

[4] Stephen B. Pope. *Turbulent flows*. Cambridge University Press, 2000.

[5] Ivan Marusic, Jason P Monty, Marcus Hultmark, and Alexander J Smits. On the logarithmic region in wall turbulence. *Journal of Fluid Mechanics*, 716, 2013.

[6] Javier Jiménez. Computers and turbulence. *European Journal of Mechanics / B Fluids*, 79:1–11, 2020.

[7] John Kim, Parviz Moin, and Robert Moser. Turbulence statistics in fully developed channel flow at low reynolds number. *Journal of fluid mechanics*, 177:133–166, 1987.

[8] Sergio Hoyas, Martin Oberlack, Stefanie Kraheberger, and Francisco Alcantara-Avila. Turbulent channel flow at Reτ= 10000. In *Bulletin of the American Physical Society*, 2018.

[9] Nicholas Hutchins, Kapil Chauhan, Ivan Marusic, Jason Monty, and Joseph Klewicki. Towards reconciling the large-scale structure of turbulent boundary layers in the atmosphere and laboratory. *Boundary-layer meteorology*, 145(2):273–306, 2012.

[10] Stephen J Kline, William C Reynolds, FA Schraub, and PW Runstadler. The structure of turbulent boundary layers. *Journal of Fluid Mechanics*, 30(4):741–773, 1967.

[11] Javier Jiménez. Coherent structures in wall-bounded turbulence. *Journal of Fluid Mechanics*, 842, 2018.

[12] José I Cardesa, Alberto Vela-Martín, and Javier Jiménez. The turbulent cascade in five dimensions. *Science*, 357(6353):782–784, 2017.

[13] Adrián Lozano-Durán and Javier Jiménez. Time-resolved evolution of coherent structures in turbulent channels: characterization of eddies and cascades. *Journal of Fluid*

Mechanics, 759:432–471, 2014.

[14] Sergio Gandía-Barberá, Sergio Hoyas, Martin Oberlack, and Stefanie Kraheberger. The link between the reynolds shear stress and the large structures of turbulent couette-poiseuille flow. *Physics of Fluids*, 30(4):041702, 2018.

[15] Sergio Hoyas and Javier Jiménez. Scaling of the velocity fluctuations in turbulent channels up to Re τ= 2003. *Physics of fluids*, 18(1):011702, 2006.

[16] Sergio Hoyas and Javier Jiménez. Reynolds number effects on the reynolds-stress budgets in turbulent channels. *Physics of Fluids*, 20(10):101511, 2008.

[17] V. Avsarkisov, S. Hoyas, M. Oberlack, and J. P. García-Galache. Turbulent plane couette flow at moderately high reynolds number. *Journal of Fluid Mechanics*, 751:R1, 2014.

[18] Sanjiva K Lele. Compact finite difference schemes with spectral-like resolution. *Journal of computational physics*, 103(1):16–42, 1992.

[19] Jinhee Jeong and Fazle Hussain. On the identification of a vortex. *Journal of fluid mechanics*, 285:69–94, 1995.

[20] Brenden Epps. Review of vortex identification methods. In *55th AIAA Aerospace Sciences Meeting*, page 0989, 2017.

[21] Julian CR Hunt, Alan A Wray, and Parviz Moin. Eddies, streams, and convergence zones in turbulent flows. In *Studying Turbulence Using Numerical Simulation Databases, 2. Proceedings of the 1988 Summer Program*, pages 193–208, 1988.

[22] Yves Dubief and Franck Delcayre. On coherent-vortex identification in turbulence. *Journal of turbulence*, 1(1):011–011, 2000.

[23] Min S Chong, Anthony E Perry, and Brian J Cantwell. A general classification of three-dimensional flow fields. *Physics of Fluids A: Fluid Dynamics*, 2(5):765–777, 1990.

[24] Juan C Del Álamo, Javier Jimenez, Paulo Zandonade, and Robert D Moser. Self-similar vortex clusters in the turbulent logarithmic region. *Journal of Fluid Mechanics*, 561:329–358, 2006.

[25] Frank Harary. *Graph theory*. Addison-Wesley, 1995.

[26] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2 edition, 2001.

[27] Francis Y Chin, John Lam, and I-Ngo Chen. Efficient parallel algorithms for some graph problems. *Communications of the ACM*, 25(9):659–665, 1982.

[28] The HDF Group. Hierarchical Data Format, version 5. `http://www.hdfgroup.org/HDF5/`, 1997-2019.

[29] Margaret Wright. The interior-point revolution in optimization: history, recent developments, and lasting consequences. *Bulletin of the American mathematical society*, 42(1):39–56, 2005.