Máster en Ingeniería de Sistemas y de Control

Trabajo de Fin de Máster

Modelado del transporte de cianobacterias en aguas lénticas y planificación de trayectorias de USVs para la toma de muestras

Gonzalo Carazo Barbero

Septiembre 2020

Directores: Eva Besada Portas
José Antonio López Orozco

Máster en Ingeniería de Sistemas y de Control

Trabajo de Fin de Máster

Modelado del transporte de cianobacterias en aguas lénticas y planificación de trayectorias de USVs para la toma de muestras

Tipo A: Proyecto específico propuesto por un profesor

Gonzalo Carazo Barbero

Septiembre 2020

Directores: Eva Besada Portas
José Antonio López Orozco

Autorización

Autorizamos a la Universidad Complutense y a la Universidad Nacional de Educación a Distancia a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado: Gonzalo Carazo Barbero

Firma del alumno

Resumen

Monitorizar y predecir la localización de las floraciones de cianobacterias de forma automática y eficaz es de suma importancia, ya que estas pueden producir toxinas que ponen en riesgo la salud por el consumo y el uso recreativo del agua. Para lograrlo, este proyecto propone dirigir, de forma óptima y automática, a un vehículo de superficie hacia las zonas donde se prevé que haya una concentración de cianobacterias elevada. Con este fin, el proyecto se estructura en dos objetivos complementarios: i) la modelización matemática y simulación numérica del transporte de cianobacterias, y ii) la planificación y optimización de la trayectoria de vehículos autónomos de superficie encargados de la toma de muestras de estos contaminantes. En la primera parte, se utiliza un modelo tridimensional de transporte de partículas en fluidos, en mallas no estructuradas, que incluye las variaciones de densidad de las cianobacterias causadas por la producción de carbohidratos al recibir luz solar, que a su vez provocan un desplazamiento vertical de las colonias a lo largo del tiempo. La resolución del modelo se divide en varios pasos que hacen uso de diversas técnicas. La dinámica de fluidos, por su complejidad, se resuelve con software comercial (COMSOL Multiphysics, en particular), y sus resultados se utilizan para simular el resto de los componentes del modelo (p.e. los efectos de la luminosidad, la dinámica de la densidad de la colonia, etc.) en MATLAB con técnicas numéricas más sencillas, como la integración en tiempo mediante el método de Euler. La segunda parte del proyecto, la planificación de las trayectorias, consiste en la colocación en espacio y tiempo de los nodos de una spline de orden 4 que sirva como señal de referencia para situar y desplazar a un sensor ideal, embarcado en el vehículo, que deberá determinar la concentración de cianobacterias a lo largo de la trayectoria. Las splines, que precisan la posición tridimensional de la sonda a lo largo del tiempo, se obtienen mediante un proceso de optimización genética multi-objetivo que minimiza la duración y la longitud de la trayectoria y maximiza la concentración de cianobacterias a lo largo de la misma. Incluye restricciones que hacen que el recorrido de la spline esté contenido en un dominio tridimensional dado y evite zonas prohibidas. La velocidad y aceleración máxima también se restringen, de manera que la trayectoria pueda ser seguida por un vehículo autónomo real, y la duración de la misión se mantiene por debajo de un límite establecido. La memoria también recoge un conjunto de escenarios de simulación y de optimización sobre los que se ha analizado el funcionamiento de los métodos desarrollados.

Palabras clave: monitorización de cianobacterias, USV, modelado, simulación, fluidos, transporte, migración vertical, planificación de trayectorias, algoritmos genéticos, zonas prohibidas.

Abstract

Automatically and effectively monitoring and predicting the location of cyanobacterial blooms is of utmost importance, as these can produce toxins that put health at risk due to the consumption and recreational use of water. To achieve this, this project proposes to optimally and automatically direct a surface vehicle to the areas where a high concentration of cyanobacteria is expected. To this end, the project is structured around two complementary objectives: i) the mathematical modeling and numerical simulation of the transport of cyanobacteria, and ii) the planning and optimization of the trajectory of autonomous surface vehicles responsible for taking samples of these pollutants. In the first part, a three-dimensional model in unstructured meshes of the transport of particles in fluids is used, which includes the variations in density of cyanobacteria caused by the production of carbohydrates when receiving sunlight, which in turn cause a vertical displacement of the colonies over time. The resolution of the model is divided into several steps that make use of various techniques. Fluid dynamics, due to its complexity, is solved with commercial software (COMSOL Multiphysics, in particular), and its results are used to simulate the rest of the components of the model (i.e. the effects of luminosity, the dynamics of the density of colony, etc.) in MATLAB with simpler numerical techniques, such as time integration using Euler's method. The second part of the project, the planning of the trajectories, consists of the placement in space and time of the nodes of a spline of fourth order that serves as a reference signal to locate and move an ideal sensor, on board the vehicle, which you will need to determine the concentration of cyanobacteria along the path. The splines, which specify the three-dimensional position of the probe over time, are obtained through a multi-objective genetic optimization process that minimizes the duration and length of the path and maximizes the concentration of cyanobacteria along the path. It also includes constraints so that the path of the spline can be contained in a given three-dimensional domain and avoid forbidden areas. Top speed and acceleration are also restricted, so that the trajectory can be followed by a real autonomous vehicle, and the mission duration is kept below a set limit. The report also includes a set of simulation and optimization scenarios on which the performance of the developed methods has been analyzed.

Key words: cyanobacteria monitoring, USV, modeling, simulation, fluids, transport, vertical migration, trajectory planning, genetic algorithms, forbidden zones.

Índice General

Índice de Figuras

Índice de Tablas

Lista de Símbolos

A	Matriz conocida
a	Aceleración
$a_{\max}$	Aceleración máxima
B	Bloque matricial
$\boldsymbol{b}$	Vector conocido
b_φ	Coeficiente booleano relacionado con $\varphi \in \{v, a, t\}$
C	Matriz de coeficientes
C_ρ	Densidad celular mínima
c	Coeficiente
$\boldsymbol{c}$	Vector de coeficientes
c_φ	Coeficiente de importancia relacionado con $\varphi \in \{t, l, \rho, P\}$
d_p	Diámetro de la partícula
F	Función de la zona prohibida
$\boldsymbol{F}$	Fuerzas externas
f	Función real cualquiera
$\boldsymbol{g}$	Aceleración gravitatoria
H	Función escalón de Heaviside
H^1	Espacio de Sóbolev
I	Intensidad lumínica
$I_{\lim}$	Intensidad lumínica límite entre régimen luminoso y oscuro
I_s	Intensidad lumínica en la superficie
I_0	Límite de foto-inhibición celular
i, j, k, n	Índice del paso discretizado
$\boldsymbol{K}$	Variable auxiliar
K_i	Coeficiente de atenuación lumínica
L	Matriz triangular inferior
l	Longitud de la trayectoria
l_{inter}	Longitud de la intersección
m	Función test tal que $m \in H^1(\Omega)$
$\mathbb{N}$	Conjunto de los números naturales (incluyendo 0)
N_{inter}	Número de intersecciones
N_n	Número de nodos
N_p	Número de partículas

N_s	Número de muestreos
N_φ	Número de pasos en el eje $\varphi \in \{x, y, z, t, r\}$
$\boldsymbol{n}$	Vector normal
n_{cell}	Proporción volumétrica celular en la colonia
n_{gas}	Proporción volumétrica gaseosa en la colonia
n_V	Número de partículas contenidas en un volumen
P	Polinomio
p	Presión
$\boldsymbol{q}$	Vector de números aleatorios de $U(-1, 1)$
$\mathbb{R}$	Conjunto de los números reales
$\mathbb{R}^+$	Conjunto de los números reales positivos
$\mathbb{R}^-$	Conjunto de los números reales negativos
r	Radio
r_m	Radio de mutación
S	Spline de trayectoria del USV
S_a	Spline de aceleraciones
S_v	Spline de velocidades
$\boldsymbol{s}$	Vector de números aleatorios de $U(0, 1)$
s_{lim}	Probabilidad de mutación
T_{day}	Duración de un día
t	Coordenada temporal
t_{max}	Duración máxima permitida
$[t_0, T]$	Dominio temporal
U	Matriz triangular superior
$\boldsymbol{u}$	Velocidad del fluido
$\boldsymbol{u}^*$	Velocidad del fluido intermedia (compresible)
$\boldsymbol{u}_{\text{in}}$	Velocidad del fluido de entrada
$\boldsymbol{u}_p$	Velocidad de la partícula
V	Magnitud de un volumen
$\mathcal{V}$	Set volumétrico
v	Velocidad
v_{max}	Velocidad máxima
w	Función test tal que $w \in \left(H^1(\Omega)\right)^3$
$\boldsymbol{x} = (x, y, z)$	Coordenadas espaciales
$\boldsymbol{x}_p$	Posición de la partícula

$\boldsymbol{x}_p^0$	Posición inicial de la partícula
Z	Función objetivo
Z_P	Función de penalizaciones
α	Velocidad de decaimiento de la densidad celular
β	Factor normativo de la densidad celular en régimen luminoso
Γ_{bed}	Frontera inferior
Γ_{in}	Frontera de entrada
Γ_{out}	Frontera de salida
Γ_{top}	Frontera superior
γ	Tasa de cambio en la densidad celular en condiciones de radiación nula
$\Delta\varphi$	Tamaño del paso discretizado de la variable $\varphi \in \{x, y, z, t, \xi\}$
μ	Valor medio
ν	Viscosidad del fluido
ξ	Eje espacial cualquiera, $\xi \in \{x, y, z\}$
ρ_c	Densidad de la colonia
ρ_{cell}	Densidad celular
ρ_{cell}^0	Densidad celular inicial
ρ_f	Densidad del fluido
ρ_{muc}	Densidad del mucílago
ϱ_p	Concentración de partículas / Marcador de partícula vista
ϱ_S	Concentración de partículas a lo largo de la trayectoria
σ	Desviación estándar
τ	Instante de tiempo
φ	Variable real cualquiera
Ω	Dominio espacial

Lista de Abreviaturas

CAD	Diseño asistido por computador
CFD	Dinámica de fluidos computacional
EDO	Ecuación diferencial ordinaria
EDP	Ecuación en derivadas parciales
EF1 - EF2	Escenario de flujo 1 - 2
EO1 - EO6	Escenario de optimización 1 - 6
EP1 - EP3	Escenario de partículas 1 - 3
FDM	Método de las diferencias finitas
FEM	Método de los elementos finitos
FVM	Método de los volúmenes finitos
STL	Formato de archivo informático StereoLithography
USV	Vehículo autónomo de superficie

1. Introducción

Se comienza la memoria de este Trabajo de Fin de Máster presentando la motivación que lleva a la realización del trabajo y el estado del arte en problemas similares al aquí presentado. A continuación, se presentan los objetivos que se desean alcanzar, se describe el software utilizado a lo largo del proyecto y, finalmente, se describe la estructura que sigue la memoria del proyecto al completo.

1.1. Motivación

En 2010 se reconoció como esencial para la vida el derecho humano al agua potable limpia y al saneamiento [1]. Sin embargo, en la actualidad 2.1 billones de personas no tienen acceso al agua potable, 4.5 billones al saneamiento básico y 3.4 millones de personas mueren cada año por enfermedades como diarrea, cólera, disentería, fiebres tifoideas o poliomielitis, provocadas por el consumo de agua contaminada [2]. El agua potable es un bien escaso y se prevé que la situación se agrave debido principalmente al incremento del uso de agua, la contaminación, la mala gestión y el cambio climático, lo que puede provocar que en 2030 el mundo deba enfrentarse a un déficit global de agua del 40% [3].

En estas preciadas y escasas aguas habitan unas plantas llamadas cianobacterias; organismos microscópicos con pigmentos que les otorgan una coloración verde-azulada, por lo que también son conocidas como «algas verde-azules». En baja abundancia su presencia es muy importante para el ecosistema, ya que tienen la capacidad de generar oxígeno durante el proceso fotosintético, reciclar nutrientes, y captar el carbono y nitrógeno atmosférico del agua, al mismo tiempo que son fuente de alimento para zooplancton y bacterias.

Sin embargo, como consecuencia de diversas actividades humanas, las aguas eutróficas pueden verse enriquecidas, artificialmente y en exceso, con nutrientes, principalmente nitrógeno y fósforo, que son la fuente de alimento de las cianobacterias, lo cual provoca su desmedido y acelerado crecimiento que, sumado a otros factores como las altas temperaturas, la disponibilidad de luz y la baja renovación del agua, lleva a grandes acumulaciones de biomasa (o floraciones) [4], tal y como se puede ver en la Figura 1.

Figura 1: Floración de cianobacterias en aguas lénticas [5].

Dichas floraciones, también denominadas blooms, pueden agruparse, envueltas por una espesa masa gelatinosa llamada mucílago, y formar grandes colonias. Son fácilmente detectables, ya que tienen la posibilidad de flotar debido a las vacuolas de gas presentes en sus células, mientras la acción del viento y las corrientes las transporta y acumula en aguas lénticas. Otro factor importante en su transporte es la migración vertical, que se produce debido a cambios de densidad celular al acumular carbohidratos, lo cual está regulado por la cantidad de luz recibida. Son también las responsables de que el agua se vuelva turbia, adquiera un color verde, mal olor y sabor, de que se impida la penetración de la luz y oxígeno a las zonas profundas, y de que se provoque la muerte de la fauna acuática.

No obstante, la peor consecuencia de las colonias de cianobacterias es que algunas variedades producen una amplia gama de toxinas, denominadas cianotoxinas, que son nocivas para la salud del ser humano y de los animales, ya sea por ingestión, inhalación o contacto. Estas floraciones pueden estar formadas por una o diversas especies y no todas han de ser tóxicas. Incluso dentro de una misma variedad, puede haber decenas o cientos de cepas (subgrupos genéticamente específicos) con diferentes características y grados de toxicidad. Las más adversas para la salud, de acuerdo a [6], son:

- **Neurotoxinas**. Producidas principalmente por los géneros: Anabaena, Aphanizomenon, Oscillatoria, Trichodesmium, y Cylindrospermopsis, que interfieren en la transmisión de los impulsos nerviosos, tanto del sistema central como del periférico, y pueden provocar parálisis, fallos respiratorios y la muerte en un breve espacio de tiempo.
- **Hepatotoxinas**. En caso de ingestión, y dependiendo de la cantidad, pueden causar vómitos, diarreas, cefaleas, proliferación de células cancerígenas y la muerte en horas o pocos días. Afectan a las células y la función hepática, destacando las microcistinas y las nodularias. Ambas son las más comunes proliferaciones tóxicas de cianobacterias en aguas dulces y potables.
- **Citotoxinas**. Pueden causar daños a diversos órganos como hígado, corazón, riñones, estómago, sistema vascular y linfático y glándulas adrenales.
- **Dermatoxinas**. Aunque no son nocivas para la salud provocan irritaciones cutáneas por contacto.

Actualmente se ha convertido en una necesidad a nivel mundial la utilización de métodos de prevención, detección, y control de las cianobacterias tóxicas, a fin de evitar que alcancen las redes de distribución de agua potable. Para ello, es importante vigilar las aguas con riesgo de eutrofización, removerlas con turbinas para evitar que se remansen, controlar las concentraciones de nitrógeno, fósforo, y clorofila-a, etc. [7]. En caso de que se produzca floración, será imprescindible tomar muestras extraídas con red de la zona donde sea más evidente la acumulación. Esto permitirá identificar las cianobacterias tóxicas recogidas mediante botellas muestreadoras o bombas de succión para cuantificar la abundancia de células, nutrientes, fitoplacton y la cantidad de cianotoxinas, y proceder a su eliminación mediante diversos métodos como filtros de carbón activado y ozono, ultrasonidos, alguicidas, secuestradores de nutrientes, agentes bacterianos, etc.

En la actualidad, existen principalmente dos maneras de realizar un seguimiento a las colonias de cianobacterias. Por un lado, se contemplan las tomas de muestras periódicas, comúnmente realizados mediante la recolección de datos desde una red de boyas colocadas cerca de la salida de la masa de agua en cuestión [8]. Por otro lado, también se realizan seguimientos esporádicos que comúnmente se producen cuando se observa la floración a través de imágenes satelitales, en cuyo caso, la recolección de muestras se realiza de manera manual [4], aproximándose en lanchas a la zona afectada.

(a) Boya de muestreo periódico [9].

(b) Muestreo manual [10].

(c) USV de control de aguas [11].

Figura 2: Métodos de muestreo de cianobacterias.

Cabe mejorar estas estrategias mediante la utilización de vehículos autónomos de superficie (Unmanned Surface Vehicles, USV). Tales vehículos, puestos en práctica, serían capaces de reunir las mejores cualidades de ambas técnicas. Por una parte, su movilidad les permite tomar muestras en cualquier lugar de la masa de agua, factor que lo diferencia de las redes de alerta actuales. Por otra parte, al ser autónomos, no requieren que haya un operario directamente en el vehículo, razón por la cual pueden desplegarse varios USVs de manera simultánea y la función del operario sería la de supervisar la red de vehículos, bien de manera presencial o remota, incluso pudiendo controlar varios embalses al mismo tiempo. Esto haría posible la toma de muestras de manera automática y periódica, factor que lo diferencia de los métodos manuales actuales.

1.2. Objetivos

Se han definido varios objetivos para este proyecto. Al igual que el proyecto en su conjunto, los objetivos pueden dividirse en dos partes, los relacionados con la simulación numérica del transporte de cianobacterias y los relacionados con la optimización de trayectorias de un USV.

1.2.1. Objetivos del simulado numérico del transporte de cianobacterias

El objetivo final, como es evidente, es el de desarrollar una metodología que permita simular el transporte de las cianobacterias en una masa de agua léntica como puede ser un lago, una laguna, un pantano o un embalse. Para simplificar el problema, se asume que el flujo en dicha masa de agua es laminar y estacionario, hecho que se explora con más detalle en la Sección 2.1. También han de modelarse las dinámicas de las cianobacterias, o al menos, una aproximación de su comportamiento real.

Los objetivos secundarios en los que se articula esta meta son:

1. Estudiar los diferentes modelos de transporte de cianobacterias existentes en la literatura.
2. Estudiar los diferentes modelos de dinámica de fluidos existentes en la literatura.
3. Integrar ambos modelos para poder simular el transporte en un caso particular.
4. Diseñar una metodología capaz de poner en práctica esta simulación.
5. Comprobar el funcionamiento de la metodología utilizando varios escenarios ficticios.
6. Obtener un modelo implementable en casos reales.

1.2.2. Objetivos de la optimización de trayectorias de un USV

Basándose en los resultados de la parte anterior, los objetivos de esta parte se focalizan en la obtención de trayectorias que un USV sea capaz de poner en práctica.

Los objetivos secundarios de esta parte son:

1. Estudiar diferentes codificaciones de las trayectorias.
2. Diseñar una codificación adecuada para el problema.
3. Formular la optimización de las trayectorias del USV como un problema de optimización con múltiples objetivos y restricciones.
4. Implementar la codificación seleccionada y las funciones que evalúan los objetivos y las restricciones.
5. Implementar zonas prohibidas y limitaciones del dominio.
6. Diseñar un algoritmo de optimización capaz de proporcionar una solución al problema.
7. Comprobar el funcionamiento de la metodología utilizando varios escenarios ficticios.

1.3. Software

Se han empleado varias herramientas de software a lo largo del proyecto. En primer lugar, se ha utilizado la versión 2.82a de Blender [12] para el diseño asistido por ordenador (Computer Assisted Design, CAD) de la geometría de la masa de agua utilizada en este proyecto. Se trata de un software libre de código abierto que se usa principalmente para el modelado y la animación en 3D. Incluye la posibilidad de añadir no solo geometría, sino también iluminación, cámaras, shaders y hasta simulaciones básicas de ciertos fenómenos físicos, siempre orientados hacia la animación digital y no como herramienta científica. En este proyecto se ha empleado brevemente para modelar el dominio del agua y exportar la geometría a un formato aceptado por el resto de paquetes utilizados.

Para el modelado y simulación numérica del flujo de los fluidos en la masa de agua se ha utilizado COMSOL Multiphysics [13], que es un software comercial que implementa el simulado numérico mediante el método de elementos finitos de multitud de fenómenos físicos, incluida la dinámica de fluidos. Es un software muy versátil que es capaz de simular fenómenos multi-físicos acoplados en 0D, 1D, 2D, 3D e incluso en modelos con simetría axial o radial. En particular, para este proyecto se ha utilizado la versión 5.3.0.316 y el paquete de flujo laminar (spf), dentro del

conjunto de flujo monofásico, que se encuentra en el módulo de dinámica de fluidos computacional (Computational Fluid Dynamics, CFD).

Para el resto del proyecto se ha utilizado la versión R2018b (9.5.0.944444) de MATLAB [14], que es un software comercial de cómputo numérico con un lenguaje de programación integrado orientado al cálculo matricial. Todos los algoritmos implementados se han desarrollado en este software. Además, se ha empleado la versión 3.1 de su biblioteca de resolución de ecuaciones diferenciales parciales (PDE Toolbox) [15], y la versión 3.3 de la función `inpolyhedron` [16], que es una función proporcionada en el centro de intercambio de archivos (File Exchange) por Sven Holcombe.

Ya que en partes posteriores de esta memoria se hace referencia a tiempos de cómputo, se presentan en la Tabla 1 las características del ordenador utilizado en la totalidad de este proyecto.

Tabla 1: Características del ordenador.

Procesador	Intel Core i7-3632QM CPU @ 2.20 GHz
Memoria instalada (RAM)	8.00 GB
Arquitectura	Sistema Operativo 64 bits, procesador x64

1.4. Estructura de la memoria

Este Trabajo de Fin de Máster está dividido en dos partes bien diferenciadas. La primera, trata del modelado y la simulación del transporte de cianobacterias, y la segunda de la optimización de las trayectorias de un USV.

Se comienza, en el Capítulo 2, describiendo el modelo matemático utilizado para simular el transporte de cianobacterias. En primera instancia se desarrolla el modelo continuo diferencial, prestando atención a cada una de las ecuaciones y desarrollando las aproximaciones necesarias, para el caso particular que se aborda en este proyecto. A continuación, se describen las discretizaciones (tanto temporales como espaciales) empleadas para obtener el modelo discreto. Se describen también dos métodos de resolución de ecuaciones: los basados en diferencias finitas y los soportados por elementos finitos.

En el siguiente capítulo, el Capítulo 3, se describe la implementación del modelo y los resultados obtenidos. Se proporcionan los procedimientos específicos utilizados en su implementación, de manera general, de modo que puedan ser aplicadas a cualquier escenario, y se presentan los resultados de varios escenarios en los que se varía la posición inicial de las cianobacterias sobre el mismo lago ficticio.

El Capítulo 4 comienza la segunda parte del proyecto. Describe cada una de los componentes del problema de optimización de las trayectorias de un USV, incluyendo el desarrollo de la codificación de la trayectoria mediante splines 4-3-4 y el cálculo de cada uno de los objetivos y restricciones. Se presentan diversas maneras de implementar zonas prohibidas y se comparan diferentes métodos para encontrar y medir las intersecciones de la trayectoria candidata con dichas zonas.

De forma análoga al Capítulo 3, en el Capítulo 5 se describe la implementación específica del algoritmo genético, que incluye ciertas particularidades que lo diferencian de un algoritmo genético típico, y se presentan resultados para algunos escenarios simulados en el Capítulo 3. Para aumentar el número de casos disponibles, sobre un escenario de simulación particular se generan distintos escenarios de optimización variando el número, tamaño o posición de las zonas prohibidas, o el instante y lugar inicial y final de la misión de monitorización de las floraciones de cianobacterias.

Finalmente, en el Capítulo 6, se presentan las conclusiones obtenidas y los objetivos logrados. También se proponen algunas líneas de trabajo futuras relacionadas con este proyecto.

2. Modelo Matemático

La primera parte de este Trabajo de Fin de Máster consiste en la simulación del transporte de las colonias de cianobacterias en una masa de agua léntica. Los fenómenos físicos a menudo son demasiado complejos para ser descritos con fiabilidad absoluta, de modo que se utilizan modelos que aproximan el comportamiento real del sistema. Algunos de ellos, como el propuesto en este Trabajo, solamente son válidos si se cumplen ciertas condiciones, como es en este caso el flujo laminar y estacionario, de modo que es imperativo tener cuidado al seleccionar una descripción apropiada para cada caso particular.

En lo relativo a la dinámica de las cianobacterias, el modelo desarrollado en este proyecto tiene en cuenta la influencia del flujo del fluido (laminar y estacionario), las variaciones en la densidad celular producidas por diferencias en la intensidad lumínica, y la consecuente migración vertical producida por las diferencias de densidad. Sin embargo, no contempla variaciones en la concentración de biomasa en las colonias, que se consideran como partículas, ni la difusión producida por la multiplicación celular.

Los modelos matemáticos de fenómenos físicos complejos normalmente dependen de una formulación diferencial cuya solución exacta no es conocida, de modo que se recurre a las soluciones numéricas para hallar aproximaciones a esta solución exacta ideal. Este capítulo se divide en dos partes. En primer lugar, se formula el modelo matemático que describe el transporte de cianobacterias en un medio continuo, desarrollando cada parte del mismo. A continuación, se formula el mismo modelo discretizado de forma que pueda ser resuelto mediante métodos numéricos.

2.1. Modelo continuo

Como se ha mencionado con anterioridad, el objetivo de la primera parte del proyecto es realizar un modelado matemático y una simulación numérica del transporte de cianobacterias a lo largo del tiempo. Para simplificar en cierto modo el problema, se considera que el flujo en un lago con una entrada y una salida se encuentra en régimen estacionario y laminar, es decir, que no existen variaciones del flujo en el tiempo y que el efecto de las turbulencias se asume despreciable. El asumir que dicho efecto es despreciable permite construir un modelo en el cual el flujo es estacionario, lo que simplifica las ecuaciones del transporte de partículas. La diferencia entre ambos tipos de flujos se muestra en la Figura 3.

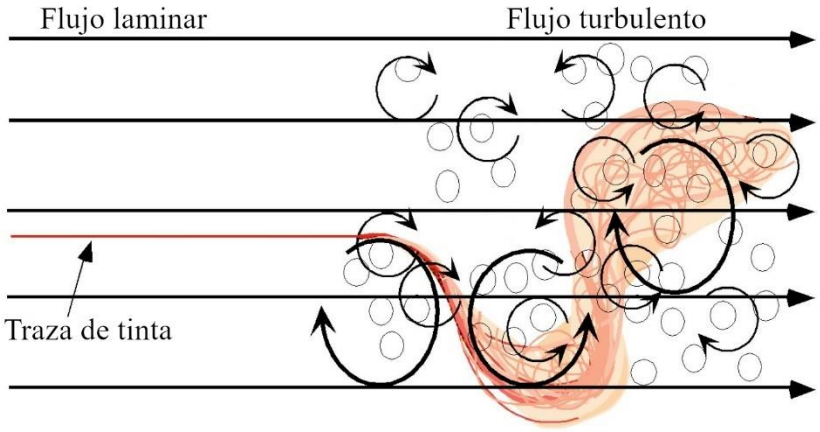


Figura 3: Flujo laminar (izquierda) vs. flujo turbulento (derecha) [17].

El modelo matemático utilizado se trata de un caso particular del presentado en el tercer capítulo de la Tesis Doctoral de Evelyn Aparicio Medrano [18], en particular del modelo dimensional, por lo que se comenzará introduciendo cada una de las ecuaciones aplicadas en el dominio espacial tridimensional $\Omega \in \mathbb{R}^3$. El problema puede dividirse en varios subproblemas ya que el acoplamiento entre las partes es unidireccional. Los subproblemas a resolver son: la dinámica de fluidos, la intensidad lumínica, la densidad de la colonia, y finalmente el transporte de partículas.

2.1.1. Dinámica de fluidos

En primer lugar, se ha de resolver la dinámica del agua del lago, que es una de las principales fuerzas que influyen en el transporte de las colonias de cianobacterias. Dado que el agua es un fluido incompresible, se aplican las ecuaciones de Navier-Stokes para el caso incompresible, que son

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\frac{1}{\rho_f}\nabla p + \boldsymbol{F} + \nu \nabla^2 \boldsymbol{u}\,, \tag{1}$$

$$\nabla \cdot \boldsymbol{u} = 0\,, \tag{2}$$

donde $\boldsymbol{u} \in \mathbb{R}^3$ es un campo vectorial tridimensional que describe la velocidad del fluido en coordenadas Eulerianas y se trata de la primera de las variables dependientes; $p \in \mathbb{R}^+$ es la presión como campo escalar y es la segunda variable dependiente; ρ_f y $\nu \in \mathbb{R}^+$ son la densidad y la viscosidad cinemática del fluido respectivamente; y $\boldsymbol{F} \in \mathbb{R}^3$ es el vector de fuerzas externas. La variable de interés es la velocidad en particular, ya que es esta la que influye directamente en la trayectoria de las partículas sumergidas en el fluido. La primera de estas ecuaciones es la ecuación de balance del momento, que no impone que el flujo sea incompresible como ha de serlo en el caso del agua, de modo que se añade la segunda ecuación de continuidad, que se deriva del principio de conservación de la masa.

2.1.2. Intensidad lumínica

Como se explica en el siguiente apartado, la cantidad de luz recibida por las cianobacterias influye en la densidad de estas, que a su vez hace que se incremente o reduzca su flotabilidad y se produzca una migración vertical. A medida que una onda electromagnética como la luz viaja a través de una columna de agua, esta es absorbida y su intensidad va disminuyendo. Para describir este fenómeno se utiliza la ley de Beer-Lambert, que describe cómo la intensidad disminuye de forma exponencial con la profundidad de acuerdo a

$$I = I_s \exp(K_i z)\,, \tag{3}$$

donde $I_s \in \mathbb{R}^+$ es la intensidad en la superficie, $K_i \in \mathbb{R}^+$ es el coeficiente de atenuación lumínica, y $I \in \mathbb{R}^+$ es la intensidad lumínica a una profundidad $z \in \mathbb{R}^-$ de acuerdo a los ejes establecidos en la Figura 4. Esta figura muestra el dominio del fluido, y es de notar que el eje z se orienta en vertical hacia arriba, teniendo su origen a la altura de la superficie del lago. Esto representa una diferencia con respecto a otros modelos como el de Medrano [18], donde el eje z se orienta en el sentido opuesto.

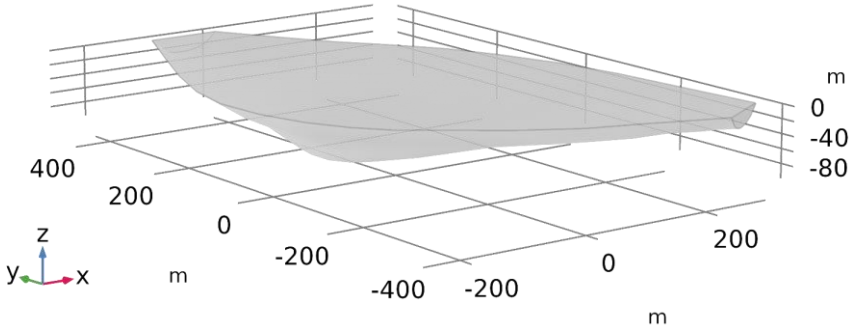


Figura 4: Dominio geométrico.

En la Figura 5 se muestra la propagación de la intensidad lumínica de acuerdo a la ley de Beer-Lambert a diferentes intensidades y profundidades. En la parte de arriba se muestra la irradianza en la superficie del lago durante un periodo de 48 h. Bajo esta, la propagación de la luz a diferentes profundidades y a lo largo del tiempo, y a la derecha el decaimiento de la intensidad a medida que se incrementa la profundidad. Las líneas rojas discontinuas marcan el límite de profundidad donde se realiza el cambio de régimen, de luminoso a oscuro. La variación de la intensidad lumínica con la profundidad a lo largo del día hará que la colonia de cianobacterias se sitúe a distintas profundidades a lo largo del mismo. Los parámetros utilizados en esta gráfica se toman de la Tabla 2, que se presenta en la Sección 3.1.2.

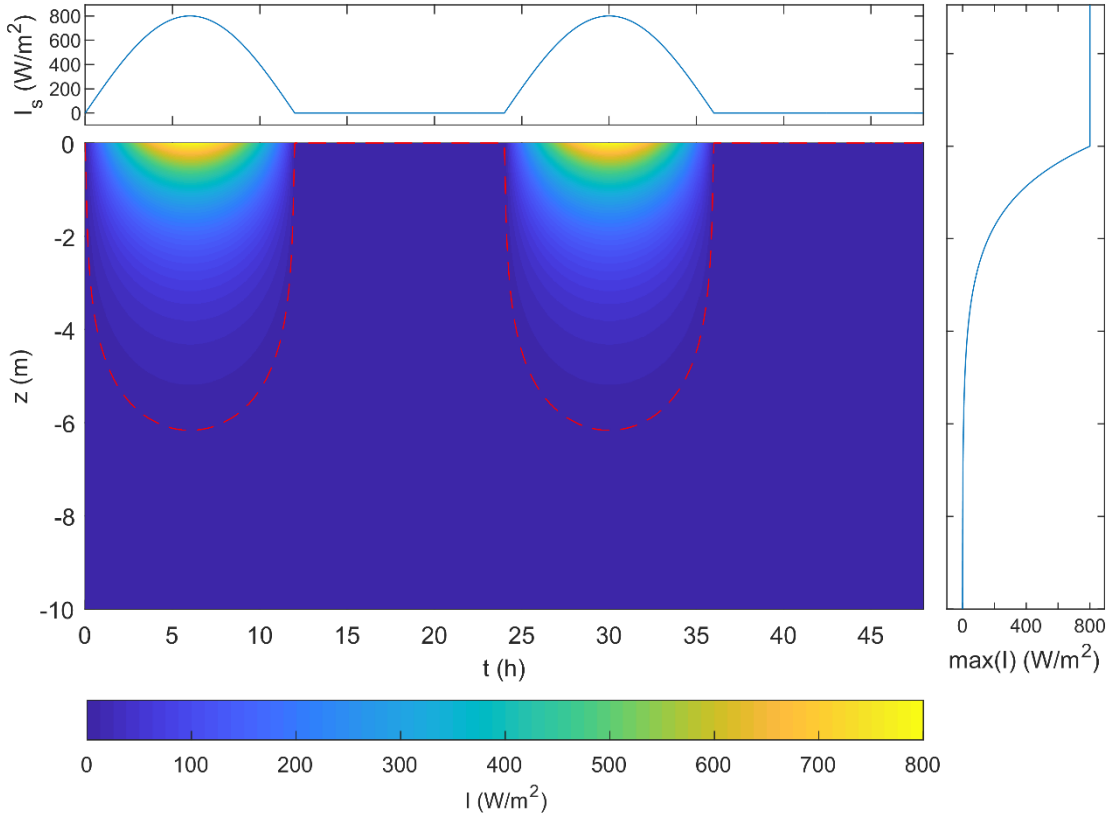


Figura 5: Propagación de la intensidad lumínica.

2.1.3. Densidad de la colonia

El cálculo de la densidad de la colonia se realiza en dos pasos. Primero se obtiene la densidad de las células y, a continuación, se une a la densidad (conocida) del mucílago que las rodea para obtener la densidad total de la colonia. La densidad de una célula de cianobacteria depende de la cantidad de luz absorbida. Bajo una fuente de luz suficientemente intensa, la célula produce y almacena carbohidratos, que la hacen más densa, mientras que en la oscuridad o entornos con fuentes de luz atenuadas estos carbohidratos son consumidos y la densidad de la célula disminuye. Se diferencia en el modelo cada uno de estos regímenes luminosos, que quedan limitados por el parámetro de intensidad lumínica $I_{\text{lim}} \in \mathbb{R}^+$.

Por una parte, en régimen luminoso ($I > I_{\text{lim}}$) la densidad celular queda descrita de acuerdo a

$$\frac{d\rho_{\text{cell}}}{dt} = \beta I \exp(-I/I_0) + \gamma\,, \tag{4}$$

donde $\rho_{\text{cell}} \in \mathbb{R}^+$ es la densidad celular y β, γ e $I_0 \in \mathbb{R}$ son parámetros biológicos: el factor normativo, la tasa de cambio en la densidad en condiciones de radiación nula ($I = 0$), y el límite de foto-inhibición de las células respectivamente. La intensidad lumínica I se obtiene de la fórmula del anterior apartado utilizando la profundidad a la que se encuentra la célula como variable independiente. Obsérvese en la Figura 6(a), que muestra la tasa de variación en la densidad celular con respecto a la intensidad lumínica, que al incrementar la cantidad de luz recibida la densidad de la célula aumenta, ya que los valores de la tasa de crecimiento son positivos; pero al sobrepasar el límite marcado con la línea discontinua roja, entra en juego el efecto de la foto-inhibición y la tasa de aumento disminuye hasta llegar a ser negativa. Los parámetros utilizados en esta gráfica se toman de la Tabla 2, que se presenta en la Sección 3.1.2.

Por otra parte, en régimen oscuro ($I \leq I_{\text{lim}}$) la ecuación que modela la densidad es

$$\frac{d\rho_{cell}}{dt} = -\alpha\big(\rho_{\text{cell}} - C_\rho\big)\, H\big(\rho_{\text{cell}} - C_\rho\big)\,, \tag{5}$$

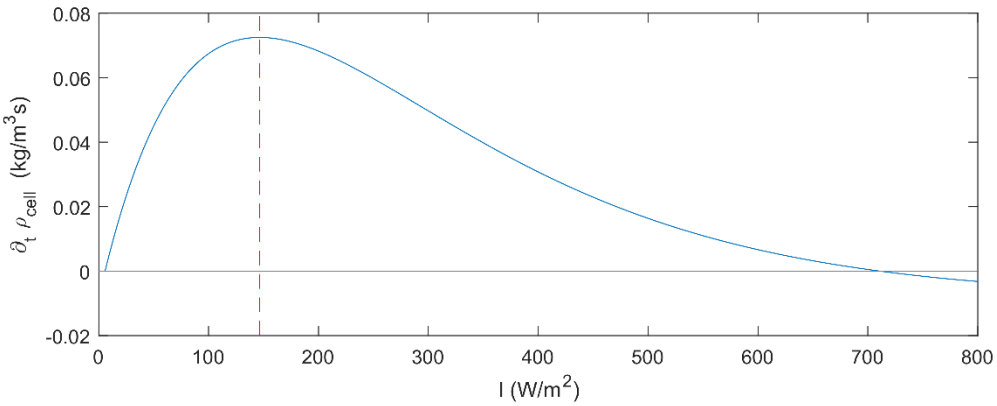
donde $\alpha \in \mathbb{R}$ es el parámetro biológico de la velocidad de decaimiento de la densidad, $C_\rho \in \mathbb{R}^+$ es la densidad mínima de la célula y H es la función escalón de Heaviside definida como

$$H(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases}, \qquad \forall x \in \mathbb{R}\,. \tag{6}$$

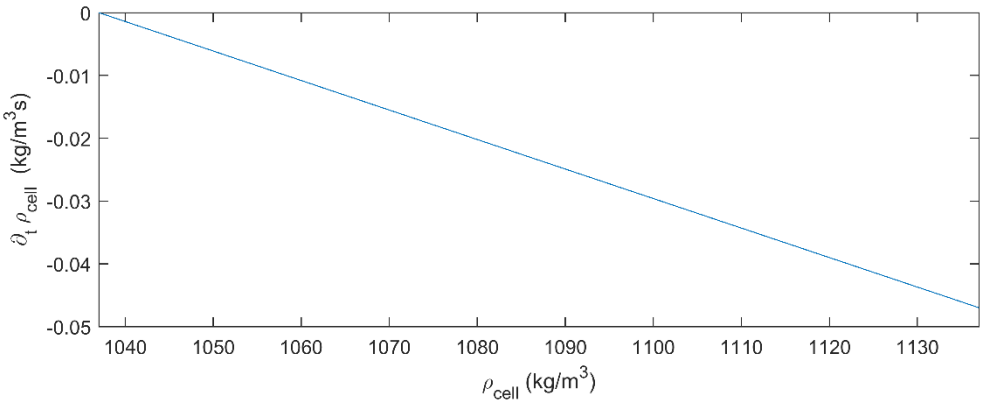
Las variaciones de la densidad celular con respecto a sí misma (no con respecto a la intensidad lumínica como en el caso anterior) se muestran en la Figura 6(b). Además, para completar el modelo, hay que tener en cuenta que las cianobacterias no existen en aislamiento, sino que forman colonias que se transportan como una unidad. Las células contienen vesículas gaseosas y están rodeadas de una sustancia viscosa llamada mucílago, de modo que la densidad efectiva que ha de ser considerada para hallar la flotabilidad de las cianobacterias es la de la colonia al completo, no la de la célula individual. Estas dos cantidades se relacionan se acuerdo a

$$\rho_c = \rho_{\text{cell}} n_{\text{cell}}\big(1 - n_{\text{gas}}\big) + \rho_{\text{muc}}(1 - n_{\text{cell}})\,, \tag{7}$$

donde ρ_c y $\rho_{\text{muc}} \in \mathbb{R}^+$ son las densidades de la colonia y el mucílago respectivamente, y n_{cell} y $n_{\text{gas}} \in [0,1]$ son las proporciones de células y gas con respecto al total de la colonia en volumen.



(a) Régimen luminoso.



(b) Régimen oscuro.

Figura 6: Variación de la densidad celular.

2.1.4. Transporte de partículas

Los resultados anteriores son necesarios para poder calcular el transporte de las colonias a lo largo del tiempo. En especial, es necesario conocer la velocidad del fluido $\boldsymbol{u}$, y la densidad de la colonia ρ_c. La Tesis de Evelyn Aparicio Medrano [18] utiliza un caso particular de las ecuaciones de transporte de esferas propuestas por Maxey y Riley [19], con una formulación en coordenadas Lagrangianas. La ecuación del transporte en la Tesis es

$$\left(1+\frac{1}{2}\frac{\rho_f}{\rho_c}\right)\frac{d\boldsymbol{u}_p}{dt}=\left(1-\frac{\rho_f}{\rho_c}\right)\boldsymbol{g}+\frac{18\nu\,\rho_f}{d_p^2\,\rho_c}\left(\boldsymbol{u}-\boldsymbol{u}_p\right)+\frac{3}{2}\frac{\rho_f}{\rho_c}\frac{D\boldsymbol{u}}{Dt}, \tag{8}$$

donde $\boldsymbol{u}_p \in \mathbb{R}^3$ es el vector de velocidades de las partículas, que no necesariamente ha de ser igual a la velocidad del fluido donde se hallan sumergidas, $\boldsymbol{g} \in \mathbb{R}^3$ es el vector de la aceleración gravitatoria, siguiendo la orientación de los ejes establecidos en la Figura 4, y $d_p \in \mathbb{R}^+$ es el diámetro de las partículas. Además, al tratarse de una formulación en coordenadas Lagrangianas, se definen los siguientes operadores:

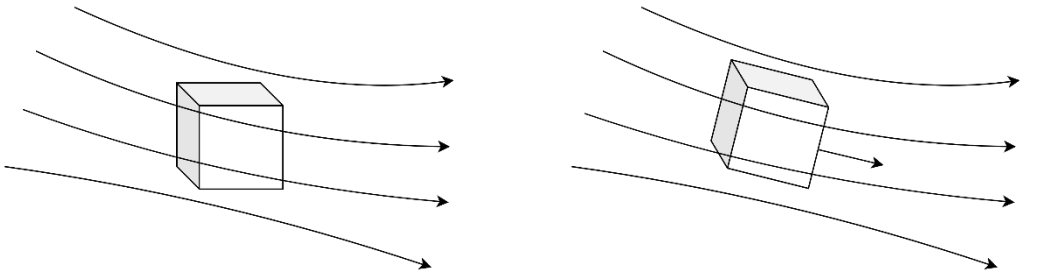
$$\frac{d}{dt}=\frac{\partial}{\partial t}+\left(\boldsymbol{u}_p\cdot\nabla\right), \tag{9}$$

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + (\boldsymbol{u} \cdot \nabla)\,, \tag{10}$$

que se corresponden con la derivada temporal a lo largo de la trayectoria de la partícula y la derivada material. La razón por la que Medrano utiliza esta formulación es que el flujo de su Tesis depende del tiempo. Sin embargo, en este Trabajo de Fin de Máster se considera solamente flujo en régimen estacionario (o lo que es lo mismo, invariante en tiempo), de modo que resulta más sencillo abordar el problema desde un marco Euleriano. De esta manera, (8) puede ser escrita como

$$\left(1 + \frac{1}{2}\frac{\rho_f}{\rho_c}\right)\frac{\partial \boldsymbol{u}_p}{\partial t} = \left(1 - \frac{\rho_f}{\rho_c}\right)\boldsymbol{g} + \frac{18\nu\,\rho_f}{d_p^2\,\rho_c}\left(\boldsymbol{u} - \boldsymbol{u}_p\right) + \frac{3}{2}\frac{\rho_f}{\rho_c}\frac{\partial \boldsymbol{u}}{\partial t}\,. \tag{11}$$

Las diferencias entre las coordenadas Eulerianas y Lagrangianas se muestran en la Figura 7.



(a) Coordenadas Eulerianas. El elemento es estático.

(b) Coordenadas Lagrangianas. El elemento se mueve con el fluido.

Figura 7: Coordenadas Eulerianas y Lagrangianas.

Como consecuencia de utilizar flujo estacionario y coordenadas Eulerianas, la derivada temporal del flujo en cualquier punto del dominio es nula. Al haber elegido este tipo de coordenadas, es forzoso interpretar $\boldsymbol{u}_p$ de la misma manera que $\boldsymbol{u}$, como un campo de velocidades (tetradimensional en este caso, incluyendo la densidad celular como dimensión) definido en puntos estáticos del dominio, de modo que las derivadas temporales se cancelan:

$$0 = \left(1 - \frac{\rho_f}{\rho_c}\right)\boldsymbol{g} + \frac{18\nu\,\rho_f}{d_p^2\,\rho_c}\left(\boldsymbol{u} - \boldsymbol{u}_p\right). \tag{12}$$

Reordenando la ecuación anterior, se obtiene finalmente la expresión que describe el campo de velocidades de las partículas:

$$\boldsymbol{u}_p = \boldsymbol{u} + \frac{d_p^2}{18\nu}\left(\frac{\rho_c}{\rho_f} - 1\right)\boldsymbol{g}. \tag{13}$$

En esta expresión, el primer término a la derecha de la igualdad representa el campo de velocidades del fluido, mientras que el segundo representa la velocidad inducida por la flotabilidad de la colonia, que depende de su densidad. Para hallar la posición de las partículas $\boldsymbol{x}_p \in \mathbb{R}^3$, basta con tomar sus posiciones iniciales $\boldsymbol{x}_p^0 \in \mathbb{R}^3$ tal que $\boldsymbol{x}_p(t_0,\ \cdot\ ) = \boldsymbol{x}_p^0$ e integrar el campo de velocidades $\boldsymbol{u}_p$ en tiempo, ya que

$$\frac{\partial \boldsymbol{x}_p}{\partial t} = \boldsymbol{u}_p \,, \tag{14}$$

o, tomando la integral de ambos lados,

$$\boldsymbol{x}_p(t) = \boldsymbol{x}_p^0 + \int_{t_0}^{t} \boldsymbol{u}_p \, d\tau \,, \tag{15}$$

teniendo en cuenta que las coordenadas espaciales de integración irán variando a lo largo del tiempo, que es lo que produce las variaciones en el campo de velocidades, que de otra manera es estacionario.

2.1.5. Modelo completo

Antes de unir todas las ecuaciones anteriores y presentar el sistema completo es necesario establecer el dominio, las condiciones de contorno y las condiciones iniciales.

El dominio espacial Ω sobre el que se trabaja es tridimensional y arbitrario. Se ha definido mediante un archivo STL (formato de archivo informático, StereoLithography) que representa el volumen de agua de un lago ficticio, con una profundidad máxima de 100 m y una superficie de 3.17×10^5 m^2.

El contorno del dominio Ω se puede dividir en los cuatro sub-contornos que se muestran en la Figura 8: Γ_{top}, que se trata de una superficie deslizante tanto para el flujo como para las partículas que representa la superficie del lago; Γ_{bed}, que es una superficie sin deslizamiento en la frontera que forma el lecho del lago; Γ_{in}, que es el plano vertical por donde se produce la entrada de agua a velocidad constante; y Γ_{out}, que es la salida a presión relativa nula.

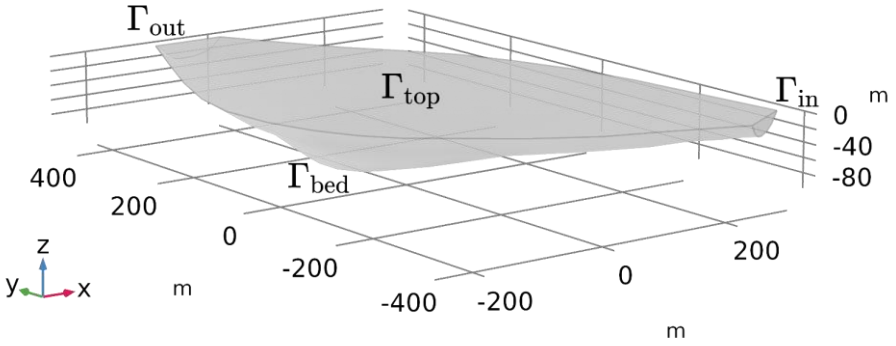


Figura 8: Contornos de la geometría.

Las condiciones iniciales para la resolución de las ecuaciones del sistema son simplemente las de las dos variables dependientes del tiempo: $\boldsymbol{x}_p(t_0) = \boldsymbol{x}_p^0$ y $\rho_{\text{cell}}(t_0) = \rho_{\text{cell}}^0$. Por lo tanto, el modelo completo a utilizar es el siguiente:

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\frac{1}{\rho_f}\nabla p + \boldsymbol{F} + \nu \, \nabla^2 \boldsymbol{u} \,, \qquad \text{en } \Omega \tag{16}$$

$$\nabla \cdot \boldsymbol{u} = 0 \,, \qquad \text{en } \Omega \tag{17}$$

$$I = I_s \exp(K_i z)\,, \qquad \text{en } \Omega \times [t_0, T] \tag{18}$$

$$\frac{d\rho_{\text{cell}}}{dt} = \beta I \exp(-I/I_0) + \gamma\,, \qquad \text{en } \Omega \times [t_0, T]\,,\ I > I_{\text{lim}} \tag{19}$$

$$\frac{d\rho_{cell}}{dt} = -\alpha\big(\rho_{\text{cell}} - C_\rho\big)\, H\big(\rho_{\text{cell}} - C_\rho\big)\,, \qquad \text{en } \Omega \times [t_0, T]\,,\ I \le I_{\text{lim}} \tag{20}$$

$$\rho_c = \rho_{\text{cell}} n_{\text{cell}}\big(1 - n_{\text{gas}}\big) + \rho_{\text{muc}}(1 - n_{\text{cell}})\,, \qquad \text{en } \Omega \times [t_0, T] \tag{21}$$

$$\boldsymbol{u}_p = \boldsymbol{u} + \frac{d_p^2}{18\nu}\left(\frac{\rho_c}{\rho_f} - 1\right)\boldsymbol{g}\,, \qquad \text{en } \Omega \tag{22}$$

$$\frac{\partial \boldsymbol{x}_p}{\partial \mathrm{t}} = \boldsymbol{u}_p\,, \qquad \text{en } \Omega \times [t_0, T] \tag{23}$$

$$\boldsymbol{u} \cdot \boldsymbol{n} = 0\,, \qquad \text{en } \Gamma_{\text{top}} \tag{24}$$

$$\boldsymbol{K} - (\boldsymbol{K} \cdot \boldsymbol{n})\boldsymbol{n} = 0\,, \qquad \text{en } \Gamma_{\text{top}} \tag{25}$$

$$\boldsymbol{u}_p \cdot \boldsymbol{n} = 0\,, \qquad \text{en } \Gamma_{\text{top}} \times [t_0, T] \tag{26}$$

$$\boldsymbol{u} = 0\,, \qquad \text{en } \Gamma_{\text{bed}} \tag{27}$$

$$\boldsymbol{u}_p = 0\,, \qquad \text{en } \Gamma_{\text{bed}} \times [t_0, T] \tag{28}$$

$$\boldsymbol{u} \cdot \boldsymbol{n} = \boldsymbol{u}_{\text{in}}\,, \qquad \text{en } \Gamma_{\text{in}} \tag{29}$$

$$p = 0\,, \qquad \text{en } \Gamma_{\text{out}} \tag{30}$$

siendo $\boldsymbol{n}$ el vector normal a la superficie del contorno y que apunta hacia el exterior, y $\boldsymbol{K}$ una variable auxiliar definida como

$$\boldsymbol{K} = [\nu(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T)]\boldsymbol{n}\,. \tag{31}$$

Examinando el modelo es posible establecer un orden en el cual las ecuaciones han de ser resueltas. El objetivo es determinar la posición $\boldsymbol{x}_p$ de las partículas (cianobacterias) a lo largo del tiempo mediante la ecuación (23). Para ello es necesario obtener el campo de velocidades $\boldsymbol{u}_p$ dado por la ecuación (22) y las condiciones de contorno (26) y (28). A su vez, para resolver esto, es imperativo disponer, por un lado, del campo de velocidades del fluido $\boldsymbol{u}$ (de ecuaciones (16) y (17) con condiciones de contorno (24), (25), (27), (29) y (30)), y, por otro, de la densidad de la colonia ρ_c (21), que se calcula con la densidad celular ρ_{cell} de las ecuaciones (19) y (20), la primera de las cuales necesita el resultado de la intensidad lumínica I en la posición de interés dada por (18).

Además, a excepción del bucle de realimentación de la posición de las partículas a la intensidad lumínica, el resto del modelo está acoplado unidireccionalmente. Esto quiere decir que los efectos del transporte, como puede ser un cierto desplazamiento del volumen del fluido, se consideran despreciables. El hecho de que se disponga de un modelo como este, favorece la utilización de un esquema de integración temporal explícito. La relación entre todas las ecuaciones se resume en la Figura 9.

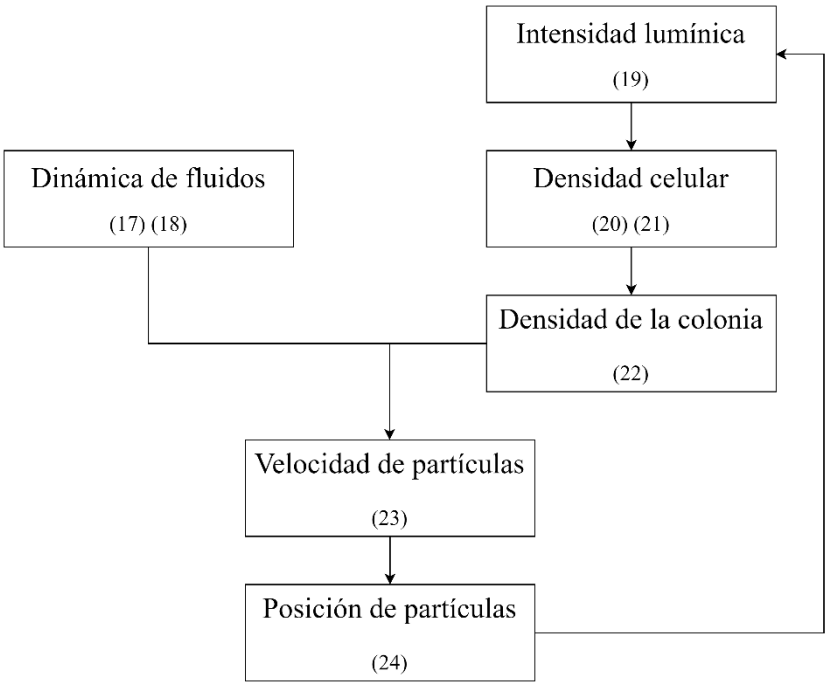


Figura 9: Diagrama de acoplamiento del modelo matemático.

2.2. Modelo discreto

Debido a la complejidad del modelo, especialmente de la parte de dinámica de fluidos, resulta imposible hallar una solución exacta, de modo que ha de recurrirse a métodos numéricos para resolver el problema.

Más concretamente, algunos módulos del modelo son simplemente expresiones algebraicas que pueden ser resueltas con facilidad, en otros casos una integración temporal explícita es suficiente, y finalmente en el caso de la dinámica de fluidos su complejidad hace que sea necesario el uso de técnicas más avanzadas, como pueden ser los métodos numéricos. A continuación, se describe la discretización usada y los dos métodos numéricos utilizados en este proyecto: el método de diferencias finitas (Finite Difference Method, FDM) y el método de elementos finitos (Finite Element Method, FEM).

2.2.1. Discretización

En ciertos casos, para poder calcular la solución del módulo, ha de realizarse una discretización espacial y/o temporal, consistente en dividir el dominio continuo en pequeñas unidades donde ciertos campos puedan considerarse constantes, de manera que la aplicación de las ecuaciones se vea simplificada. Normalmente esto consigue transformar el sistema de ecuaciones diferenciales a un sistema algebraico de mayores dimensiones, y algunas veces en un sistema lineal. Al haber modificado el marco en el cual se describe el transporte de partículas y haber aplicado simplificaciones sobre el modelo presentado por Medrano [18], resulta forzoso que la discretización difiera de la utilizada en su Tesis y es necesario utilizar otras técnicas para la resolución del sistema.

2.2.1.1. Discretización temporal

En la física clásica el flujo del tiempo es unidireccional, lo que implica que un estado no puede afectar a estados pasados en el tiempo, pero los pasados sí que pueden afectar al estado futuro. Esto posibilita la aplicación de un esquema de discretización explícito donde, de la misma manera, un estado solamente depende de los anteriores, no de los futuros. Esta discretización permite el cálculo secuencial a lo largo del tiempo, lo que simplifica la implementación, al no ser necesario resolver secuencias implícitas que impliquen la resolución de sistemas acoplados.

Hay que tener en cuenta que tanto esquemas implícitos como explícitos tienen ventajas e inconvenientes. Por un lado, los esquemas explícitos son más sencillos de implementar y su tiempo de ejecución es menor, pero ofrecen soluciones menos precisas, sobre todo al incrementar el tamaño del paso en tiempo, y pueden incluso llegar a ser inestables. Una solución es reducir el tamaño del paso, pero esto hace que su resolución sea más lenta y consuma más memoria. Por otro lado, los métodos implícitos no suelen tener problemas de estabilidad y la solución que ofrecen es más precisa, pero el hecho de que se necesite conocer las derivadas en tiempos futuros implica que es inevitable resolver todos los momentos de tiempo simultáneamente, lo cual, en un problema tan complejo como el de este proyecto, no resulta trivial en absoluto.

Debido a que el flujo se considera estático en este problema, todas las variables que dependen del tiempo pueden ser expresadas o bien algebraicamente o bien como ecuaciones diferenciales ordinarias (EDO) de primer orden, de la forma

$$\frac{\partial \varphi}{\partial t} = f(t, \varphi)\,, \tag{32}$$

donde $\varphi \in \mathbb{R}^n$ es una variable cualquiera y $f \in \mathbb{R}^m$ una función cualquiera, con condición inicial $\varphi(t_0) = \varphi_0$. El eje temporal puede dividirse en N_t intervalos de tamaño

$$\Delta t = \frac{T - t_0}{N_t}\,. \tag{33}$$

Los momentos de tiempo en los que se realiza la discretización son

$$t_i = t_0 + i\Delta t, \qquad \forall i \in \{0, \dots, N_t\}\,, \tag{34}$$

y la aproximación de φ en $t = t_i$ se denomina φ_i, es decir, $\varphi(t_i) \approx \varphi_i$.

Para la integración numérica se ha utilizado el método de Euler explícito, que es un método de primer orden que consiste en aproximar la derivada $\partial\varphi/\partial t$ en t_i utilizando la expansión de Taylor de primer orden en torno al momento τ:

$$\varphi = \varphi(\tau) + \frac{\partial}{\partial t}\varphi(\tau)(t - \tau) + \mathcal{O}((t - \tau)^2)\,, \tag{35}$$

que particularizado para el paso de tiempo i, e ignorando el término de segundo orden por tomar valores cercanos al cual se ha centrado la expansión, es

$$\varphi_{i+1} = \varphi_i + \Delta t \frac{\partial \varphi_i}{\partial t}\,. \tag{36}$$

Sustituyendo la expresión de la derivada, que se da en la EDO, se obtiene la siguiente expresión que permite calcular el valor de la variable en un paso y en función del valor en el paso anterior:

$$\varphi_{i+1} = \varphi_i + \Delta t\, f(t_i, \varphi_i)\,. \tag{37}$$

Este es el método de Euler explícito.

2.2.1.2. Discretización espacial

Debido a que tras la simplificación del modelo de transporte solamente el módulo de dinámica de fluidos resulta ser una ecuación diferencial parcial (EDP), solamente este precisa de una discretización espacial para su resolución. El resto de los módulos pueden ser resueltos directamente sobre las coordenadas de una partícula en un cierto instante de tiempo dado. A diferencia de la discretización temporal, un esquema explícito no tiene sentido ya que aparecen en el modelo tanto gradientes como laplacianos, cuya aproximación fuerza a considerar vecinos en ambas direcciones de cada eje, no solo en una de ellas.

Existe además una característica fundamental que hace que la resolución de las ecuaciones de Navier-Stokes sea particularmente complicada. Se trata de un sistema de cuatro ecuaciones (la ecuación de continuidad y una para cada eje espacial de la ecuación del momento) con cuatro incógnitas (la presión y tres ejes espaciales para el flujo), que en principio hacen que el sistema sea determinado, pero la presión no aparece en la ecuación de continuidad que actúa más como una restricción, de modo que la resolución de la ecuación del momento no implica que se mantenga un flujo con divergencia nula. Por esta razón la estabilidad del sistema tiende a ser difícil de obtener con métodos sencillos, y han de aplicarse métodos implícitos en tiempo o utilizar técnicas de predicción-corrección como el método de la proyección [20] [21] [22] [23].

En primer lugar, se ha intentado la implementación del método de las diferencias finitas (Finite Difference Method, FDM) con un esquema de discretización temporal explícito y utilizando la proyección de Chorin-Temam [24] [25], ya que esta es la manera más sencilla de resolver las ecuaciones de Navier-Stokes numéricamente. Sin embargo, ante la inestabilidad de la solución y el hecho de que este método se limita a la resolución de las ecuaciones en mallas estructuradas, finalmente se ha optado por utilizar el software comercial COMSOL Multiphysics [13] para la resolución de la dinámica de fluidos, que hace uso de una técnica más compleja pero más robusta y que además puede ser aplicada a mallas no estructuradas: el método de elementos finitos (Finite Elements Method, FEM).

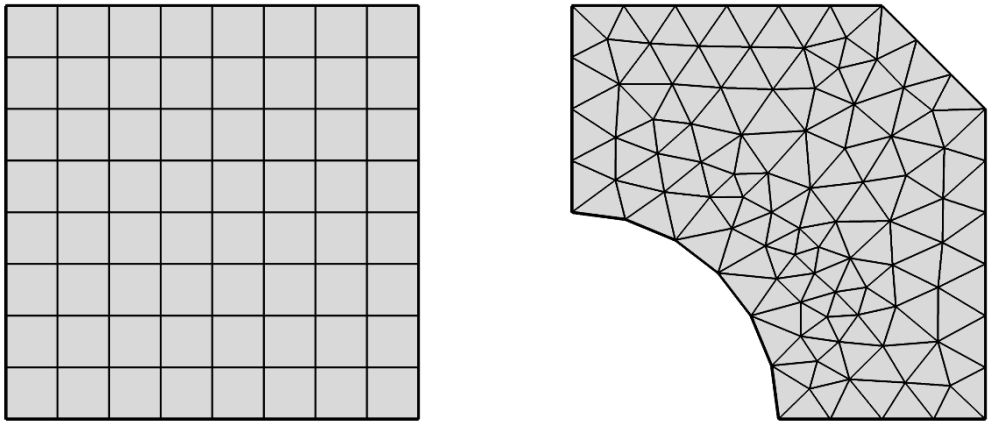
El método de diferencias finitas utiliza mallas estructuradas como la mostrada en la Figura 10(a), generalmente formadas por rectángulos o cuadrados en 2D, o por prismas rectangulares o cubos en 3D, y a pesar de que no es necesario, el dominio sobre el que se suele aplicar es rectangular o con forma de prisma rectangular también. En ese caso la discretización espacial resulta muy evidente, ya que es análoga a la expuesta para la discretización temporal pero extendida a dimensiones más altas. Tomando como dominio un prisma rectangular situado en el intervalo $[x_0, X]$ en el eje x, $[y_0, Y]$ en el eje y, y $[z_0, Z]$ en el eje z, este se puede dividir en $N_x \times N_y \times N_z$ elementos de tal manera que el elemento i, j, k sea representado por un nodo situado en

$$(x, y, z) = (x_0 + i\Delta x, \quad y_0 + j\Delta y, \quad z_0 + k\Delta z), \qquad \forall i \in \{0, \dots, N_x\}, j \in \{0, \dots, N_y\}, k \in \{0, \dots, N_z\}, \tag{38}$$

donde

$$\Delta x = \frac{X - x_0}{N_x}, \qquad \Delta y = \frac{Y - y_0}{N_y}, \qquad \Delta z = \frac{Z - z_0}{N_z}, \tag{39}$$

son los pasos en cada una de los ejes espaciales.



(a) Malla estructurada ortogonal. (b) Malla no estructurada triangular.

Figura 10: Tipos de malla.

A diferencia de FDM, en FEM no resulta necesario que los vecinos se hallen en posiciones ortogonales al nodo, sino que la solución se calcula en cada nodo con funciones de decrecimiento definidas a trozos hacia cada celda contigua, independientemente de su localización. Por esta razón no es necesario tener un vecino en cada eje ni que el número de estos sea determinado, es decir, la malla puede ser no estructurada, como la que se muestra en la Figura 10(b). Para la generación de la malla existen muchas posibilidades, pero la más común es utilizar triángulos en 2D y tetraedros en 3D, que fija el número de vecinos en 3 y 4 respectivamente, uno por cara. El cálculo de la malla puede realizarse también a través de diversos métodos, como puede ser la triangulación de Delaunay [26] o algoritmos de avance de frentes, pero lo más normal es que este paso se realice mediante bibliotecas o en funciones proporcionadas por el software que se esté utilizando para resolver el modelo, como es el caso de COMSOL o la PDE Toolbox de MATLAB [15]. En las dos secciones siguientes presentaremos la discretización del modelo utilizando los dos métodos de discretización espacial que acabamos de discutir.

2.2.2. Método de las diferencias finitas

El método de las diferencias finitas (FDM) [27] [28] [29] [30] combina la discretización temporal propuesta con una discretización espacial en mallas estructuradas. Es el método numérico más sencillo ya que puede plantearse a partir de la formulación diferencial del problema, en lugar de tener que transformarlo a una formulación variacional, como es el caso de FEM. Se basa en aproximar las derivadas parciales de manera que puedan ser sustituidas en la ecuación diferencial y los valores de las variables puedan ser resueltos algebraicamente en cada nodo.

Partiendo de nuevo de la expansión de Taylor centrada en $t = \tau$ y reordenando (35) obtenemos una expresión para las derivadas temporales:

$$\frac{\partial}{\partial t}\varphi(\tau) = \frac{\varphi - \varphi(\tau)}{(t-\tau)} - \mathcal{O}((t-\tau)^2)\,. \tag{40}$$

Es posible particularizar para la discretización temporal explícita utilizada como se ha hecho antes e ignorar el término de segundo orden,

$$\frac{\partial \varphi_i}{\partial t} = \frac{\varphi_{i+1} - \varphi_i}{\Delta t}\,. \tag{41}$$

Esta es una aproximación de las derivadas que guarda un paralelismo con la definición de derivada, donde el límite ha sido ignorado y en su lugar se toman valores de Δt cercanos a cero:

$$\frac{\partial \varphi}{\partial t} = \lim_{\Delta t \to 0} \frac{\varphi(\tau + \Delta t) - \varphi(\tau)}{\Delta t}. \tag{42}$$

Lo mismo puede ser aplicado a las espaciales, de modo que para el eje espacial $\xi \in \{x, y, z\}$ puede escribirse

$$\frac{\partial \varphi_i}{\partial \xi} = \frac{\varphi_{i+1} - \varphi_i}{\Delta \xi}, \tag{43}$$

donde ahora el índice i denota el paso en espacio en la dirección ξ en lugar de tiempo, y $\Delta\xi$ es el tamaño del paso. La ecuación (43) muestra el esquema de diferencias finitas progresiva, ya que φ_{i+1} se encuentra más adelantado que φ_i, y esto es lo que necesitamos en la discretización temporal. Sin embargo, también es posible tomar las diferencias finitas en la dirección contraria, obteniendo las diferencias finitas regresivas,

$$\frac{\partial \varphi_i}{\partial \xi} = \frac{\varphi_i - \varphi_{i-1}}{\Delta \xi}, \tag{44}$$

o en ambas direcciones simultáneamente, obteniendo las diferencias finitas centradas,

$$\frac{\partial \varphi_i}{\partial \xi} = \frac{\varphi_{i+1} - \varphi_{i-1}}{2\Delta \xi}. \tag{45}$$

Esta última también puede entenderse como la media de las dos anteriores. Las dos primeras aproximaciones son de orden uno, mientras que el esquema centrado es de orden dos. Es por esto que se utiliza aquí para la discretización espacial. Además, se puede aplicar esto de manera recursiva para obtener aproximaciones de las segundas derivadas, que serán necesarias para discretizar el laplaciano:

$$\frac{\partial^2 \varphi_i}{\partial \xi^2} = \frac{\varphi_{i+1} - 2\varphi_i + \varphi_{i-1}}{\Delta \xi^2}. \tag{46}$$

En lo sucesivo se utilizará la notación $\varphi_\xi^{i,j,k,n}$ donde los superíndices denotan el paso en los ejes x, y, z y t respectivamente y el subíndice $\xi \in \{x, y, z\}$ denota el componente a considerar en variables vectoriales, y la plantilla de discretización es la presentada en la Figura 11.

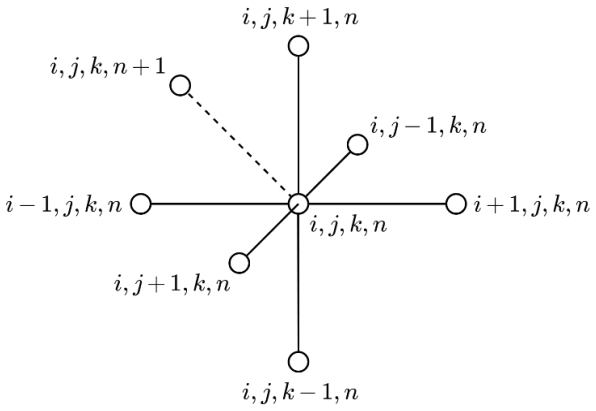


Figura 11: Plantilla de discretización.

Tomando la ecuación del momento (16) podemos escribir cada uno de sus componentes de manera explícita:

$$\frac{\partial u_x}{\partial t} + u_x \frac{\partial u_x}{\partial x} + u_y \frac{\partial u_x}{\partial y} + u_z \frac{\partial u_x}{\partial z} = -\frac{1}{\rho_f}\frac{\partial p}{\partial x} + F_x + \nu\left(\frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_x}{\partial y^2} + \frac{\partial^2 u_x}{\partial z^2}\right), \quad (47)$$

$$\frac{\partial u_y}{\partial t} + u_x \frac{\partial u_y}{\partial x} + u_y \frac{\partial u_y}{\partial y} + u_z \frac{\partial u_y}{\partial z} = -\frac{1}{\rho_f}\frac{\partial p}{\partial y} + F_y + \nu\left(\frac{\partial^2 u_y}{\partial x^2} + \frac{\partial^2 u_y}{\partial y^2} + \frac{\partial^2 u_y}{\partial z^2}\right), \quad (48)$$

$$\frac{\partial u_z}{\partial t} + u_x \frac{\partial u_z}{\partial x} + u_y \frac{\partial u_z}{\partial y} + u_z \frac{\partial u_z}{\partial z} = -\frac{1}{\rho_f}\frac{\partial p}{\partial z} + F_z + \nu\left(\frac{\partial^2 u_z}{\partial x^2} + \frac{\partial^2 u_z}{\partial y^2} + \frac{\partial^2 u_z}{\partial z^2}\right). \quad (49)$$

Aplicando las diferencias finitas (41), (45) y (46), es posible aproximar (47) como

$$\begin{aligned}
&\frac{u_x^{i,j,k,n+1} - u_x^{i,j,k,n}}{\Delta t} + u_x^{i,j,k,n}\frac{u_x^{i+1,j,k,n} - u_x^{i-1,j,k,n}}{2\Delta x} + u_y^{i,j,k,n}\frac{u_x^{i,j+1,k,n} - u_x^{i,j-1,k,n}}{2\Delta y} \\
&+u_z^{i,j,k,n}\frac{u_x^{i,j,k+1,n} - u_x^{i,j,k-1,n}}{2\Delta z} = -\frac{1}{\rho_f}\frac{p^{i+1,j,k,n} - p^{i-1,j,k,n}}{2\Delta x} + F_x \\
&+\nu\left(\frac{u_x^{i+1,j,k,n} - 2u_x^{i,j,k,n} + u_x^{i-1,j,k,n}}{\Delta x^2} + \frac{u_x^{i,j+1,k,n} - 2u_x^{i,j,k,n} + u_x^{i,j-1,k,n}}{\Delta y^2}\right. \\
&\left.+\frac{u_x^{i,j,k+1,n} - 2u_x^{i,j,k,n} + u_x^{i,j,k-1,n}}{\Delta z^2}\right).
\end{aligned} \quad (50)$$

Dado que solamente aparece una derivada temporal puede despejarse $u_x^{i,j,k,n+1}$, de manera que pueda ser resuelto mediante un método explícito de avance temporal a partir de ciertas condiciones iniciales,

$$\begin{aligned}
u_x^{i,j,k,n+1} = u_x^{i,j,k,n} + \Delta t\Bigg[&-u_x^{i,j,k,n}\frac{u_x^{i+1,j,k,n} - u_x^{i-1,j,k,n}}{2\Delta x} \\
&-u_y^{i,j,k,n}\frac{u_x^{i,j+1,k,n} - u_x^{i,j-1,k,n}}{2\Delta y} - u_z^{i,j,k,n}\frac{u_x^{i,j,k+1,n} - u_x^{i,j,k-1,n}}{2\Delta z} \\
&-\frac{1}{\rho_f}\frac{p^{i+1,j,k,n} - p^{i-1,j,k,n}}{2\Delta x} + F_x + \nu\left(\frac{u_x^{i+1,j,k,n} - 2u_x^{i,j,k,n} + u_x^{i-1,j,k,n}}{\Delta x^2}\right. \\
&\left.+\frac{u_x^{i,j+1,k,n} - 2u_x^{i,j,k,n} + u_x^{i,j-1,k,n}}{\Delta y^2} + \frac{u_x^{i,j,k+1,n} - 2u_x^{i,j,k,n} + u_x^{i,j,k-1,n}}{\Delta z^2}\right)\Bigg].
\end{aligned} \quad (51)$$

La misma idea puede aplicarse a (48) y (49) para obtener expresiones en el resto de los ejes. Sin embargo, esta ecuación no cumple necesariamente la condición de continuidad (17) (es decir, $\nabla \cdot \boldsymbol{u} \neq 0$), de modo que es preciso proyectar el campo de velocidades a un campo con divergencia nula de acuerdo al método de la proyección de Chorin-Temam [24] [25].

El método de la proyección de Chorin-Temam propone el cálculo de un campo de velocidades intermedio, $\boldsymbol{u}^*$, que se calcula ignorando el gradiente de la presión en la formulación anterior y, a continuación, utilizar este para obtener el campo de velocidades final en el siguiente paso de tiempo. Es decir, lo primero que ha de hacerse es resolver explícitamente

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} + (\boldsymbol{u}^n \cdot \nabla)\boldsymbol{u}^n = \boldsymbol{F} + \nu\nabla^2\boldsymbol{u}^n \,, \tag{52}$$

utilizando la misma discretización espacial que en (51), que reordenando resulta

$$\boldsymbol{u}^* = \boldsymbol{u}^n + \Delta t[-(\boldsymbol{u}^n \cdot \nabla)\boldsymbol{u}^n + \boldsymbol{F} + \nu\nabla^2\boldsymbol{u}^n] \,, \tag{53}$$

o explícitamente

$$\begin{aligned} u_\xi^{i,j,k,*} = u_\xi^{i,j,k,n} + \Delta t \Bigg[& -u_x^{i,j,k,n} \frac{u_\xi^{i+1,j,k,n} - u_\xi^{i-1,j,k,n}}{2\Delta x} \\ & - u_y^{i,j,k,n} \frac{u_\xi^{i,j+1,k,n} - u_\xi^{i,j-1,k,n}}{2\Delta y} - u_z^{i,j,k,n} \frac{u_\xi^{i,j,k+1,n} - u_\xi^{i,j,k-1,n}}{2\Delta z} + F_\xi \\ & + \nu \Bigg(\frac{u_\xi^{i+1,j,k,n} - 2u_\xi^{i,j,k,n} + u_\xi^{i-1,j,k,n}}{\Delta x^2} + \frac{u_\xi^{i,j+1,k,n} - 2u_\xi^{i,j,k,n} + u_\xi^{i,j-1,k,n}}{\Delta y^2} \\ & + \frac{u_\xi^{i,j,k+1,n} - 2u_\xi^{i,j,k,n} + u_\xi^{i,j,k-1,n}}{\Delta z^2} \Bigg) \Bigg] \,, \qquad \xi = \{x, y, z\}. \end{aligned} \tag{54}$$

Dado este campo de velocidades, se ejecuta el siguiente paso, el cálculo de la presión mediante

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^*}{\Delta t} = -\frac{1}{\rho_f}\nabla p^{n+1} \,. \tag{55}$$

Para resolver esta última ecuación puede tomarse la divergencia de ambos lados,

$$\frac{1}{\Delta t}\nabla \cdot \boldsymbol{u}^{n+1} - \frac{1}{\Delta t}\nabla \cdot \boldsymbol{u}^* = -\frac{1}{\rho_f}\nabla \cdot (\nabla p^{n+1}) \,, \tag{56}$$

aplicar la ecuación de continuidad (17), que hace que $\nabla \cdot \boldsymbol{u}^{n+1} = 0$, y el hecho de que $\nabla \cdot (\nabla p^{n+1}) = \nabla^2 p^{n+1}$ para obtener el problema de Poisson para la presión

$$\frac{\Delta t}{\rho_f}\nabla^2 p^{n+1} = \nabla \cdot \boldsymbol{u}^* \,. \tag{57}$$

Para resolver (57) puede utilizarse la misma discretización y, a continuación, resolver el sistema lineal resultante, que puede hacerse multi-diagonal como muestra la estructura de la Figura 12. Por ello no queda más remedio que formular la solución implícitamente como

$$\begin{aligned} \frac{\Delta t}{\rho_f} \Bigg(& \frac{p^{i-1,j,k,n+1} - 2p^{i,j,k,n+1} + p^{i+1,j,k,n+1}}{\Delta x^2} \\ & + \frac{p^{i,j-1,k,n+1} - 2p^{i,j,k,n+1} + p^{i,j+1,k,n+1}}{\Delta y^2} \\ & + \frac{p^{i,j,k-1,n+1} - 2p^{i,j,k,n+1} + p^{i,j,k+1,n+1}}{\Delta z^2} \Bigg) \end{aligned} \tag{58}$$

$$= \frac{u_x^{i+1,j,k,*} - u_x^{i-1,j,k,*}}{2\Delta x} + \frac{u_y^{i,j+1,k,*} - u_y^{i,j-1,k,*}}{2\Delta y} + \frac{u_z^{i,j,k+1,*} - u_z^{i,j,k-1,*}}{2\Delta z}.$$

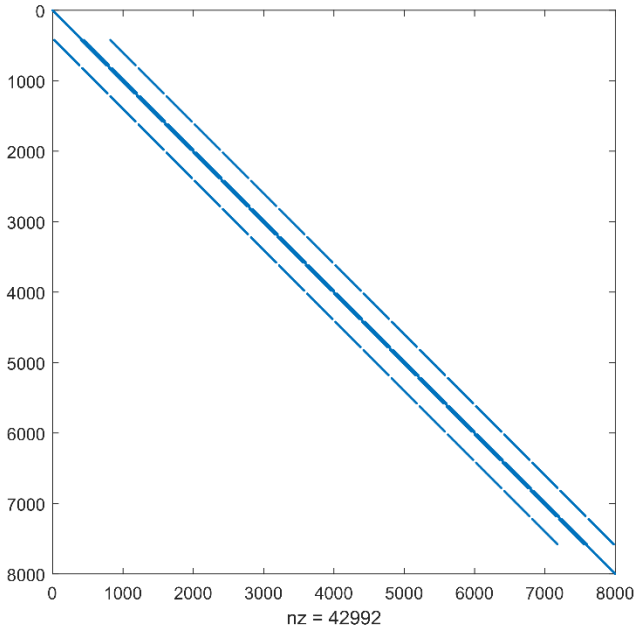


Figura 12: Estructura de la matriz sparse de una malla de $20 \times 20 \times 20$ nodos.

Como el nodo (i, j, k) solamente se relaciona con sus vecinos ortogonales directos, la mayor parte de las entradas de la matriz son ceros, lo cual favorece su resolución gracias a técnicas de matrices huecas (existen tanto técnicas directas [31] como iterativas [32]). Además, es importante notar que la matriz de coeficientes no varía, de modo que, para acelerar la resolución del problema, que ha de realizarse en cada paso de tiempo, es posible realizar una factorización LU [33] al principio del algoritmo y utilizar esta para resolver el sistema con el vector de la mano derecha, b, cambiando en cada paso de tiempo. La factorización LU consiste en descomponer la matriz original A en dos matrices triangulares, una inferior (L) y una superior (U), donde la diagonal de L está compuesta por unos. Entonces, para resolver el sistema lineal

$$A\varphi = \boldsymbol{b} \tag{59}$$

no es necesario invertir A, sino que basta con resolver secuencialmente (que resulta mucho más rápido)

$$L\tilde{\varphi} = \boldsymbol{b} \tag{60}$$

y

$$U\varphi = \tilde{\varphi}\,. \tag{61}$$

En cualquier caso, se obtiene una solución para p^{n+1} que puede ser utilizada, aplicando el principio de Helmholtz, para calcular el campo de velocidades final,

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^{*} - \frac{\Delta t}{\rho_f}\nabla p^{n+1}\,, \tag{62}$$

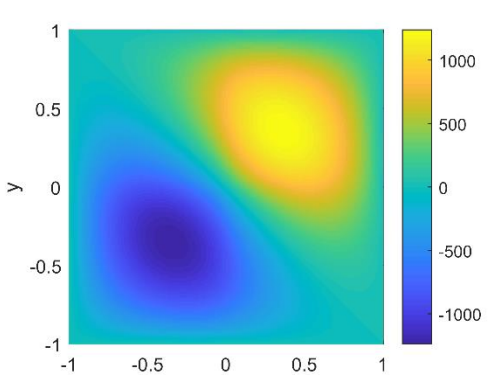
o aplicando la discretización,

$$u_x^{i,j,k,n+1} = u_x^{i,j,k,*} - \frac{\Delta t}{\rho_f}\frac{p^{i+1,j,k,n+1} - p^{i-1,j,k,n+1}}{2\Delta x}, \tag{63}$$

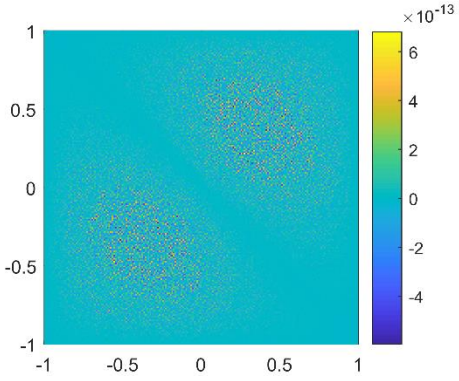
$$u_y^{i,j,k,n+1} = u_y^{i,j,k,*} - \frac{\Delta t}{\rho_f}\frac{p^{i,j+1,k,n+1} - p^{i,j-1,k,n+1}}{2\Delta y}, \tag{64}$$

$$u_z^{i,j,k,n+1} = u_z^{i,j,k,*} - \frac{\Delta t}{\rho_f}\frac{p^{i,j,k+1,n+1} - p^{i,j,k-1,n+1}}{2\Delta z}. \tag{65}$$

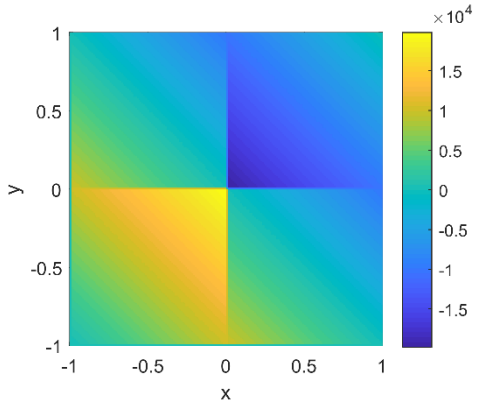
La aplicación del método se realiza en tres pasos. En primer lugar se realiza la aproximación del campo de velocidades, $\boldsymbol{u}^{*}$, mediante la ecuación (54). El segundo paso consiste en utilizar este campo para obtener la presión dada por el sistema lineal definido por (58), que a su vez se resuelve utilizando las ecuaciones (60) y (61). Finalmente se obtiene el campo de velocidades utilizando las ecuaciones explícitas (63), (64) y (65).



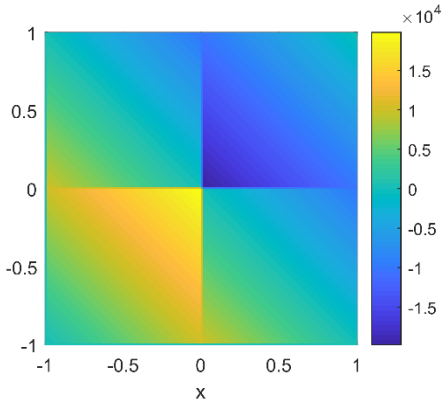
(a) Presión p.



(b) Error relativo.



(c) $\nabla^2 p$ aproximado.



(d) $\nabla^2 p$ exacto.

Figura 13: Ejemplo de solución del problema de Poisson en 2D.

Es posible comprobar el correcto funcionamiento del segundo paso, la resolución del problema de Poisson, independientemente de la solución del flujo, como se ha hecho en la Figura 13. Los ejes horizontales y verticales se corresponden con los ejes espaciales y el eje de color muestra el campo especificado en cada caso. Arriba a la izquierda se muestra la presión, solución del problema (bidimensional en este ejemplo). En la parte de arriba a la derecha se muestra el error relativo cometido entre la solución exacta, que en este test unitario es conocida, y la obtenida mediante los métodos numéricos. El error es tan solo del orden de 10^{-13}, lo cual muestra que se trata de una excelente aproximación. En la parte de abajo se compara el laplaciano de la solución obtenida, a la izquierda, con el laplaciano exacto, a la derecha (ver ecuación (57) para comprobar su significancia).

Este método ha sido implementado, pero no se ha logrado alcanzar la convergencia. Debido a sus grandes limitaciones, en lugar de condicionarlo para lograr obtener una solución válida, se ha optado por cambiar de método y utilizar COMSOL para obtener la solución del flujo, que hace uso del método de los elementos finitos.

2.2.3. Método de los elementos finitos

El método de los elementos finitos (FEM) [34] [35] [36] es un método numérico mucho más robusto que FDM o incluso que el método de volúmenes finitos (FVM). Permite la utilización de mallas no estructuradas y geometrías complejas en problemas no solo de dinámicas de fluido computacional (Computational Fluid Dynamics, CFD), sino también en otro tipo de problemas físicos, razón por la cual es el implementado en multitud de softwares comerciales multi-física. En este proyecto, su implementación no ha sido necesaria ya que se ha optado por utilizar el software comercial COMSOL, pero para facilitar su desarrollo futuro en fases posteriores del proyecto, a continuación, se describe brevemente el método aplicado a las ecuaciones de Navier-Stokes.

En primer lugar, es necesario transformar la formulación diferencial a la correspondiente formulación variacional, sobre la cual se aplican los esquemas de discretización comentados con anterioridad y se representan las variables independientes como combinaciones lineales de funciones definidas a trozos que forman una base. Entonces, el sistema lineal resultante puede ser resuelto para obtener los coeficientes de las funciones base y así aproximar la variable de interés.

Para transformar las ecuaciones de Navier-Stokes diferenciales (16) y (17) a forma variacional es necesario multiplicar cada una de las ecuaciones por funciones test, $w \in \left(H^1(\Omega)\right)^3$ y $m \in H^1(\Omega)$ respectivamente, donde $H^1(\Omega)$ representa el correspondiente espacio de Sóbolev aplicado al dominio espacial tridimensional Ω. Tras ello, es necesario integrar cada ecuación en el dominio, con lo cual se obtiene

$$\int_\Omega w\frac{\partial \boldsymbol{u}}{\partial t}\,d\Omega + \int_\Omega w(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\,d\Omega = -\int_\Omega \frac{1}{\rho_f} w\nabla p\,d\Omega + \int_\Omega w\boldsymbol{F}\,d\Omega + \int_\Omega \nu w\nabla^2\boldsymbol{u}\,d\Omega\,, \tag{66}$$

$$\int_\Omega m\nabla\cdot\boldsymbol{u}\,d\Omega = 0\,. \tag{67}$$

Aplicando la primera identidad de Green al último término de la parte derecha de (66), se obtiene

$$\begin{aligned}\int_\Omega w\frac{\partial \boldsymbol{u}}{\partial t}\,d\Omega + \int_\Omega w(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\,d\Omega &= -\int_\Omega \frac{1}{\rho_f} w\nabla p\,d\Omega + \int_\Omega w\boldsymbol{F}\,d\Omega \\ &+ \int_\Gamma \nu w(\nabla\boldsymbol{u}\cdot\boldsymbol{n})\,d\Gamma - \int_\Omega \nu(\nabla w\cdot\nabla\boldsymbol{u})\,d\Omega\,,\end{aligned} \tag{68}$$

donde $\Gamma \in \{\Gamma_{\text{top}} \cup \Gamma_{\text{bed}} \cup \Gamma_{\text{in}} \cup \Gamma_{\text{out}}\}$ representa el contorno completo del dominio. La integral en el contorno puede particularizarse para cada una de las sub-fronteras

$$\begin{aligned}\int_\Omega w\frac{\partial \boldsymbol{u}}{\partial t}\,d\Omega + \int_\Omega w(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\,d\Omega = -\int_\Omega \frac{1}{\rho_f} w\nabla p\,d\Omega + \int_\Omega w\boldsymbol{F}\,d\Omega \\ + \int_{\Gamma_{\text{top}}} \nu w(\nabla\boldsymbol{u}\cdot\boldsymbol{n})\,d\Gamma_{\text{top}} + \int_{\Gamma_{\text{bed}}} \nu w(\nabla\boldsymbol{u}\cdot\boldsymbol{n})\,d\Gamma_{\text{bed}} + \int_{\Gamma_{\text{in}}} \nu w(\nabla\boldsymbol{u}\cdot\boldsymbol{n})\,d\Gamma_{\text{in}} \\ + \int_{\Gamma_{\text{out}}} \nu w(\nabla\boldsymbol{u}\cdot\boldsymbol{n})\,d\Gamma_{\text{out}} - \int_\Omega \nu(\nabla w\cdot\nabla\boldsymbol{u})\,d\Omega\,,\end{aligned} \tag{69}$$

y sobre estas aplicar las condiciones de contorno pertinentes. Con esto, obtenemos el siguiente problema variacional: *Encontrar* $\boldsymbol{u} \in \left(H^1(\Omega)\right)^3$ *y* $p \in H^1(\Omega)$ *tal que para todo* $w \in \left(H^1(\Omega)\right)^3$ *y* $m \in H^1(\Omega)$ *se cumplan (69) y (67).*

Entonces se aplica la discretización de modo que las funciones continuas $\boldsymbol{u}, p, w$ y m se aproximan mediante las funciones a trozos $\boldsymbol{u}^h, p^h, w^h$ y m^h respectivamente. El espacio de Sóbolev discretizado se denomina $H_h^1(\Omega)$. Estas funciones se definen mediante la suma de una serie de funciones base $\{\psi_1, \dots, \psi_{N_e}\}$ (también definidas a trozos, centradas en los nodos de la malla y cuyo valor solamente difiere de cero en los entornos del nodo correspondiente), y una serie de coeficientes para los cuales se resuelve el sistema lineal, $\{\varphi_1^h, \dots, \varphi_{N_e}^h\}$, donde N_e es el número de elementos (o celdas) de la malla. La combinación de las bases se expresa en forma del sumatorio

$$\varphi^h(\boldsymbol{x}) = \sum_{i=1}^{N_e} \varphi_i^h \psi_i(\boldsymbol{x})\,, \tag{70}$$

donde $\varphi \in \{\boldsymbol{u}, p, w, m\}$. Aplicando estas funciones discretizadas al problema variacional (con las condiciones de contorno pertinentes aplicadas) se obtiene el problema de elementos finitos. La elección de estas funciones base puede incrementar o reducir la precisión de la solución. Lo común es seleccionar funciones polinómicas a trozos y, cuanto mayor sea el orden, mayor es el tamaño del elemento que se puede elegir para obtener un nivel de error determinado, pero más difícil de computar resulta.

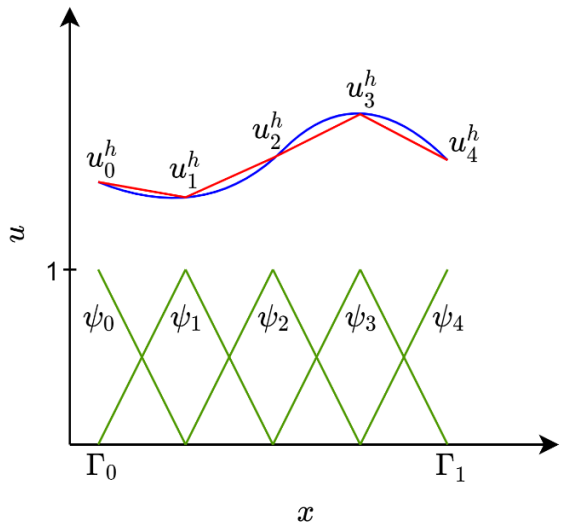


Figura 14: Ejemplo de FEM unidimensional.

En la Figura 14 se muestra un ejemplo unidimensional de la aproximación proporcionada por FEM. La línea azul representa la solución exacta, u; la línea roja la aproximación proporcionada por u^h; y las líneas verdes representan cada una de las funciones base, que en este caso son lineares. En las fronteras Γ_0 y Γ_1 se muestra que, mediante la aplicación de funciones base ligeramente diferentes (con una sola vertiente), puede conseguirse que el campo de la variable a resolver sea no nulo en las fronteras.

3. Resolución de la Simulación de los Modelos

Una vez se ha descrito el modelo físico, el siguiente paso es resolverlo y obtener las soluciones que se buscan. Al hacer uso de métodos numéricos para su resolución, es necesario antes de obtener dichas soluciones implementar el modelo, ya sea directamente en código o haciendo uso de software específico. A continuación, se presentan la implementación final utilizada en este proyecto y los resultados obtenidos.

3.1. Implementación de la simulación

La obtención de una solución del problema de transporte de partículas propuesto en el Capítulo 7 se realiza en dos partes. Primero es necesario resolver el modelo de dinámica de fluidos para obtener el flujo estacionario en COMSOL [13], en particular el campo de velocidades en el marco Euleriano. A continuación, se ha de ejecutar un código escrito en MATLAB [14] para importar el flujo y resolver el resto de componentes del modelo.

COMSOL ofrece un módulo de transporte de partículas. Sin embargo, este está diseñado para operar con partículas de densidad constante, de modo que resulta imposible introducir en el modelo la dinámica de la densidad celular, influenciada por la radiación solar. Por esta razón se opta por cambiar de herramienta a MATLAB para la segunda parte.

3.1.1. Simulación de la dinámica de fluidos

Inicialmente se quería implementar la resolución del flujo en MATLAB también mediante la implementación de algún método numérico como FDM, FEM o FVM. Sin embargo, la biblioteca de resolución de ecuaciones diferenciales parciales (PDE Toolbox) de MATLAB [15] no tiene soporte para las ecuaciones de Navier-Stokes, y la implementación del método numérico desde cero resulta demasiado complicada. Además, los métodos más sencillos de implementar como FDM tienen limitaciones importantes en cuanto a su estabilidad y a su limitado soporte de mallas complejas. Por esa razón se ha optado por utilizar un paquete de software externo para simular la dinámica del fluido y, posteriormente, importar la solución a MATLAB para poder simular el resto de los componentes del modelo.

Existen numerosos programas capaces de resolver la dinámica de fluidos en mallas no estructuradas, algunos de ellos de código abierto, como puede ser OpenFOAM [37] o Gerris [38]; y otros comerciales, como ANSYS Fluent [39], Autodesk CFD [40] o el elegido en este proyecto, COMSOL Multiphysics [13]. En particular, se ha seleccionado este basado en la experiencia previa del autor de este Trabajo y no por ninguna característica técnica en especial. También pueden encontrarse implementaciones de terceros que resuelvan el problema del flujo en MATLAB, como las presentadas en [41] o [42], aunque suelen presentar limitaciones por estar diseñadas para problemas específicos.

A continuación, se describe el proceso de configuración del modelo en COMSOL Multiphysics para que pueda servir de referencia, ya que algunas técnicas pueden ayudar a reducir el tiempo de computación y aumentar la precisión de la solución. En concreto, se explica cómo definir la geometría del modelo, cómo configurar la simulación y el mallado, y cómo optimizar el proceso de simulación.

3.1.1.1. Geometría

El primer paso que hay que realizar es determinar la geometría del modelo a resolver. Al tener una geometría irregular, cualquier malla cerrada es aceptable, pero la idea es utilizar un modelo del lago o embalse de interés. Dado que este proyecto no se trata de un estudio de un lago

particular, un lago ficticio es modelado en una herramienta de diseño asistido por ordenador (Computer Assisted Design, CAD), en particular en el programa de código abierto Blender [12]. Si se tratara de una masa de agua real, la geometría podría formarse mediante el escaneo directamente en la localización de interés a través de técnicas de batimetría [43] y un mallado en el post-procesado a través de técnicas de triangulación, que puede incluso computarse en MATLAB mediante la función `boundary`; o bien mediante el modelado en CAD referenciando las curvas de nivel de un mapa. Para garantizar el buen funcionamiento del modelo en etapas posteriores es imperativo que la superficie del agua sea plana y esté situada en $z = 0$, ya que de otro modo la intensidad lumínica, calculada en base a la coordenada z, necesitaría una transformación, que de haber definido correctamente la geometría, resultaría innecesaria. Finalmente, para su utilización en COMSOL es necesario exportar la geometría a un archivo STL que preserve las coordenadas, como se muestra en la Figura 15.

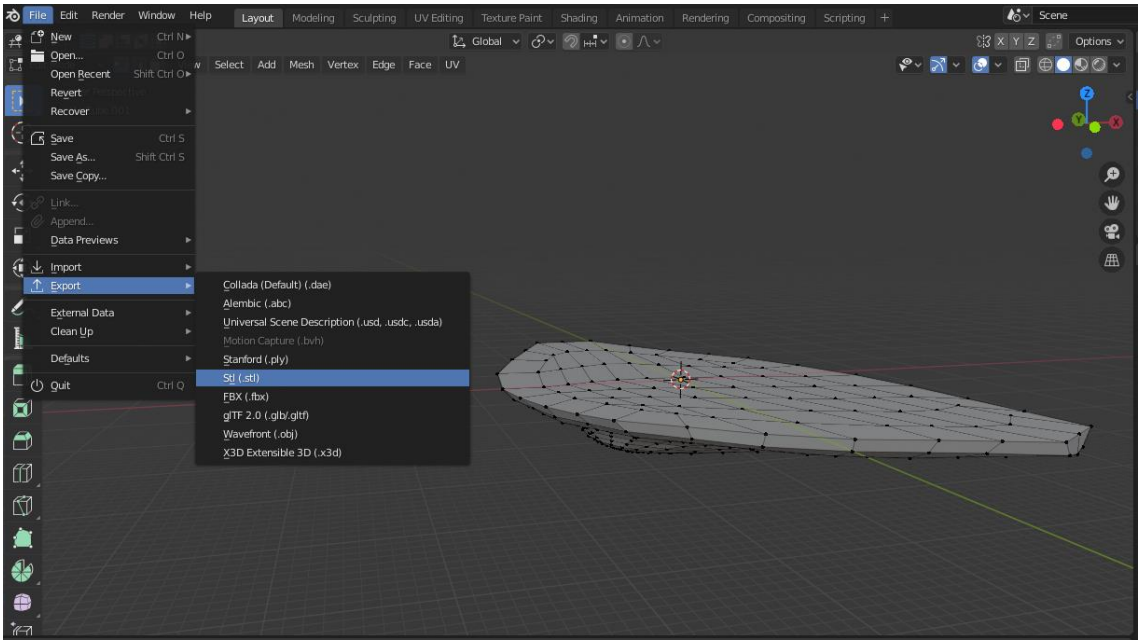
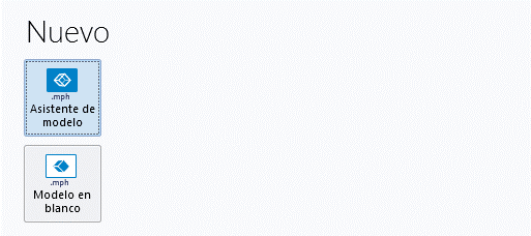


Figura 15: Geometría en Blender.

3.1.1.2. Configuración del modelo

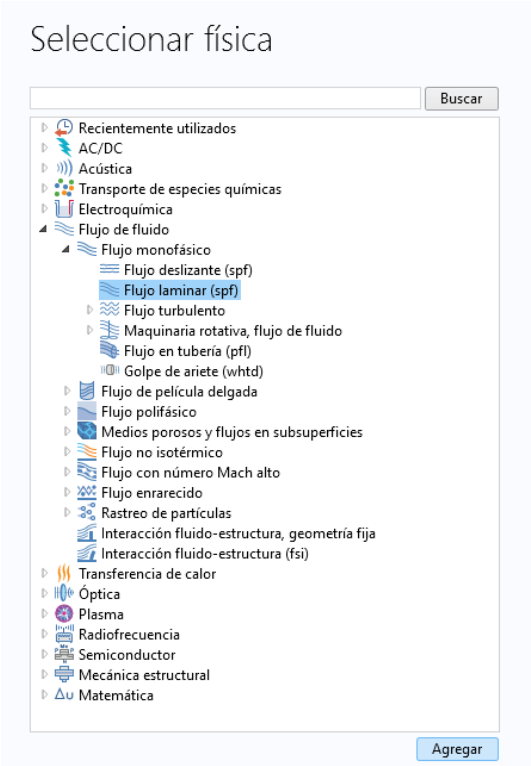
Ya en COMSOL se comienza creando el modelo. Para ello, tal y como se muestra en la Figura 16, se selecciona *Modelo 3D* en el *Asistente de modelo* y como física se elige *Flujo laminar (spf)*. El estudio elegido es *Estacionario*. Tras crear el modelo, lo primero que ha de hacerse es importar la geometría que se ha creado anteriormente mediante el submenú *Importar* que aparece al hacer click derecho en *Geometría*. A continuación, haciendo click derecho en *Materiales*, se añade un material de la biblioteca, agua en este caso, al dominio.

Predeterminadamente aparece una condición de contorno no deslizante en toda la frontera. Esto ha de cambiarse de manera que esté de acuerdo con el modelo a resolver. La *Pared* existente formará la frontera Γ_{bed}, y se añade bajo *Flujo laminar (spf)* una *Pared* adicional, donde se aplica la condición de contorno de *Deslizamiento* a la frontera Γ_{top}. Se añaden también una *Entrada* para la frontera Γ_{in}, donde se especifica una *Velocidad normal de entrada* de 5×10^{-3} m/s, y una *Salida* para Γ_{out} con presión nula y sin reflujo. Obviamente estos parámetros pueden ser modificados para otro tipo de embalse.

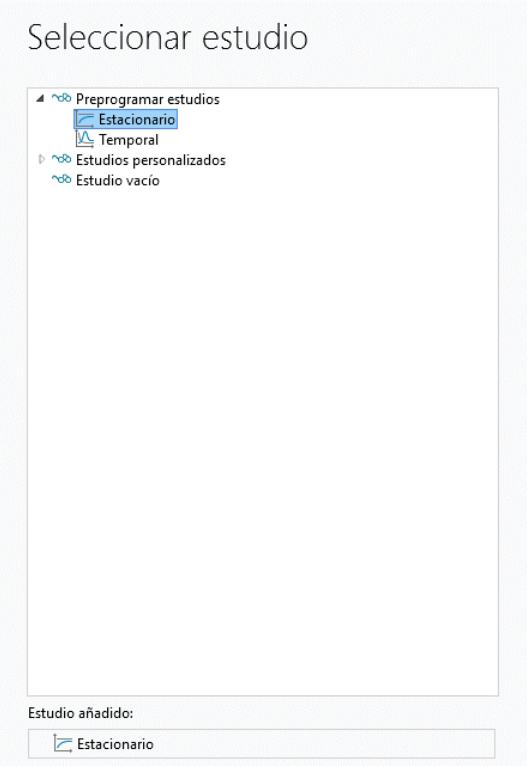


(a) Primer paso: Asistente de modelo.

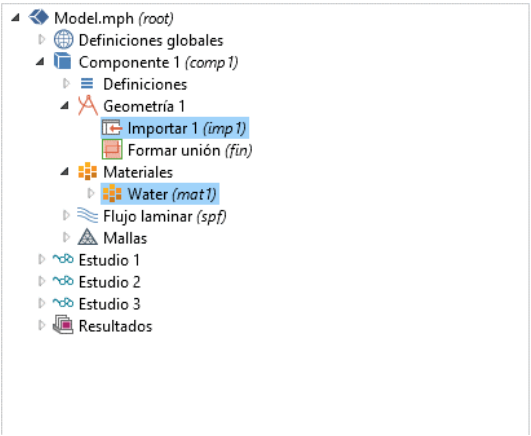
(b) Segundo paso: *Modelo 3D.*



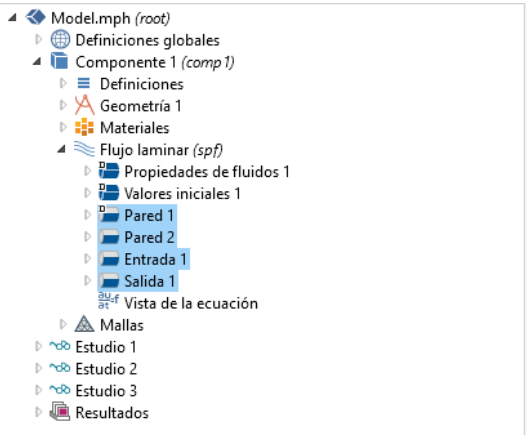
(c) Tercer paso: Flujo laminar (spf).



(d) Cuarto paso: Estudio estacionario.



(e) Quinto paso: Importar geometría y añadir material.

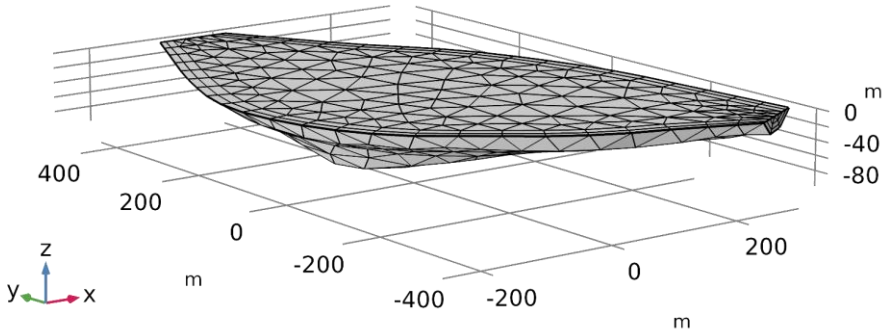


(f) Sexto paso: Añadir condiciones de contorno.

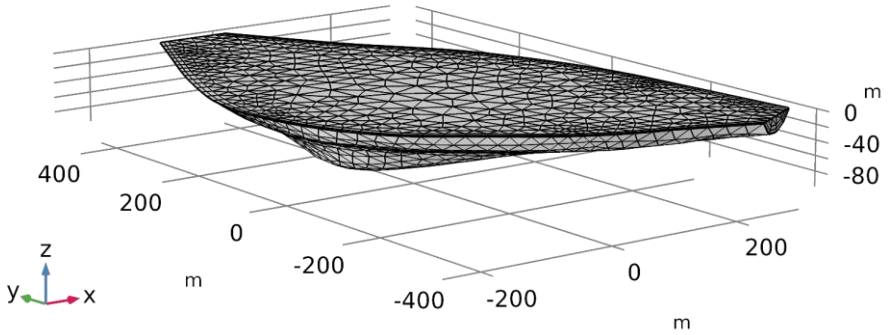
Figura 16: Configuración del modelo.

3.1.1.3. Mallado

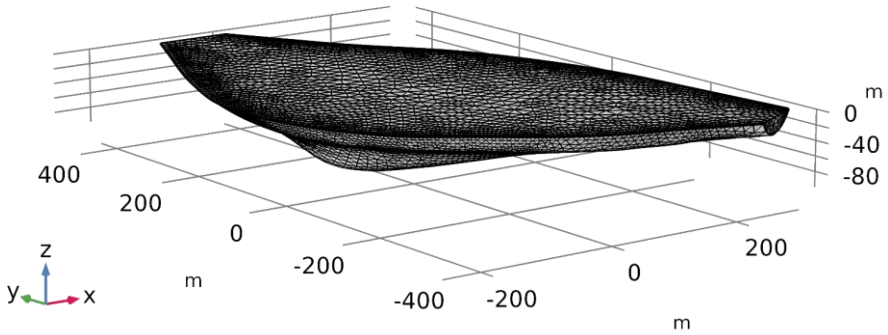
El siguiente paso es el mallado. Este es un paso crítico ya que una malla demasiado fina no permite la convergencia del modelo y consume más memoria, lo cual se propaga a etapas posteriores de la implementación y puede hacer que la posterior ejecución del código del modelo de las partículas desde MATLAB resulte lenta.



(a) Malla 1: Tamaño de elemento *Extremadamente grueso*.



(b) *Malla 2*: Tamaño de elemento *Más grueso.*



(c) *Malla 3*: Tamaño de elemento *Normal.*

Figura 17: Mallas utilizadas.

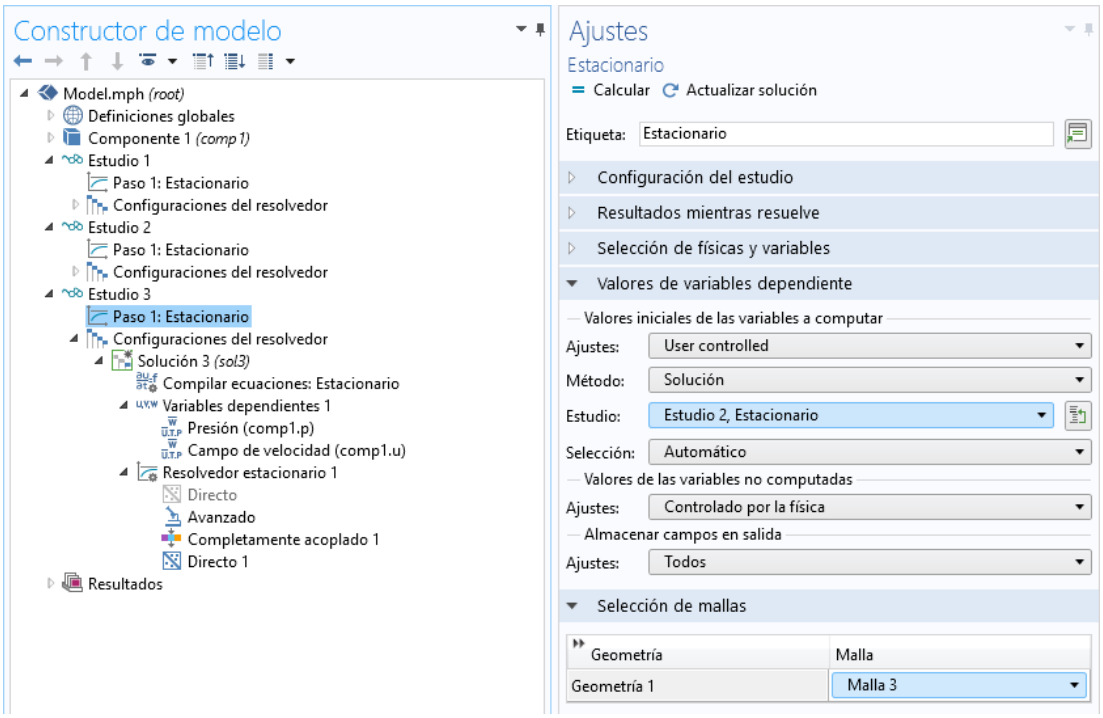
Se selecciona la malla de manera que sea controlada por la física. El objetivo es alcanzar un tamaño de elemento *Normal*, pero tratar de hallar una solución en tal malla a partir de condiciones iniciales nulas puede dar problemas de convergencia, por lo que se halla la solución en mallas progresivamente más refinadas. Para ello se crean las tres mallas que se muestran en la Figura 17:

Malla 1, de tamaño de elemento *Extremadamente grueso*, *Malla 2*, de tamaño de elemento *Más grueso*, y *Malla 3*, de tamaño de elemento *Normal*. Es de notar que la malla de COMSOL (la discretización espacial) no guarda relación alguna con la malla del CAD, excepto que mantiene el mismo volumen y forma en ambos casos.

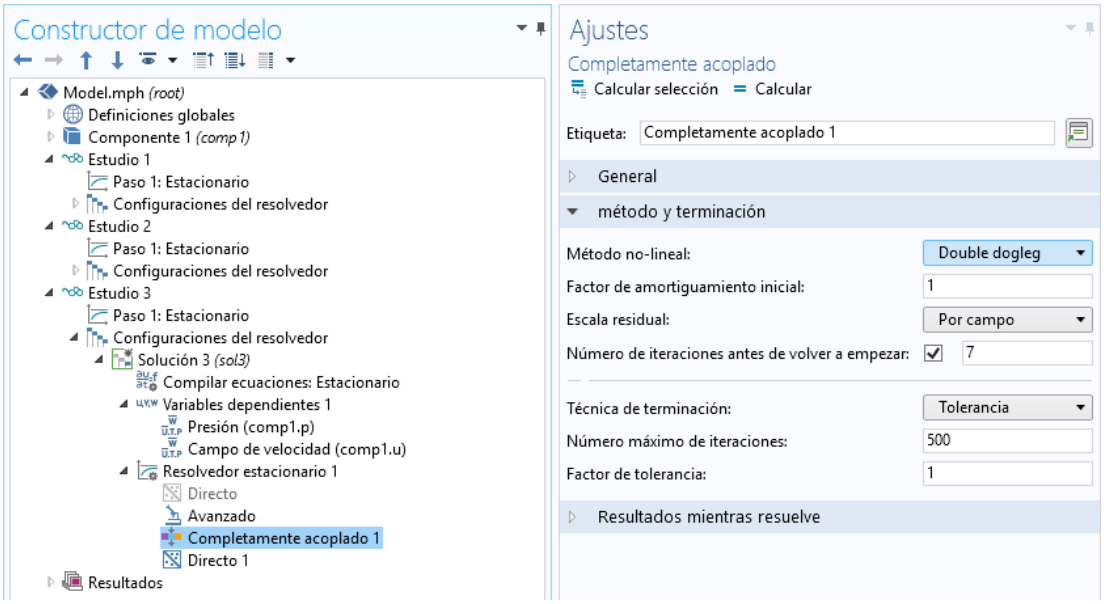
3.1.1.4. Estudios

De la misma manera que se añaden tres mallas se añaden también tres estudios, todos ellos estacionarios. Para mejorar las probabilidades de convergencia ha de configurarse el resolvedor. Se ha comprobado que el que mejores resultados proporciona es el resolvedor *Double dogleg* (en lugar de alguna variante del de Newton) para el *Resolvedor estacionario Completamente acoplado*. Si en alguno de los pasos no se halla la convergencia, es posible aumentar el *Número máximo de iteraciones* para darle más tiempo.

El *Estudio 1* utilizará la *Malla 1* y condiciones iniciales nulas, el *Estudio 2* la *Malla 2* y como condiciones iniciales la solución del *Estudio 1*, y el *Estudio 3* utilizará la *Malla 3* y como condiciones iniciales la solución del *Estudio 2*. Como se muestra en la Figura 18, la selección de la malla se realiza en *Paso 1: Estacionario* bajo *Selección de mallas*, y las condiciones iniciales se seleccionan bajo *Valores de variables dependientes*. Tras resolver los tres estudios en orden es necesario exportar los resultados del *Estudio 3* a un archivo de texto, tal y como aparece en la Figura 19.



(a) Selección de valores iniciales y malla.



(b) Selección de resolvedor no lineal.

Figura 18: Configuración de los estudios.

Para mejorar la convergencia, el tiempo de cálculo y la precisión, podría resultar útil añadir pasos con tamaños de mallas intermedios. Durante el desarrollo del modelo en COMSOL de este proyecto se ha utilizado la configuración de tres mallas que se ha detallado, pero en el cálculo de la solución definitiva, que se ha utilizado posteriormente para realizar la optimización, se han utilizado seis mallas refinando estas hasta un tamaño de elemento *Fino*. Además, se ha seleccionado esta configuración ficticia, pero en la práctica han de seleccionarse condiciones de contorno, propiedades materiales y geometrías apropiadas para cada caso. También es posible seleccionar un estudio temporal o un modelo turbulento, pero esto implicaría cambios en las ecuaciones de transporte, de modo que tales modificaciones han de hacerse con cierto cuidado. Además, es de notar que, como ha quedado reflejado en esta sección, no es necesario introducir las ecuaciones utilizadas en COMSOL, ya que estas se encuentran predeterminadamente implementadas en el módulo utilizado.

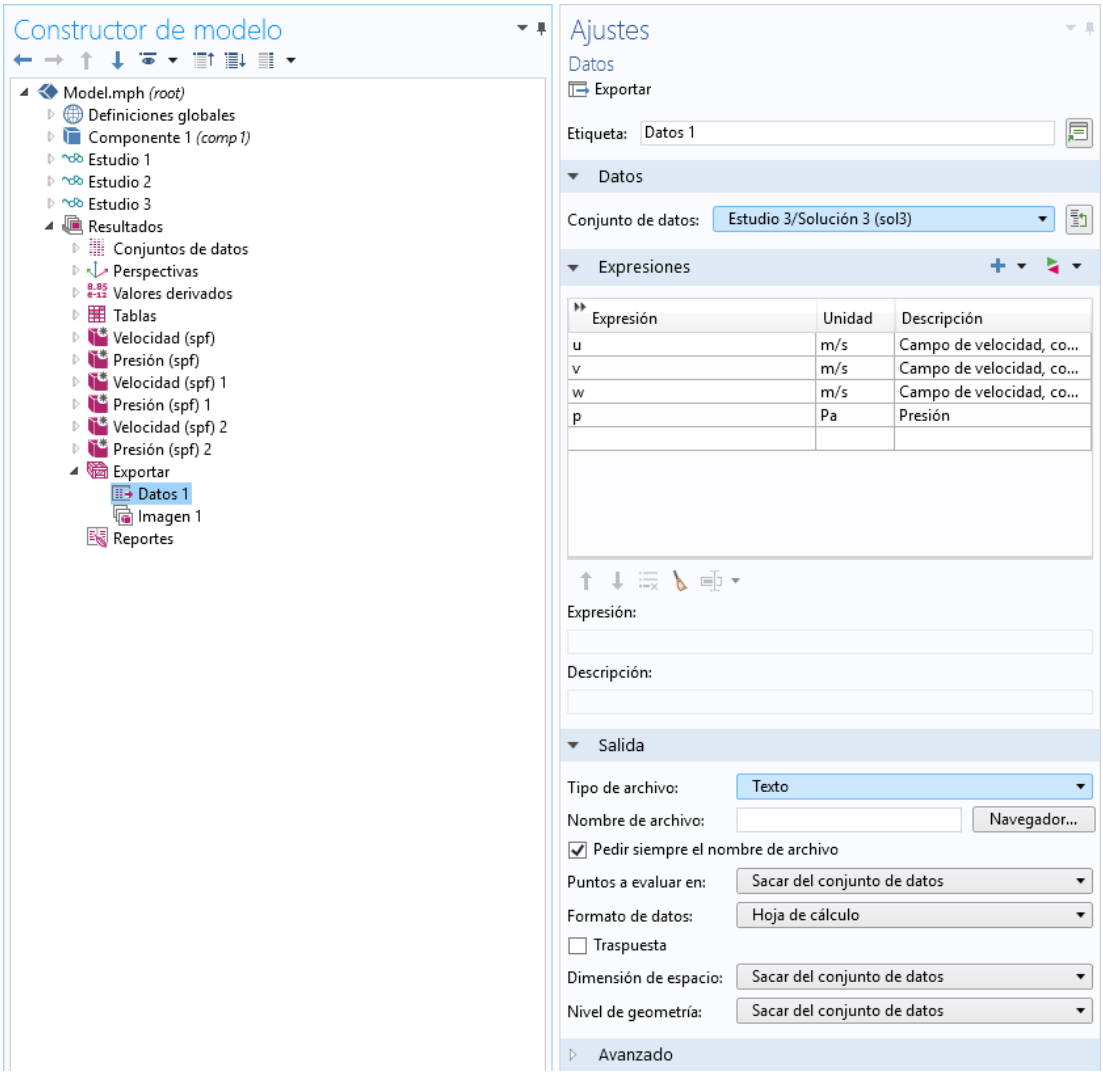


Figura 19: Exportación de resultados.

3.1.2. Simulación del transporte de partículas

A partir de los resultados obtenidos en COMSOL, es posible resolver el resto de ecuaciones del modelo. En lugar de resolver estas ecuaciones en todos los nodos de la malla, al no haber derivadas espaciales presentes, es más económico resolverlas directamente sobre las posiciones de las partículas, incluyendo en cada paso temporal el bucle de acoplamiento completo que se ha mostrado en la Figura 9. El código correspondiente ha sido escrito en MATLAB y se presenta en el Apéndice 1.

3.1.2.1. Inicialización

En primer lugar, se declaran las variables reflejadas en la Tabla 2, que son las utilizadas en la Tesis de Medrano [18]. También se define el vector temporal discretizado en el intervalo $[0, 720]$ h, con un paso de tiempo $\Delta t = 0.1$ h. Se importan los resultados obtenidos en COMSOL mediante el comando `load` y se denomina `N` al número de nodos de la malla. Dado que las ecuaciones se evaluarán en la posición de las partículas, será necesario interpolar las velocidades del fluido en espacios entre los nodos. Para ello, en lugar de interpolar en cada paso de tiempo para cada partícula, es preferible crear una función de interpolación al comienzo del código

mediante el comando `scatteredInterpolant` que pueda ser evaluada directamente sin necesidad de volver a hallar los coeficientes de interpolación cada vez. Una gran ventaja de este método es que la cantidad de nodos de la simulación de fluidos no tiene un efecto importante en el tiempo de computación, ya que una vez creada la función de interpolación, que solo ha de calcularse una vez, estos datos son completamente innecesarios, de modo que pueden incluso ser eliminados para ahorrar memoria.

Tabla 2: Parámetros físicos.

Viscosidad dinámica	ν	3×10^{-6}	$\mathrm{m^2 s^{-1}}$
Densidad del fluido	ρ_f	1000	$\mathrm{kg\ m^{-3}}$
Aceleración de la gravedad	$\boldsymbol{g}$	$-9.81\ \hat{k}$	$\mathrm{m\ s^{-2}}$
Diámetro de las partículas	d_p	8×10^{-4}	m
Factor normativo	β	1.5×10^{-3}	$\mathrm{s^2 m^{-3}}$
Límite de foto-inhibición	I_0	146.43	$\mathrm{W\ m^{-2}}$
Tasa de cambio de densidad celular	γ	-8.3×10^{-3}	$\mathrm{kg\ m^{-3} s^{-1}}$
Decaimiento de densidad celular	α	4.7×10^{-4}	$\mathrm{s^{-1}}$
Densidad celular mínima	C_ρ	1037	$\mathrm{kg\ m^{-3}}$
Contenido celular en la colonia	n_{cell}	10	%
Contenido gaseoso en la colonia	n_{gas}	5	%
Densidad del mucílago	ρ_{muc}	998	$\mathrm{kg\ m^{-3}}$
Irradianza máxima	$I_{\max}$	800	$\mathrm{W\ m^{-2}}$
Coeficiente de atenuación lumínica	K_i	0.8	$\mathrm{m^{-1}}$
Límite de irradianza	I_{lim}	5.75	$\mathrm{W\ m^{-2}}$
Duración de un día	T_{day}	24	h

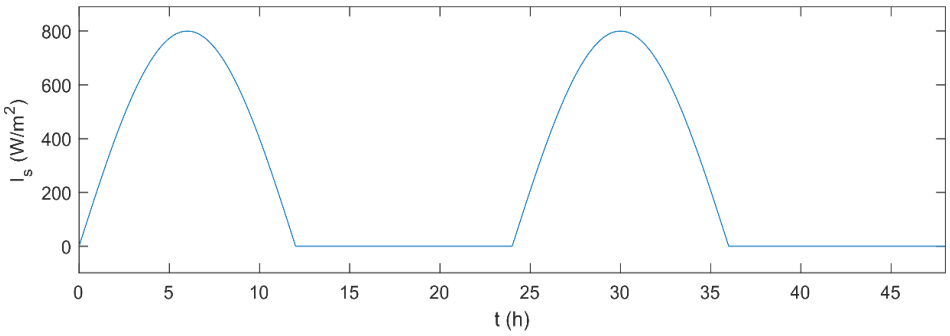


Figura 20: Intensidad lumínica en la superficie.

A continuación se definen las funciones de Heaviside y de intensidad lumínica en la superficie, y las posiciones y densidades iniciales de las partículas, $\boldsymbol{x}^0_{\mathrm{cell}}$ y ρ^0_{cell}, respectivamente. En la práctica sería sencillo utilizar intensidades lumínicas recogidas experimentalmente por sensores, bien en cada paso de tiempo, o bien hallando los valores de intensidad en cada paso temporal por interpolación. Sin embargo, al ser este un caso ficticio, se aproxima la irradianza mediante

$$I_s(t) = \max\left\{0, I_{\max} \sin\left(\frac{2\pi}{T_{\mathrm{day}}} t\right)\right\}, \tag{71}$$

con t en las mismas unidades que T_{day}. La función a lo largo de un periodo de 48 h se muestra en la Figura 20.

A partir de los nodos importados de COMSOL se reconstruyen las fronteras de la malla para poder aplicar las condiciones de contorno para las ecuaciones de las partículas y se muestran las figuras preliminares, en particular la intensidad lumínica (Figura 5) y las fronteras de la malla reconstruida.

3.1.2.2. Bucle de tiempo

El bucle de tiempo consiste principalmente en realizar los cálculos del bucle de acoplamiento en el orden establecido en la Figura 9. Se comienza obteniendo la intensidad lumínica en las posiciones de cada partícula mediante (18). A continuación, se calcula la densidad celular de manera incremental, dando un solo paso de tiempo del método de Euler (37) utilizando las expresiones de (19) y (20). Tras esto, se calcula la densidad de la colonia para cada partícula con (21), y se interpolan las velocidades del flujo en las posiciones de las partículas utilizando la función de interpolación mencionada con anterioridad. Después, se calculan las velocidades de cada partícula con (22) y se integra la posición utilizando de nuevo el método de Euler y la expresión (23). Finalmente, se imponen las condiciones de contorno del transporte, limitando la posición vertical a $z < 0$ y congelando la posición de la partícula si el paso de tiempo la envía fuera del dominio a través del resto de fronteras, lo cual se comprueba gracias a la función `inpolyhedron.m` que puede encontrarse en [16]. Esto representa una iteración temporal, tras la cual, con los nuevos valores de las posiciones y densidades celulares, se realiza la siguiente iteración.

En cada uno de los pasos se guardan las variables de interés de manera que sean accesibles tras la ejecución del algoritmo. Una vez se ha obtenido la solución, se muestran las figuras que la representan, que incluyen las posiciones y velocidades de las partículas y sus densidades.

El algoritmo utilizado, en términos generales, es el siguiente:

1. Importar solución de la dinámica de fluidos.
2. Definir parámetros, funciones, fronteras y condiciones iniciales.
3. Calcular la intensidad luminosa de la posición de cada partícula con (18).
4. Calcular la densidad celular mediante un paso del método de Euler con (19) y (20).
5. Calcular la densidad de la colonia con (21).
6. Interpolar la velocidad del fluido en las posiciones de las partículas.
7. Calcular la velocidad de las partículas con (22).
8. Calcular la posición de las partículas mediante un paso del método de Euler con (23).
9. Imponer condiciones de contorno (26) y (28).
10. Guardar variables de interés.
11. Si el paso de tiempo no es el último, volver a 3, y si lo es, finalizar.

3.2. Resultados de la simulación

La implementación presentada es suficientemente general para ser capaz de obtener soluciones a diversos escenarios que cumplan las características del modelo. Por ello, se plantean varios escenarios ficticios en los que se toman muestras de las cianobacterias en la masa de agua en cuestión una vez al mes. Se supone que la distribución inicial de las partículas es conocida y dada por las mediciones del mes anterior, las condiciones de luminosidad han sido medidas y el objetivo es simular el transporte de las colonias de cianobacterias a lo largo del periodo de tiempo transcurrido para saber dónde tomar las siguientes muestras. Se ha seleccionado un periodo de tiempo tan extendido con el objetivo de que los efectos del transporte resulten evidentes, ya que no parece haber consenso todavía entre los expertos sobre la frecuencia de medición idónea dentro de lo realizable.

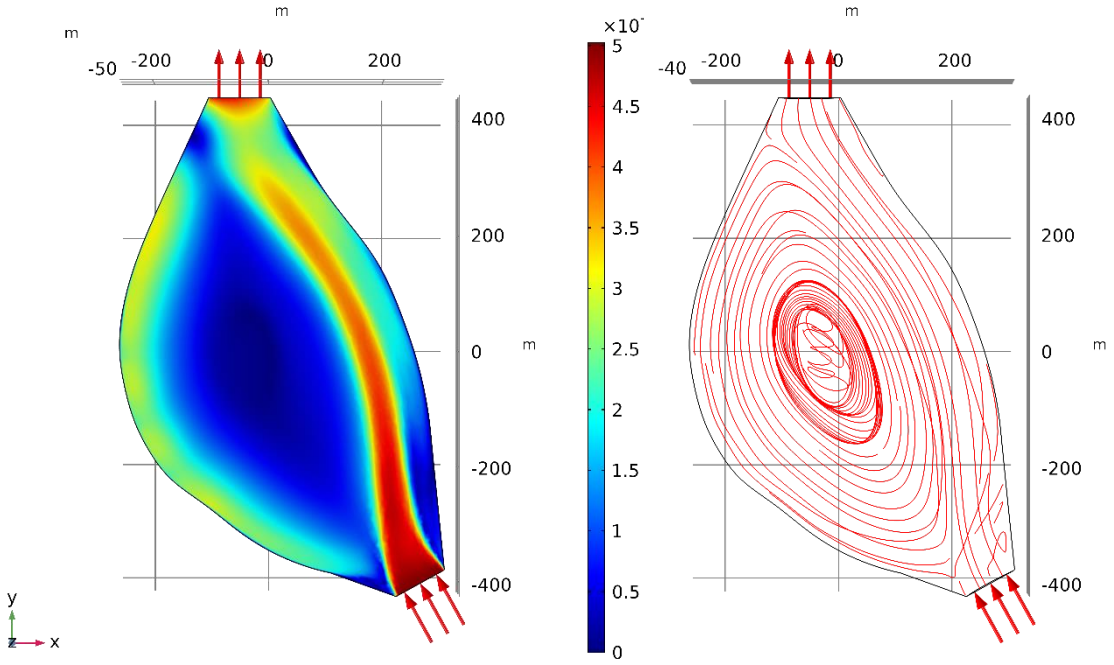
3.2.1. Simulación de la dinámica de fluidos

La primera parte de la simulación se realiza en COMSOL (ver Sección 3.1.1). Se presentan dos escenarios en el mismo lago, en los cuales se varía la velocidad del flujo de entrada. En el primer escenario, que se denominará EF1, la velocidad de entrada de agua a través de la frontera Γ_{in} es $\boldsymbol{u}_{\text{in}} = 5 \times 10^{-3}$ m/s (5.23 m^3/s de flujo de entrada), y en el segundo, EF2, es $\boldsymbol{u}_{\text{in}} = 5$ m/s (5.23×10^3 m^3/s de flujo de entrada). Nótese la diferencia con el anterior caso en el orden de las magnitudes, de 10^3. En ambos casos la malla utilizada finalmente tiene un tamaño de elemento *Fino* y es controlada por la física, con un total de 169972 elementos. La solución de la dinámica de fluidos se ha hallado utilizando seis mallas progresivamente más finas (de acuerdo con la técnica descrita en las Secciones 3.1.1.3 y 3.1.1.4). El volumen del dominio es de 79.35×10^6 m^3 y el área de la superficie del agua es de 317.25×10^3 m^2.

3.2.1.1. Escenario EF1

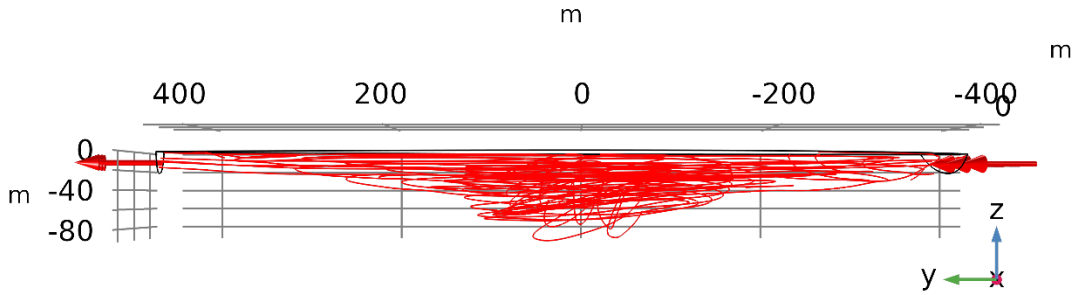
El primer escenario de flujo, como se ha comentado, muestra los efectos de un flujo de entrada lento en el lago en cuestión. Los parámetros físicos utilizados se muestran en la Tabla 2, y la velocidad del flujo de entrada es $\boldsymbol{u}_{\text{in}} = 5 \times 10^{-3}$ m/s. El tiempo total de cómputo combinado ha sido de 754 s (12 min 34 s), la mayor parte de este (un 58.22%) ha sido empleado en el estudio con la malla más fina.

Los resultados obtenidos para este escenario se exponen en la Figura 21. La escala de color muestra la magnitud de la velocidad del fluido en la superficie del flujo estacionario, y las líneas rojas muestran la dirección el flujo en todo el dominio tridimensional. Se observa cómo la mayor parte del agua que entra se dirige directamente hacia la salida, pero otra parte se ve dirigida hacia la zona de la izquierda, donde se forma un vórtice que lentamente dirige el agua hacia el centro del lago. Parte del agua consigue deslizarse bajo la corriente principal, volver a emerger en la zona de bajas velocidades de la derecha y, finalmente, unirse a esta y ser expulsada del dominio simulado, evitando así el estancamiento de agua. Esto proporciona, como veremos a continuación, una oportunidad para que las colonias de cianobacterias queden atrapadas y proliferen en el lago, ya que su flotabilidad natural puede hacer que la corriente de bajada tanto en el centro del vórtice como en su extremo inferior no sean suficientemente fuertes como para llevar las colonias hacia la salida.



(a) Magnitud de la velocidad en la superficie.

(b) Líneas de flujo en el plano horizontal.



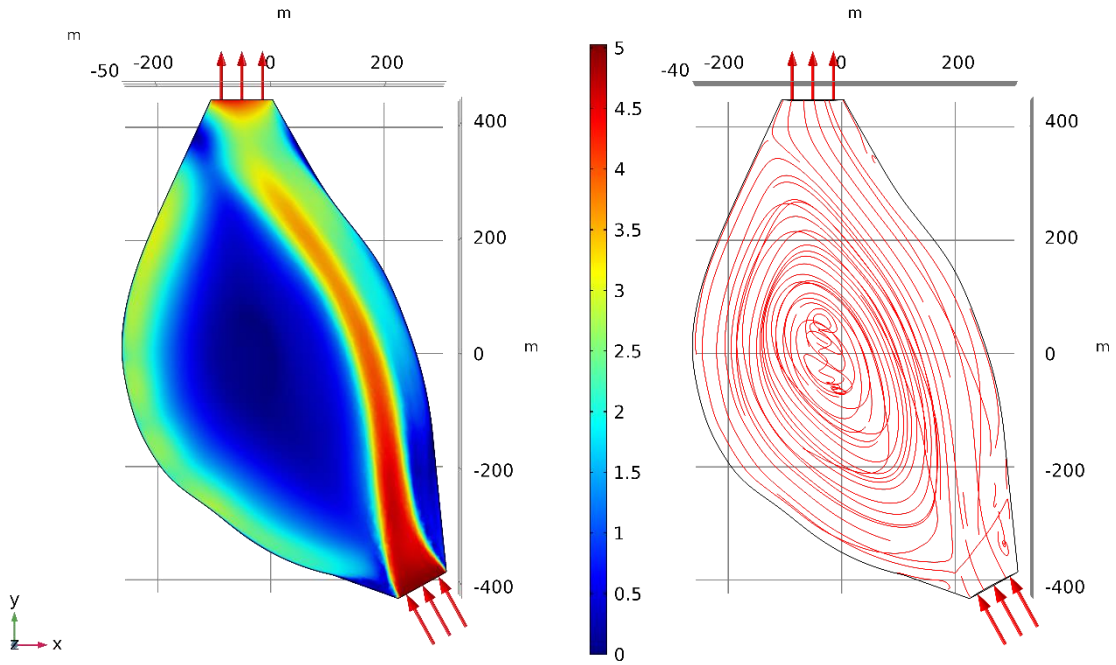
(c) Líneas de flujo en el plano vertical.

Figura 21: Solución de EF1.

3.2.1.2. Escenario EF2

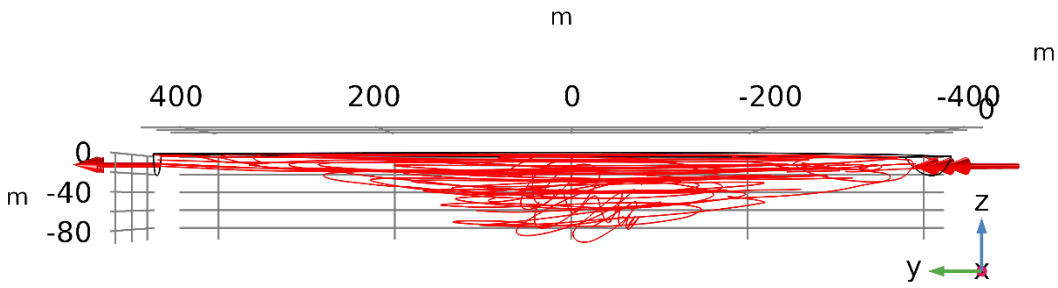
El segundo escenario de flujo del que se han obtenido resultados es muy similar al primero, pero difiere en que la velocidad de entrada es 1000 veces mayor, $\boldsymbol{u}_{\text{in}} = 5\ \text{m/s}$. El tiempo total de cómputo combinado ha sido de 743 s (12 min 23 s), y de nuevo la mayor parte de este (un 60.83% en este caso) ha sido empleado en el estudio con la malla más fina.

Los resultados son muy similares al caso anterior, pero se incrementa la velocidad de manera significativa, como puede verse en la barra de valores de Figura 22(a).



(a) Magnitud de la velocidad en la superficie.

(b) Líneas de flujo en el plano horizontal.



(c) Líneas de flujo en el plano vertical.

Figura 22: Solución de EF2.

3.2.2. Simulación del transporte de partículas

Los resultados del flujo se exportan a MATLAB, donde se ejecuta la segunda parte de la simulación, el transporte de partículas (ver Sección 3.1.2). Se plantean de nuevo varios escenarios (EP1 – EP3), que en esta ocasión se diferencian en la distribución inicial de las cianobacterias. En todos ellos la distribución inicial de las partículas es una distribución Gaussiana, y se diferencian en la media y su desviación estándar. En todos los casos se generan 500 partículas, y si alguna de ellas en el proceso de generación se encuentra fuera del dominio se descarta y se generan partículas adicionales hasta completar las 500 partículas válidas. Las densidades se asignan aleatoriamente de una distribución uniforme en el intervalo $\rho_{\text{cell}}^{0} \in [C_\rho, C_\rho + 50]$, que pronto se estabiliza en sus valores nominales. Además, se utilizan los parámetros físicos reflejados en la Tabla 2.

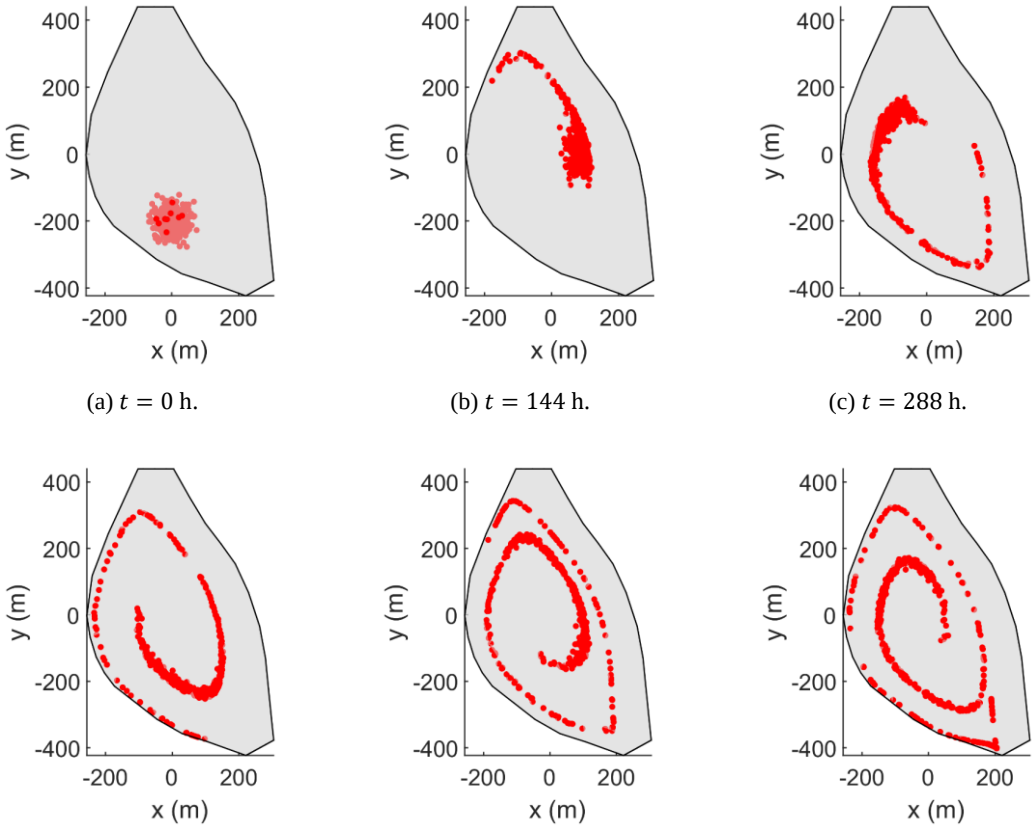
A continuación, se resuelven las ecuaciones pertinentes en cada paso de tiempo hasta alcanzar el fin del dominio temporal. En lugar de calcular la concentración de partículas para todo el dominio

y en cada paso de tiempo y utilizar este campo en el proceso de optimización de trayectorias, resulta más eficiente operar directamente sobre las posiciones de las partículas, ya que en ese caso solamente es necesario considerar las que se observan a lo largo de la trayectoria del USV. De esta manera, en lugar de calcular la concentración en un campo tetradimensional (tres dimensiones espaciales y una temporal), basta con calcular la concentración a lo largo de la trayectoria, lo cual es 1D solamente, ya que puede calcularse en el instante apropiado gracias a la parametrización de las splines seleccionada.

3.2.2.1. Escenario EP1

La distribución inicial de las partículas de este escenario es una distribución Gaussiana de reducida amplitud, centrada en $(0, -200, -5)^T$ y desviación estándar $(25, 25, 5)^T$, que se muestra en la Figura 23(a). Para obtener los resultados mostrados se ha utilizado la solución del flujo del escenario EF1, y el dominio temporal utilizado es el comprendido en el intervalo $t \in [0, 720]$ h, el equivalente a 30 días, con un paso de tiempo de 0.1 h ($N_t = 7200$). El tiempo de cómputo ha sido de 877 s (14 min 37 s).

En la Figura 23 se muestra la posición de las partículas vistas desde el eje z en diferentes momentos de la simulación, que se indican en los subtítulos. Cada partícula se ha representado con un punto rojo dentro del dominio definido por la geometría, en gris. En $t = 0$, arriba a la izquierda, la posición de las partículas se corresponde con la posición inicial $\boldsymbol{x}_p^0$, que se ha establecido como la distribución Gaussiana mencionada con anterioridad. A medida que avanza el tiempo el grupo de partículas se va dispersando en el plano horizontal, debido a las diferentes velocidades del fluido en diferentes puntos del dominio.



(a) $t = 0$ h. (b) $t = 144$ h. (c) $t = 288$ h.

(d) $t = 432$ h. (e) $t = 576$ h. (f) $t = 720$ h.

Figura 23: Posición de las partículas a lo largo del tiempo en EF1-EP1.

En la Figura 24 se representan las trayectorias de cada una de las partículas simuladas, y en la Figura 25 sus velocidades. Cada partícula se representa en un color, que es el mismo en todas las sub-figuras. Nótese que la posición vertical, controlada por la mayor influencia de la flotabilidad, tiende a cohesionar las partículas en ese eje haciendo que la gráfica de la última partícula representada se superponga a la del resto, mientras que en el plano vertical las partículas tienden a dispersarse debido a las diferencias en el flujo del agua en diferentes posiciones. Además, examinando las velocidades y posiciones en el plano horizontal, se descubre que en posiciones cercanas al centro de rotación la velocidad es menor, lo cual está de acuerdo con la velocidad del fluido presentada en la Figura 21 o la Figura 22, ya que la velocidad horizontal solamente recibe influencias de este campo.

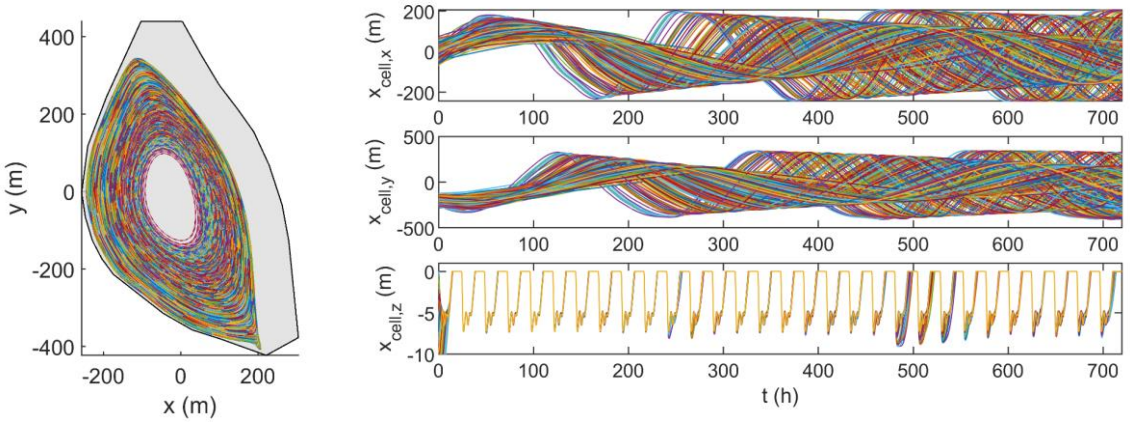


Figura 24: Trayectoria de las partículas en EF1-EP1.

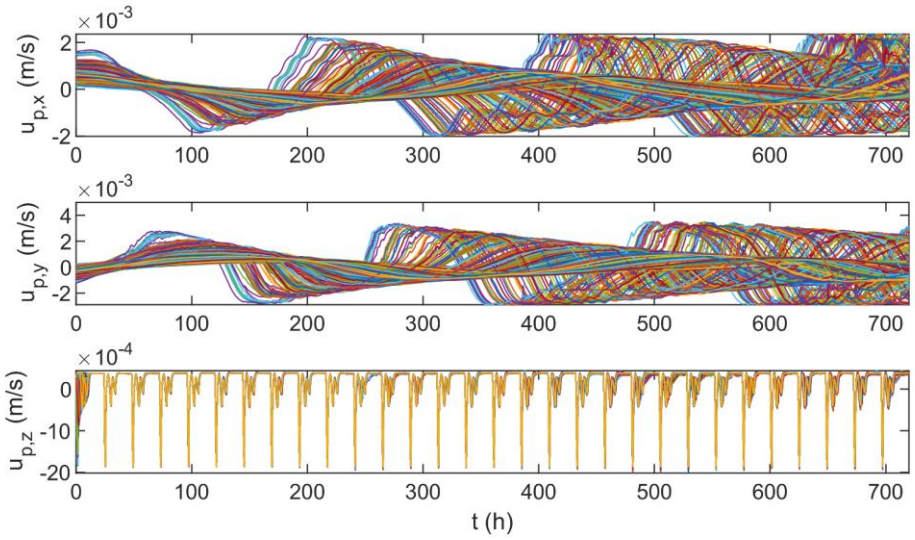


Figura 25: Velocidad de las partículas en EF1-EP1.

Observando el desplazamiento y la velocidad en el eje vertical se puede discernir una relación de proporcionalidad inversa con la densidad de las colonias, como se muestra en la Figura 26. A medida que la densidad baja, la partícula sube y, a medida que la densidad se incrementa, la partícula se hunde, como es de esperar. La línea roja discontinua denota la densidad del fluido. Si la densidad de la colonia se encuentra por encima de esta línea, la colonia se hunde, y si se encuentra por debajo, flota.

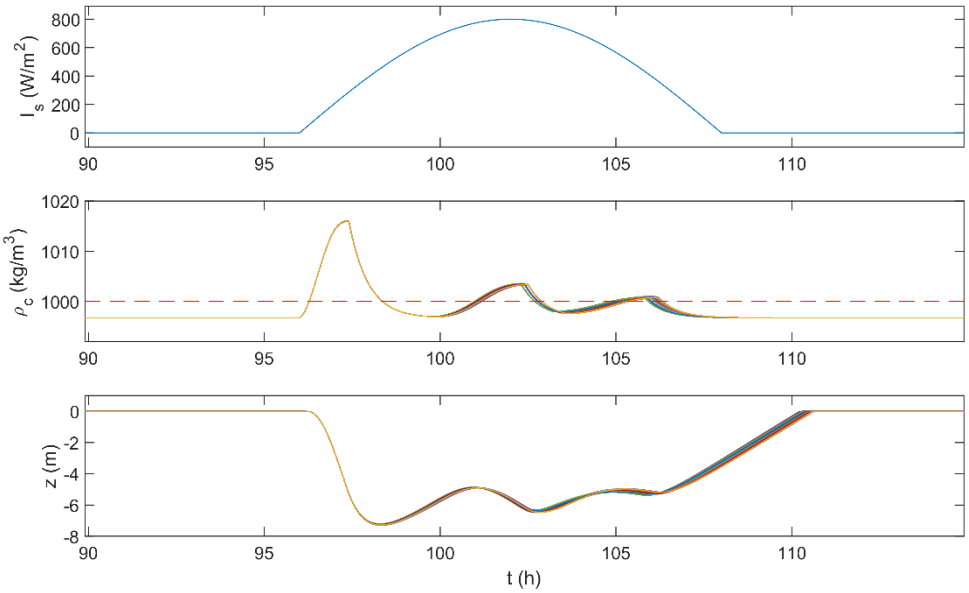
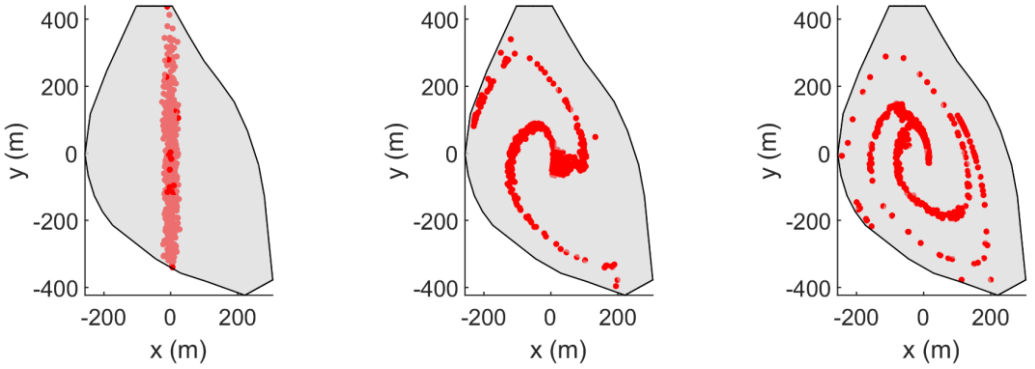


Figura 26: Relación entre la densidad de la colonia y la posición vertical a lo largo de un día.

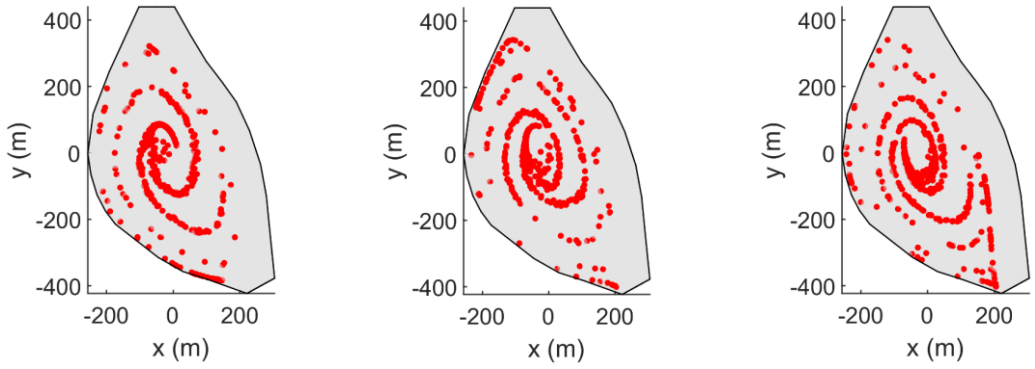
Es de notar que en situaciones como la de este escenario, donde el flujo del fluido en dirección vertical es prácticamente despreciable, las partículas no suelen alcanzar profundidades mayores a 8 m, lo cual es aproximadamente del mismo orden que las profundidades obtenidas en la Tesis de Medrano [18]. Realizar comparaciones directas con los resultados ofrecidos no es posible ya que en esta se incluyen los efectos del flujo turbulento, que modifican la trayectoria vertical de las cianobacterias. Sin embargo, tanto en estos resultados como en los expuestos en la literatura, se muestra la misma tendencia general a descender durante las horas de luz y ascender hasta la superficie durante la noche. Los resultados obtenidos también coinciden cualitativamente con los resultados ofrecidos en [44].

3.2.2.2. Escenario EP2

La distribución inicial en este caso es alargada en el eje y y estrecha en el eje x, formando una franja a lo largo del lago, como se muestra en la Figura 27. Su distribución gaussiana viene dada por $\mu = (0, -50, -2)^T$ y $\sigma = (10, 100, 2)^T$. En el resto de sub-figuras se observa un comportamiento intrínsecamente similar al exhibido en el escenario anterior, donde parte de las partículas quedan atrapadas en el vórtice. La Figura 28 muestra que a pesar de que la trayectoria horizontal difiera, el desplazamiento vertical es prácticamente igual que en EP1, y también se aprecia que las partículas situadas en la corriente principal del flujo se dirigen directamente a la salida del lago. Como flujo para obtener estos resultados se ha vuelto a utilizar el escenario EF1, y se ha mantenido el mismo dominio temporal. El tiempo de cómputo ha sido de nuevo 877 s (14 min 37 s).



(a) $t = 0$ h. (b) $t = 144$ h. (c) $t = 288$ h.



(d) $t = 432$ h. (e) $t = 576$ h. (f) $t = 720$ h.

Figura 27: Posición de las partículas a lo largo del tiempo en EF1-EP2.

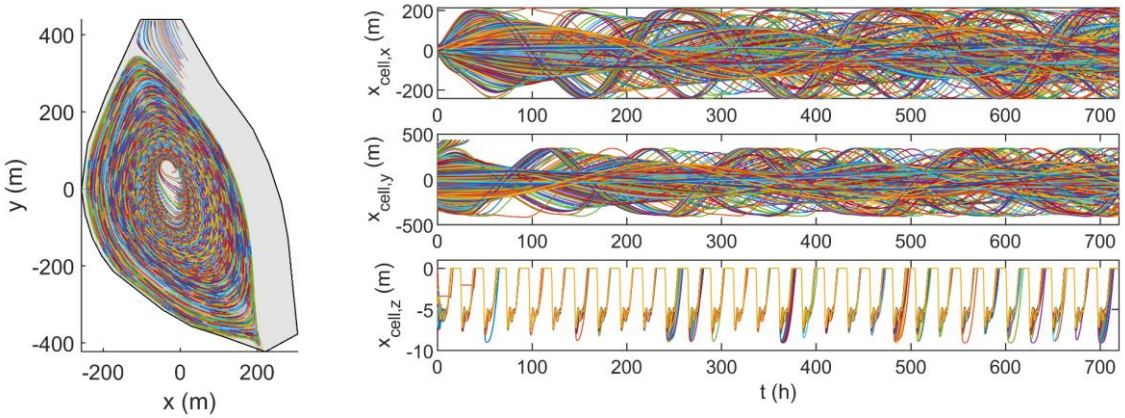
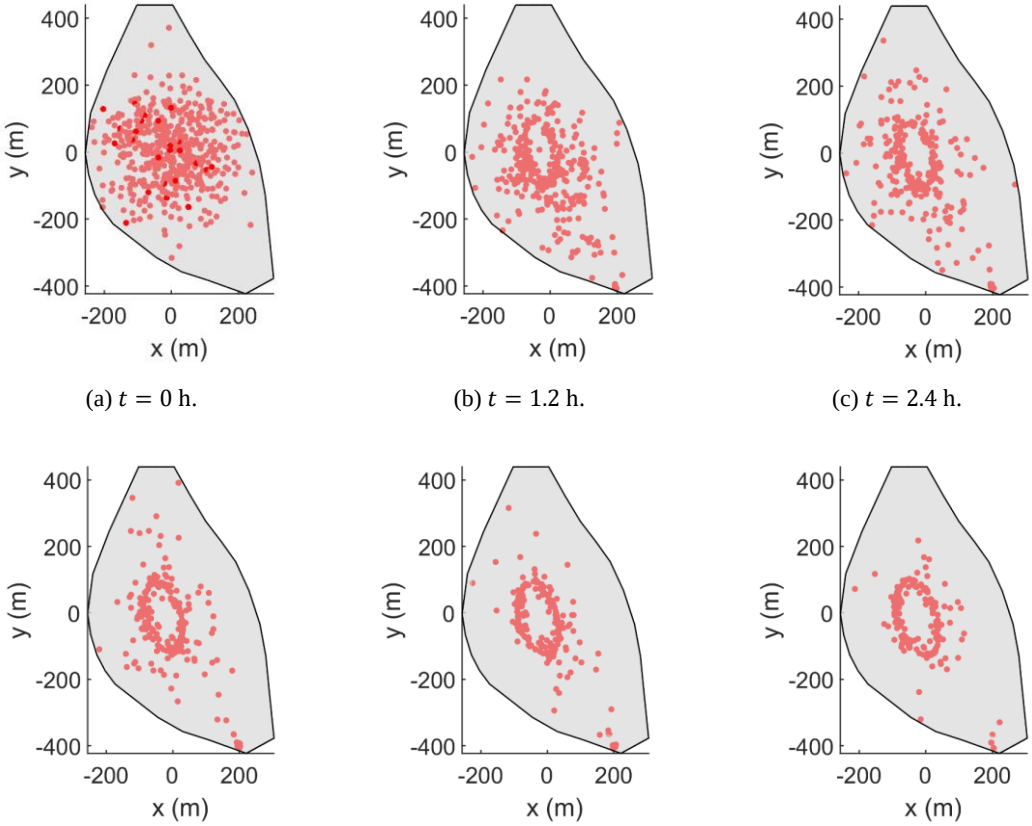


Figura 28: Trayectoria de las partículas en EF1-EP2.

3.2.2.3. Escenario EP3

El último de los escenarios modelados presenta una distribución de partículas inicial muy amplia en ambos ejes horizontales, con lo cual, con la Figura 29, se obtiene una idea más general del transporte en las diferentes zonas de la masa de agua simulada. La distribución de Gauss, que determina la posición inicial de las partículas, viene dada por $\mu = (0, 0, -1)^T$ y $\sigma = (100, 100, 1)^T$. En esta ocasión se ha utilizado el flujo más rápido, EF2, de modo que el desplazamiento en el plano horizontal resulta más evidente en una escala de tiempo reducida,

como puede observarse en la Figura 30. Además, en el eje vertical se muestran trayectorias mucho más profundas que las observadas en los escenarios anteriores, lo cual está causado por la mayor intensidad de la corriente, que es capaz de vencer la flotabilidad natural de las partículas en ciertas zonas del dominio. Debido a que el movimiento de las partículas es más rápido, el paso de tiempo ha de ser más corto, y para mantener el tiempo de ejecución y la cantidad de memoria utilizada es necesario reducir el dominio temporal, que en este caso es el comprendido en el intervalo $t \in [0, 6]$ h, con un paso de tiempo de 5×10^{-4} h ($N_t = 12000$). El tiempo de cómputo ha sido de 1519 s (25 min 19 s).



(a) $t = 0$ h. (b) $t = 1.2$ h. (c) $t = 2.4$ h.

(d) $t = 3.6$ h. (e) $t = 4.8$ h. (f) $t = 6$ h.

Figura 29: Posición de las partículas a lo largo del tiempo en EF2-EP3.

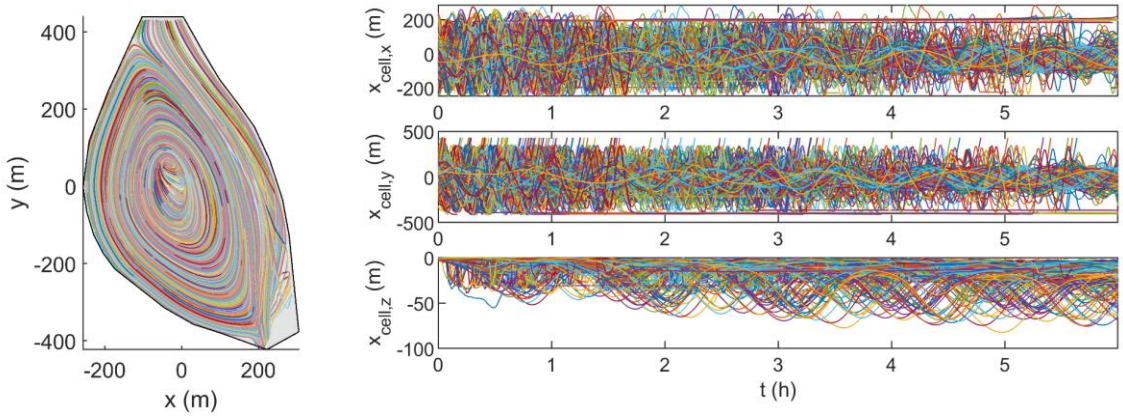


Figura 30: Trayectoria de las partículas en EF2-EP3.

4. Planificación de Trayectorias de un USV

El objetivo de la segunda parte del proyecto trata de la optimización de la trayectoria de un vehículo de superficie no tripulado (Unmaned Surface Vehicle, USV) provisto de una sonda para la recogida de muestras de cianobacterias de manera que se maximice la cantidad de muestras recogidas. Se ha optado por utilizar una codificación de las trayectorias basada en splines, con la intención de que resulte más adecuada para su aplicación en algoritmos de optimización que otras alternativas, como las secuencias de señales de control [45] o la utilización de trayectorias predefinidas [46]. En las páginas sucesivas se presenta un análisis detallado de las splines utilizadas, los objetivos a cumplir en este problema de optimización y las restricciones.

4.1. Trayectorias mediante splines

Una spline es un tipo de curva definida a trozos por funciones polinómicas mediante una serie de nodos colocados en al menos dos dimensiones. Habiendo decidido codificar la trayectoria en forma de spline, el problema pasa a ser qué tipo de spline utilizar. Existen multitud de tipos con numerosas características.

En cuanto a sus comportamientos en los extremos se distinguen las splines naturales, cuyas segundas derivadas son nulas en los extremos; las splines sujetas (clamped), en las cuales la primera derivada se especifica en los extremos y, finalmente, splines not-a-knot, que no imponen restricciones en los extremos, pero la tercera derivada ha de ser continua en esos puntos. En este problema sería conveniente que las dos primeras derivadas pudieran ser establecidas en cero, de manera que al utilizar la spline como trayectoria de un USV, su velocidad y aceleración inicial y final sea cero.

En cuanto a su orden, la mayor parte de splines utilizadas comúnmente son splines cúbicas, lo cual quiere decir que los polinomios que la forman son de orden 3. A pesar de que lo más común es utilizar este orden, no existe ninguna razón que impida su construcción mediante el uso de polinomios de un orden mayor, como se ha hecho en este Trabajo de Fin de Máster, o incluso menores, a pesar de que esto puede limitar la continuidad de la spline.

En cuanto a la influencia que tiene cada nodo en la trayectoria completa, se distinguen globales y locales. En splines globales como son las naturales, sujetas y not-a-knot, la modificación de un nodo influye en la trayectoria entera, mientras que otras splines como las beta-splines (de las cuales las b-splines son un subconjunto) o las splines cardinales (que incluyen las splines de Catmull-Roll) son locales y la modificación de un nodo solamente afecta a su entorno inmediato. Para codificar trayectorias que han de ser optimizadas resulta conveniente utilizar splines locales, ya que la modificación de un nodo no destruye el progreso en otras zonas de la spline.

En cuanto a si pasan por los nodos, se distinguen splines de interpolación, como cualquiera de las mencionadas hasta ahora a excepción de las beta-splines, y de no interpolación, como las beta-splines o las curvas de Bézier. Las splines de interpolación pasan por los nodos que las definen, mientras que el otro tipo no lo hace. Es conveniente, al menos en esta aplicación, el uso de splines de interpolación ya que limitan el dominio y facilitan la generación de puntos a lo largo de la spline.

En cuanto a su continuidad se distinguen varios tipos también. Los polinomios que forman cada intervalo pertenecen a la clase C^{∞}, ya que son infinitamente derivables, de modo que las limitaciones en cuanto a su continuidad solamente son palpables en las conexiones entre los polinomios. De entre las splines más comúnmente utilizadas, la mayor parte de splines son C^2, incluyendo las splines cúbicas naturales, sujetas, not-a-knot y las beta-splines, a excepción de las splines cardinales, que son C^1. Con el objetivo de definir trayectorias para el USV que puedan ser directamente utilizadas como señales de control, sería conveniente que la spline utilizada fuera C^2, de manera que la aceleración fuera continua.

Por estas razones lo ideal sería la utilización de una spline cuyas dos primeras derivadas sean cero en los extremos, que sea una spline local, de interpolación y que sea de clase C^2. En este Trabajo de Fin de Máster no se ha logrado cumplir todas las condiciones, ya que la spline utilizada finalmente es global. Esto es debido a que las splines locales, o bien son especificadas no solo por la posición de sus nodos sino también por la primera derivada en cada uno de ellos (lo cual incrementa el número de variables de decisión), o bien resulta necesario estimar la derivada. Esta última opción complica en cierto modo su implementación y abre otras posibilidades en cuanto a cómo aproximar las derivadas. Además, la implementación de splines cardinales de orden superior incurre en sistemas de mayor tamaño, debido a que están formadas por la combinación lineal de varias funciones de Hermite como bases, lo cual resulta más lento de resolver que en el caso de splines globales, que solamente están formada por una base. Para cumplir con el resto de objetivos, se han desarrollado las splines 4-3-4, logrando cumplir el resto de las condiciones incrementando el orden de los polinomios que forman los segmentos de los extremos de la spline.

4.1.1. Cálculo de las splines 4-3-4

Dado un set de N_n nodos tridimensionales $\boldsymbol{x}_i \in \mathbb{R}^3$ a través de los cuales la spline debe pasar en los instantes $t_i \in \mathbb{R}$, donde $i \in \{1, \dots, N_n\}$, se crea una curva definida por partes por polinomios $P(t)\colon \mathbb{R} \to \mathbb{R}^3$ de tal manera que $P(t_i) = x_i$. La spline $S(t)$ está definida como la unión de los polinomios que la forman.

$$S(t) = \begin{cases} P_1(t) & \text{en } t \in [t_1, t_2]\,, \\ P_2(t) & \text{en } t \in [t_2, t_3]\,, \\ \vdots & \vdots \\ P_{N_n-1}(t) & \text{en } t \in \left[t_{N_n-1}, t_{N_n}\right]. \end{cases} \tag{72}$$

En los extremos de la trayectoria, la spline ha de cumplir

$$\begin{cases} P_1(t_1) = x_1\,, \\ P_1'(t_1) = 0\,, \\ P_1''(t_1) = 0\,, \end{cases} \quad \text{y} \quad \begin{cases} P_{N_n-1}(t_{N_n}) = x_{N_n}\,, \\ P_{N_n-1}'(t_{N_n}) = 0\,, \\ P_{N_n-1}''(t_{N_n}) = 0\,, \end{cases} \tag{73}$$

y en los nodos intermedios,

$$\begin{cases} P_{i-1}(t_i) = P_i(t_i) = x_i\,, \\ P_{i-1}'(t_i) = P_i'(t_i)\,, \\ P_{i-1}''(t_i) = P_i''(t_i)\,. \end{cases} \tag{74}$$

Esto quiere decir simplemente que la spline tiene posición, velocidad y aceleración continuas ($S(t) \in C^2$) y que su velocidad y aceleración en los extremos es nula.

El mínimo requerimiento para que una spline cumpla estos criterios impuestos es que los polinomios utilizados sean todos de orden 3, excepto aquellos que contienen los extremos, que han de ser de orden 4, razón por la cual han sido denominadas en este proyecto splines 4-3-4. Un diagrama de este tipo de splines puede encontrarse en la Figura 31. A pesar de que la mayoría de los polinomios que forman la spline son cúbicos, el orden de una spline queda definido por el polinomio de más alto orden, de modo que en este caso la spline será de orden 4.

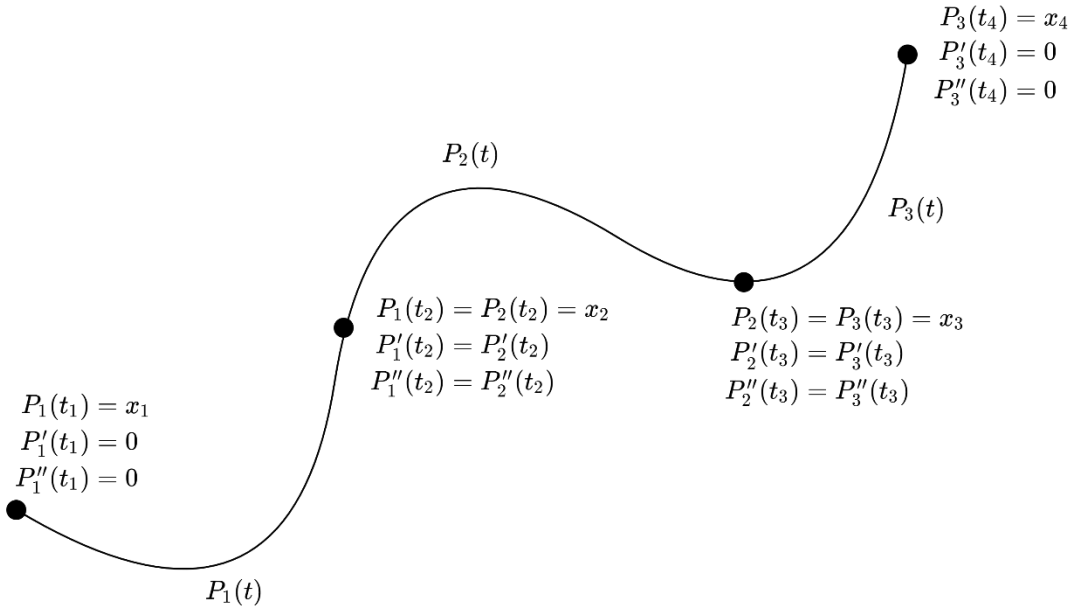


Figura 31: Diagrama de una spline 4-3-4.

Dado que los ejes espaciales son ortogonales, es posible diseñar un método que calcule una spline en una sola dimensión y aplicarlo a cada una de las dimensiones espaciales por separado, manteniendo la variable temporal como factor de unificación. Los polinomios cúbicos se definen como

$$P_i^3(t) = c_3^i(t - t_i)^3 + c_2^i(t - t_i)^2 + c_1^i(t - t_i) + c_0^i \,, \tag{75}$$

y los de cuarto orden como

$$P_i^4(t) = c_4^i(t - t_i)^4 + c_3^i(t - t_i)^3 + c_2^i(t - t_i)^2 + c_1^i(t - t_i) + c_0^i \,, \tag{76}$$

donde c_j^i es el coeficiente perteneciente al polinomio i en el término de orden j. También es posible hallar sus primeras derivadas,

$$P'^3_i(t) = 3c_3^i(t - t_i)^2 + 2c_2^i(t - t_i) + c_1^i \,, \tag{77}$$

$$P'^4_i(t) = 4c_4^i(t - t_i)^3 + 3c_3^i(t - t_i)^2 + 2c_2^i(t - t_i) + c_1^i \,, \tag{78}$$

y segundas derivadas,

$$P''^3_i(t) = 6c_3^i(t - t_i) + 2c_2^i \,, \tag{79}$$

$$P''^4_i(t) = 12c_4^i(t - t_i)^2 + 6c_3^i(t - t_i) + 2c_2^i \,. \tag{80}$$

Para encontrar los valores de los coeficientes, puede resolverse el siguiente sistema lineal, que es resultado de la unión de las ecuaciones (73) y (74).

$$\begin{bmatrix} B_{11} & & & & & \\ B_{21} & B_{22} & & & & \\ & B_{32} & B_{33} & & & \\ & & & \ddots & & \\ & & & B_{n-1,n-2} & B_{n-1,n-1} & \\ & & & & B_{n,n-1} & \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \\ \vdots \\ \boldsymbol{c}_{N_n-2} \\ \boldsymbol{c}_{N_n-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_{N_n-1} \\ \boldsymbol{b}_{N_n} \end{bmatrix}, \tag{81}$$

donde $\boldsymbol{c}_i \in \mathbb{R}^4 \cup \mathbb{R}^5$ son los coeficientes del polinomio número i,

$$\boldsymbol{c}_i = \begin{cases} [c_4^i \quad c_3^i \quad c_2^i \quad c_1^i \quad c_0^i]^T & \text{si } i = \{1, N_n - 1\}, \\ [c_3^i \quad c_2^i \quad c_1^i \quad c_0^i]^T & \text{si } i = \{2, \dots, N_n - 2\}, \end{cases} \tag{82}$$

$\boldsymbol{b}_i \in \mathbb{R}^3 \cup \mathbb{R}^4$ son el lado derecho de las ecuaciones (73) y (74) en el nodo i,

$$\boldsymbol{b}_i = \begin{cases} [x_i \quad \dot{x}_i \quad \ddot{x}_i]^T & \text{si } i = \{1, N_n\}, \\ [x_i \quad x_i \quad 0 \quad 0]^T & \text{si } i = \{2, \dots, N_n - 1\} \end{cases} \tag{83}$$

y B_{ij} son matrices reales de la forma

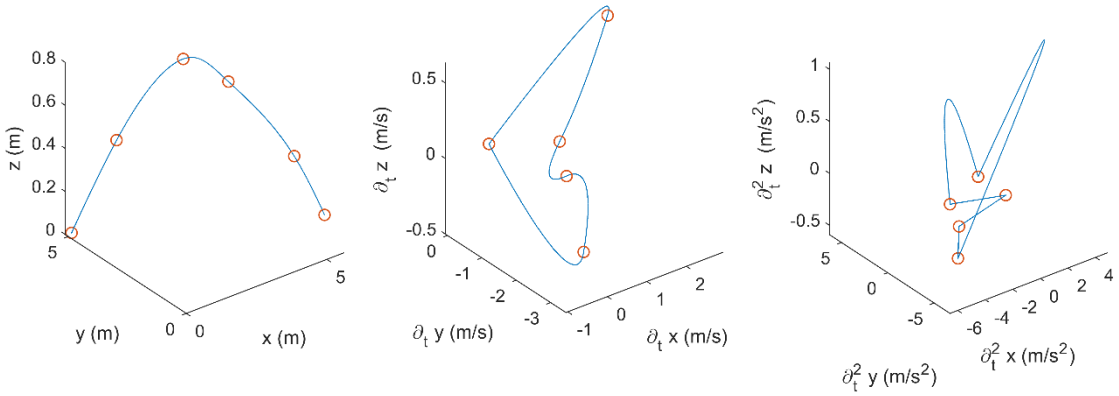
$$B_{ij} = \begin{cases} \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 & 0 \end{bmatrix} & \text{si } i = j = 1 \\ \begin{bmatrix} h_{i-2}^4 & h_{i-2}^3 & h_{i-2}^2 & h_{i-2} & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 4h_{i-2}^3 & 3h_{i-2}^2 & 2h_{i-2} & 1 & 0 \\ 12h_{i-2}^2 & 6h_{i-2} & 2 & 0 & 0 \end{bmatrix} & \text{si } i = j + 1 = 2 \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & -2 & 0 & 0 \end{bmatrix} & \text{si } i = j \neq \{1, N_n - 1\} \\ \begin{bmatrix} h_{i-2}^3 & h_{i-2}^2 & h_{i-2} & 1 \\ 0 & 0 & 0 & 0 \\ 3h_{i-2}^2 & 2h_{i-2} & 1 & 0 \\ 6h_{i-2} & 2 & 0 & 0 \end{bmatrix} & \text{si } i = j + 1 \neq \{2, N_n\} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -2 & 0 & 0 \end{bmatrix} & \text{si } i = j = N_n - 1 \\ \begin{bmatrix} h_{i-2}^4 & h_{i-2}^3 & h_{i-2}^2 & h_{i-2} & 1 \\ 4h_{i-2}^3 & 3h_{i-2}^2 & 2h_{i-2} & 1 & 0 \\ 12h_{i-2}^2 & 6h_{i-2} & 2 & 0 & 0 \end{bmatrix} & \text{si } i = j + 1 = N_n \\ 0 & \text{en otro caso} \end{cases} \tag{84}$$

donde se define $h_i = t_{i+1} - t_i$.

Debido al gran número de ceros de la matriz, esta puede considerarse hueca (sparse), de modo que la resolución del sistema pueda ser realizada de manera más eficiente. Como se ha comentado ya, este es el caso más sencillo de este tipo de spline, y es la razón por la cual este tipo de splines ha sido seleccionada, ya que tipos más complejos introducen variables de decisión al proceso de optimización, que a su vez se traducen como un aumento en la dimensionalidad del espacio de búsqueda, lo cual no es para nada deseable. Sin embargo, existirían otras splines que también

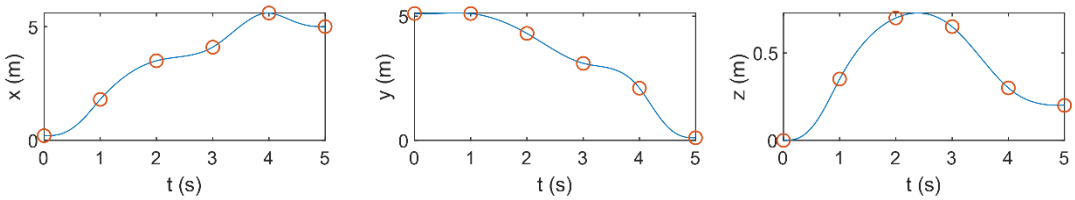
cumplirían los requisitos, como una spline puramente de orden 4 que utilizara las velocidades en los nodos como parámetros, no solo las posiciones.

A modo de ejemplo, en la Figura 32(a) se muestra una spline tridimensional que pasa por los cuatro puntos marcados en rojo en la misma, mientras que en la sub-figura central y en la de la derecha se muestra la velocidad y aceleración asociada a dicha spline, respectivamente. Además, en la Figura 33 se muestran el desplazamiento, la velocidad y la aceleración a lo largo del tiempo, para cada uno de los ejes. En ella se puede ver como las aceleraciones son lineales excepto en los polinomios de los extremos de la spline.

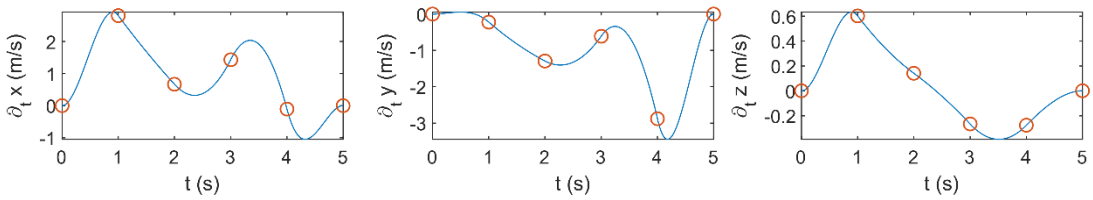


(a) Posición. (b) Velocidad. (c) Aceleración.

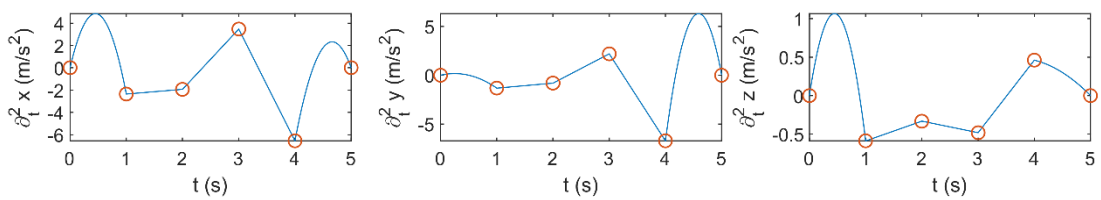
Figura 32: Ejemplo de una spline 4-3-4.



(a) Componentes de la posición.



(b) Componentes de la velocidad.



(c) Componentes de la aceleración.

Figura 33: Componentes de la spline de la Figura 32.

4.2. Objetivos de la optimización

Los objetivos del problema establecen cómo de buena es una trayectoria particular. El objetivo principal es que la trayectoria pase cerca del mayor número de partículas posibles, y los objetivos secundarios son la minimización de la duración y la distancia de la trayectoria. Ambos objetivos secundarios ayudan a obtener una trayectoria más limpia y permiten la utilización más efectiva de los USV.

4.2.1. Duración de la trayectoria

Se comienza por el objetivo más sencillo de los tres, la duración de la trayectoria. Al describir las trayectorias de acuerdo a parametrizaciones temporales, y además tener los extremos de la curva fijos en espacio, resulta sencillo comprobar la duración total de la trayectoria, ya que se trata simplemente de $t_{N_n} - t_1$, utilizando la notación de la sección anterior. Además, con el objetivo de que el sistema (81) esté bien condicionado y no dé problemas en su resolución, se fuerza a que $t_1 = 0$ y solamente se aplique un desfase temporal a la hora de extraer las posiciones de las partículas. De este modo, los instantes de tiempo utilizados en la optimización son del orden del resto de coordenadas del problema.

El inconveniente que esto tiene es que la optimización queda restringida a empezar en un instante dado, en lugar de buscar el momento idóneo a lo largo del periodo simulado; lo cual, por otra parte, puede verse como una ventaja, ya que permite al equipo que despliegue el USV comenzar la misión en el momento que ellos decidan.

4.2.2. Distancia de la trayectoria

El siguiente objetivo a optimizar se trata de la distancia total cubierta por la trayectoria. Este trabaja en contra del objetivo principal de incrementar la cantidad de partículas vistas, ya que un recorrido mayor implica mayores posibilidades de encuentros. Sin embargo, no incluir este objetivo tiene el efecto adicional de permitir trayectorias demasiado serpenteantes, que incrementan ligeramente la cantidad de partículas vistas a costa de una trayectoria ineficiente y de una elevada duración.

La longitud de una spline, como la de cualquier función continua, puede hallarse mediante la integral sobre la línea de la unidad, es decir,

$$l = \int_C ds = \int_a^b \|S'(t)\|\, dt\,, \tag{85}$$

donde $l \in \mathbb{R}^+$ es la longitud de la curva $C \in \mathbb{R}^n$ parametrizada por la spline $S(t)$ en el intervalo $[a, b]$. MATLAB proporciona una función de integración que es capaz de resolverlo utilizando técnicas de integración numérica. Sin embargo, con el objetivo de reducir el tiempo de ejecución, también se ha probado la aproximación

$$l \approx \sum_{i=1}^{N_s} \|S(\tau_i) - S(\tau_{i-1})\|, \tag{86}$$

donde se ha dividido la spline $S(t)$ en N_s pedazos cuyos extremos son miembros sucesivos de $t \in \{\tau_0, \dots, \tau_{N_s}\}$. Se comprueba, tal y como se muestra en la Tabla 3, que para obtener una aproximación de calidad, el tiempo de ejecución es mucho más lento al utilizar la aproximación por sumatorio, de modo que no se recomienda su uso. Además, en la última columna se ve reflejado que los resultados de la aproximación por sumatorio son siempre menores que los de la integración. Esto es de esperar ya que la aproximación es lineal y salta partes de la curva, mientras

que los métodos más complejos de integración numérica suelen compensar mediante infra y sobreestimaciones del resultado (como el esquema de integración centrado o el trapezoidal). Además, la convergencia de la integración numérica es más rápida en términos del tamaño del paso, y al incrementar el parámetro N_s en el sumatorio de (86), se observa que los resultados se aproximan a los proporcionados por `integral` de MATLAB, pero a costa de un mayor tiempo de computación, lo cual indica que la solución de la integral numérica es más precisa que la aproximación. Estas estadísticas han sido obtenidas al calcular por cada uno de los métodos la longitud total de 100 splines 4-3-4 generadas aleatoriamente. En la aproximación por sumatorio los puntos de muestra se encuentran equiespaciados en tiempo.

Tabla 3: Comparación de los métodos de integración de distancia.

	Tiempo medio de ejecución	Varianza del tiempo de ejecución	Longitud media de la spline
`integral` de MATLAB	6.0868×10^{-3} s	1.1752×10^{-5} s^2	131.4406 m
Aproximación por sumatorio ($N_s = 1000$)	1.6823×10^{-1} s	3.8699×10^{-5} s^2	131.3214 m
Aproximación por sumatorio ($N_s = 5000$)	8.3572×10^{-1} s	2.4214×10^{-4} s^2	131.4360 m

4.2.3. Concentración de partículas a lo largo de la trayectoria

Finalmente se expone el objetivo principal de la optimización, la concentración de partículas a lo largo de la trayectoria, ya que el objetivo principal de la misión del USV es caracterizar las zonas donde hay una elevada concentración de cianobacterias. Sin embargo, existen varias maneras de calcular dicho objetivo, y cada una de ellas presenta algunas ventajas y desventajas. Una opción es integrar la concentración a lo largo de la trayectoria, lo cual puede hacerse o bien en un número de puntos test correspondientes al lugar donde el USV tomaría periódicamente las muestras a lo largo de su trayectoria, o bien utilizando esos puntos test para integrar numéricamente el campo de concentraciones a lo largo de la trayectoria completa. Otra opción es utilizar funciones continuas de densidad basadas en la posición de las partículas. La última opción, y la utilizada en la versión final del proyecto, es comprobar si cada una de las partículas ha sido vista a lo largo de la trayectoria. Cada opción se describe a continuación.

4.2.3.1. Concentración en puntos test

La opción más intuitiva es seleccionar N_s puntos test a lo largo de la trayectoria de la spline, dados por los instantes de tiempo $t \in \{\tau_1, \dots, \tau_{N_s}\}$. Cada uno de estos puntos se corresponde con las coordenadas espaciotemporales donde el USV tomará muestras a lo largo de la misión. La concentración en cada uno de estos puntos puede ser calculada como

$$\varrho_p^i = \frac{n_V^i}{V_i}, \qquad i \in \{1, \dots, N_s\} \tag{87}$$

donde $\varrho_p^i \in \mathbb{R}^+$ es la concentración de partículas en el punto i correspondiente al punto de la spline $S(\tau_i)$, y $n_V^i \in \mathbb{N}$ es la cantidad de partículas con coordenadas dentro del set volumétrico $\mathcal{V}_i$, que tiene un volumen V_i. Una opción sencilla es definir el volumen como una esfera en torno a la posición de la sonda del USV que mantenga siempre el mismo tamaño. De esta manera $V_1 = \cdots = V_{N_s}$ y puede ajustarse el radio en el cual las partículas pueden ser detectadas. Dado un radio $r \in \mathbb{R}^+$, el volumen de la esfera V_i queda definido como

$$V = \frac{4}{3}\pi r^3 \tag{88}$$

y el set en el cual se cuentan las partículas es

$$\mathcal{V}_i = \{\boldsymbol{x} \in \mathbb{R}^3 \mid \|\boldsymbol{x} - S(\tau_i)\| \leq r\}, \qquad i \in \{1, \dots, N_s\}. \tag{89}$$

Otra opción es definir el volumen en forma de elipsoide, de manera que se pueda definir un radio diferente en cada uno de los ejes. Esto es particularmente útil para ayudar al algoritmo de optimización a alinearse en el eje vertical, que de utilizar una esfera, abarca prácticamente todo el dominio vertical y no realiza demasiados esfuerzos de optimización. Dados los semiejes del elipsoide r_1, r_2 y $r_3 \in \mathbb{R}^+$ en los ejes x, y y z respectivamente, y la spline $S(t) = \big(S_1(t), S_2(t), S_3(t)\big)^T$, el volumen del elipsoide es

$$V = \frac{4}{3}\pi r_x r_y r_z \tag{90}$$

y el set volumétrico queda definido como

$$\mathcal{V}_i = \left\{\boldsymbol{x} = (x_1, x_2, x_3)^T \in \mathbb{R}^3 \mid \sum_{d=1}^{3} \frac{\big(x_d - S_d(\tau_i)\big)^2}{r_d^2} \leq 1\right\}, \qquad i \in \{1, \dots, N_s\}. \tag{91}$$

Esta es la definición del volumen utilizada en este proyecto finalmente, donde se ha asignado $r_1 = r_2 \neq r_3$. En la Figura 34 se muestra un ejemplo de un volumen, donde la cruz azul representa el punto de muestreo $\boldsymbol{x} = S(\tau)$, la elipsoide centrada en este punto es $\mathcal{V}$, los puntos rojos son las partículas situadas en el exterior del volumen, y los verdes son las partículas situadas en el interior de $\mathcal{V}$ y que, por lo tanto, se cuentan para obtener n_V. En este caso los semiejes se han tomado $r_1 = r_2 = 2$ m y $r_3 = 1$ m, con lo cual el volumen resulta $V = 16\pi/3\ \text{m}^3$.

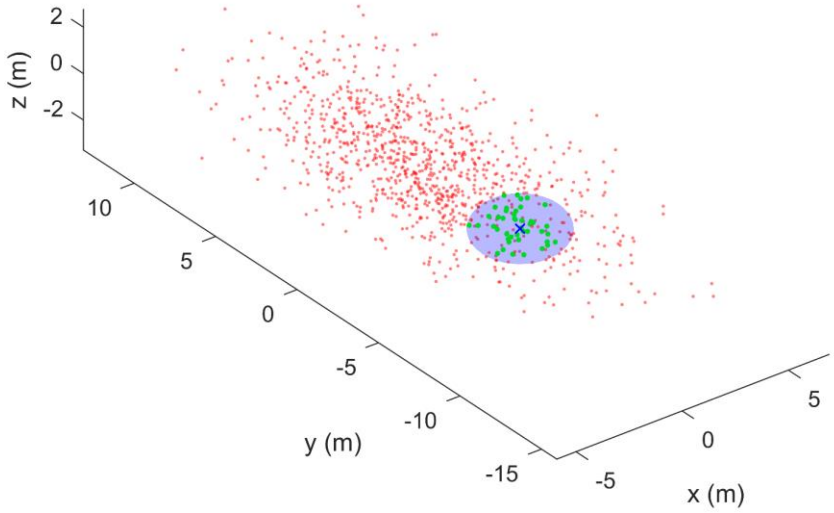


Figura 34: Ejemplo de volumen.

La ventaja de este método es que modela directamente la concentración que la muestras van a observar si las mediciones se realizan en varios puntos fijos, y no a lo largo de la trayectoria entera. Sin embargo, si este fuera el caso, se manifestaría una limitación importante de esta configuración: que los puntos de muestra no deberían ser necesariamente equiespaciados

temporalmente a lo largo de la trayectoria entera, ya que es posible que existan tramos de la trayectoria donde el objetivo sea simplemente desplazarse entre zonas de alta concentración. Por ello, surge el problema de elegir en qué momento colocar los puntos de muestreo. Lo más probable es que una codificación diferente de la trayectoria fuera más conveniente, de manera que se optimizaran, por un lado, los puntos de muestra y, por otro lado, la trayectoria que pasara por tales puntos.

4.2.3.2. Concentración a lo largo de la trayectoria

Si en lugar de realizarse la toma de muestras en puntos discretos se tomaran de forma continua, una opción sería utilizar las concentraciones en los puntos de muestra hallados con el anterior método para realizar una integración numérica a lo largo de la trayectoria, posiblemente incrementando el parámetro N_s para obtener variaciones más continuas entre los puntos. Entonces, con ϱ_p^i conocidos, la integración a lo largo de la trayectoria se realiza en forma de integración en línea,

$$\varrho_S = \int_C \frac{\varrho_p}{l}\, ds = \int_a^b \frac{\varrho_p}{l} \|S'(t)\|\, dt, \tag{92}$$

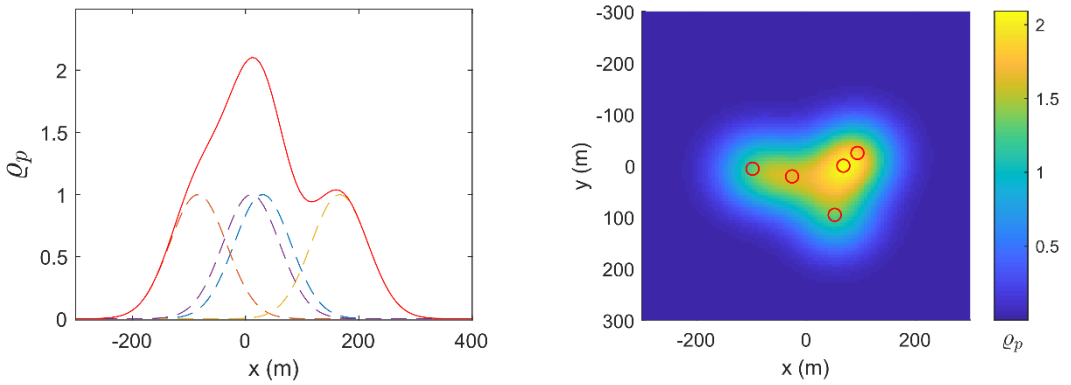
donde $\varrho_S \in \mathbb{R}^+$ es la concentración a lo largo de la trayectoria, y ϱ_p es el campo continuo de la densidad hallada con anterioridad. Lo más práctico es utilizar directamente ϱ_p^i en una integral numérica, utilizando cualquiera de las numerosas reglas de integración numérica, como puede ser la rectangular, la del punto medio, la del trapecio o la de Simpson.

Esta técnica tiene la ventaja de no depender tanto de la colocación de los puntos test, pero a su vez fuerza a incrementar la frecuencia con la cual los puntos test aparecen a lo largo de la trayectoria, lo cual incrementa el tiempo de computación. Para que la integral sea precisa es necesario tener una separación máxima cercana a $r_i/2$.

4.2.3.3. Funciones de densidad

Las opciones presentadas hasta ahora son bastante costosas en cuanto a tiempo de computación, ya que para cada punto test es necesario calcular las distancias y comprobar cuales se encuentran dentro del volumen pertinente. Además, tienen el inconveniente de dar información localizada, es decir, a no ser que la trayectoria pase por una zona del dominio donde se presente alguna partícula, no es posible saber en qué dirección se ha de avanzar para incrementar la concentración, lo cual puede presentar una gran dificultad en un dominio tan grande como el utilizado. Una solución a este problema es incrementar r_i, o hacer la búsqueda en dos pasos, el primero con un radio grande para encontrar las partículas, y el segundo con el radio efectivo de la sonda. Otra solución es utilizar funciones continuas de densidad que tengan un rango de decaimiento muy amplio, de modo que se cree un gradiente que conduzca al algoritmo hacia las zonas de mayor concentración.

En este proyecto se ha probado a utilizar una suma de funciones de Gauss. Estas funciones pueden ser ajustadas paramétricamente mediante la desviación estándar para reflejar mejor el radio de acción y, al no llegar nunca a cero, los extremos de las funciones pueden influir en la trayectoria desde puntos alejados del dominio. Obsérvese la Figura 35 donde la concentración ϱ_p es mayor donde hay más partículas pero no cae a cero inmediatamente. Las líneas discontinuas en la figura de la izquierda representan cada una de las funciones de Gauss sumadas y los puntos rojos en la figura de la derecha representan la posición de cada una de las partículas simuladas en un instante dado.



(a) Funciones en 1D. (b) Funciones en 2D.

Figura 35: Función de concentración.

La función de distribución, suma de una función de Gauss por cada partícula, está definida en un espacio $\mathbb{R}^n$ como

$$\varrho_p(\boldsymbol{x}) = \sum_{i=1}^{N_p} c \exp\left(-\sum_{j=1}^{n} \frac{\left(x_j - x_{p,j}^i\right)^2}{2\sigma_j^2}\right), \tag{93}$$

donde $\varrho_p \in \mathbb{R}^+$ es la función continua de concentración de partículas, $\boldsymbol{x} = (x_1, \dots, x_n)^T \in \mathbb{R}^n$ es la entrada de la función, $N_p \in \mathbb{N}$ es el número de partículas simuladas, $c \in \mathbb{R}^+$ es la amplitud (o el valor máximo) de la función de Gauss, $\boldsymbol{x}_p = \left(x_{p,1}, \dots, x_{p,n}\right)^T \in \mathbb{R}^n$ es la posición de las partículas simuladas, y $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_n) \in \mathbb{R}^n$ es la desviación estándar en cada una de las direcciones. Dado que las posiciones de las partículas varían a lo largo del tiempo, estas funciones pueden ser precalculadas, pero han de precalcularse en cada paso de tiempo simulado. En el ejemplo de la Figura 35 se ha tomado $\sigma_1 = \sigma_2 = 50$ y $c = 1$.

Utilizando esta función se puede emplear cualquiera de las técnicas anteriores para implementar la optimización, bien tomando el valor de la función en puntos test discretos, o bien calculando la integral en línea mediante (92), igualando las coordenadas espaciales en (93) a la spline, $\boldsymbol{x} = S(t)$.

4.2.3.4. Partículas vistas

La mayor inconveniencia de los métodos descritos hasta ahora es que solamente tienen en cuenta la densidad, bien sea en puntos concretos o en la trayectoria completa. Esto no impide que el algoritmo optimice una trayectoria que siga a lo largo de su traslación una zona de alta concentración, ignorando el resto de partículas que se encuentran más dispersas. Por ello, se ha seleccionado esta última opción, que consiste en comprobar cuáles de las partículas han estado a una distancia menor que una distancia dada r de la trayectoria en cualquier instante de tiempo. Esta es la técnica utilizada en [47].

Para ello, como en la primera opción, se toman una serie de puntos test a lo largo del spline que define la trayectoria del USV y en cada uno de ellos se comprueban las distancias entre el punto test y cada una de las partículas. Si cierta partícula se encuentra en el interior de la región $\mathcal{V}_i$, entonces se marca como vista. Una vez que una partícula ha sido vista no importa cuántas veces lo sea, ya que el marcador que indica que ha sido encontrada no sufre cambio alguno. Esto puede

conseguirse inicializándose a cero un vector de marcadores de longitud N_p y, a medida que se van encontrando partículas, cambiando el marcador correspondiente a uno. Puede actualizarse la lista mediante un operador booleano "or".

4.2.3.5. Comparativa de métodos

En la Tabla 4 se muestran las estadísticas de tiempos de cómputo de las diferentes opciones, sobre un test realizado con 100 splines aleatorias de 15 nodos y una distribución de partículas obtenida de una simulación dada. Tanto el instante inicial como los extremos de la spline se han mantenido fijos en todas las splines. Se aprecia que la opción más veloz es la primera por un estrecho margen, que utiliza la concentración en puntos test. Sin embargo, para evitar el estancamiento del proceso de optimización en una zona con una concentración alta de partículas, se sacrifica ligeramente el tiempo de cómputo y se implementa la última opción, que es capaz de discriminar entre cada una de las partículas.

Tabla 4: Comparación de los métodos de cálculo de la concentración.

	Tiempo medio de ejecución	Varianza del tiempo de ejecución
En puntos test	3.6697×10^{-1} s	9.8374×10^{-5} s^2
A lo largo de la trayectoria	3.7654×10^{-1} s	9.0805×10^{-5} s^2
Funciones de densidad	4.4902×10^{-2} s	2.3303×10^{-2} s^2
Partículas vistas	3.7392×10^{-1} s	9.2896×10^{-5} s^2

El mayor inconveniente de la técnica elegida es que, a diferencia de la función de densidades continua, su efecto es local, por lo que puede ser necesario aumentar el radio de búsqueda en dominios grandes o realizar la búsqueda en dos pasos, como se ha mencionado con anterioridad. Además, carece de significado físico, ya que un sensor real es incapaz de distinguir unas partículas de otras. Sin embargo, proporciona una buena herramienta a la hora de obtener los resultados que se buscan.

4.3. Restricciones

A parte de los objetivos, las restricciones juegan un papel fundamental en un problema de optimización. Restringen el dominio de búsqueda basándose en ciertas condiciones, que comúnmente tienen que ver con limitaciones físicas de los elementos del problema, como es el caso de la velocidad o aceleración máximas de un USV; restricciones operacionales, como la duración máxima; o restricciones de dominio físico, como la existencia de zonas prohibidas.

4.3.1. Velocidad y aceleración máxima del USV

Idealmente se podrían calcular trayectorias del USV sin restricciones en la velocidad y la aceleración, pero la realidad es que cualquier sistema físico las posee. En el caso de vehículos como los USV, estas limitaciones se encuentran normalmente medidas y comprobadas, de modo que su imposición en el modelo de optimización resulta necesaria y sencilla para el usuario, ya que los valores límite son conocidos y no es necesario estimarlos.

Al haber codificado las trayectorias como splines, cuya parametrización es temporal, resulta muy sencillo hallar las velocidades y aceleraciones en cualquier instante de tiempo. Como la trayectoria representa la posición del USV en un momento dado, la velocidad simplemente será su derivada y la aceleración la derivada de esta:

$$S_v(t) = \frac{d}{dt}S(t), \qquad S_a(t) = \frac{d}{dt}S_v(t)\,, \tag{94}$$

donde $S_v \in \mathbb{R}^n$ es la spline de velocidades y $S_a \in \mathbb{R}$ es la spline de aceleraciones.

4.3.1.1. Derivada de una spline

Debido a que la estructura de la spline es polinómica y además se conoce el orden de los polinomios, es posible calcular la derivada de forma analítica sin tener que recurrir a derivadas numéricas. Ordenando los coeficientes de la spline S a derivar en forma de matriz, colocando en cada fila los coeficientes de cada segmento polinómico y en cada columna los coeficientes correspondientes a una potencia ordenados de mayor a menor, se tiene que para la spline unidimensional

$$S(t) = \begin{cases} c_n^1(t-t_i)^n + \cdots + c_1^1(t-t_i) + a_0^1 & \text{en } t \in [t_0, t_1]\,, \\ c_n^2(t-t_i)^n + \cdots + c_1^2(t-t_i) + a_0^2 & \text{en } t \in [t_1, t_2]\,, \\ \vdots & \vdots \\ c_n^{N_n-1}(t-t_i)^n + \cdots + a_1^{N_n-1}(t-t_i) + a_0^{N_n-1} & \text{en } t \in \left[t_{N_n-1}, t_{N_n}\right], \end{cases} \tag{95}$$

siendo $n \in \mathbb{N}$ el orden de los polinomios utilizados, en este caso 4, y $N_n \in \mathbb{N}$ la cantidad de nodos en la spline; es posible ordenar sus coeficientes en la siguiente matriz:

$$C = \begin{bmatrix} c_n^1 & \cdots & c_1^1 & c_0^1 \\ c_n^2 & \cdots & c_1^2 & c_0^2 \\ \vdots & \ddots & \vdots & \vdots \\ c_n^{N_n-1} & \ldots & c_1^{N_n-1} & c_0^{N_n-1} \end{bmatrix}. \tag{96}$$

A pesar de que en este caso los polinomios interiores de la spline son de orden 3, la spline completa es de orden 4 y, por lo tanto, los polinomios de tercer orden se almacenan como polinomios de cuarto orden con el coeficiente del término de mayor orden nulo. Esto permite ordenar los coeficientes en la matriz C, que de otro modo no sería consistente en el número de columnas, ya que la primera y última fila tendrían 5 columnas y el resto solamente 4. De hecho, es de este modo como se guardan en el objeto de polinomio por partes (piecewise polynomial) de MATLAB. Para calcular los coeficientes de la spline derivada, que resulta en una spline con el mismo número de tramos pero de orden $n-1$, es suficiente con aplicar el operador

$$D(\varphi) = \varphi \begin{bmatrix} 0 & n & 0 & \cdots & 0 \\ 0 & 0 & n-1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix} \tag{97}$$

sobre la matriz de coeficientes, de modo que los coeficientes de la derivada de la spline S pueden ser calculados mediante $D(C)$. Otra opción es utilizar la función de MATLAB `fnder`, pero esto resulta cerca de un orden de magnitud más lento.

4.3.1.2. Máximo de las derivadas

El objetivo no es simplemente hallar las derivadas, sino encontrar el máximo de tal manera que no supere el límite impuesto. Una opción es simplemente tomar el valor absoluto de los valores de las splines en puntos test a lo largo de la misma y seleccionar el máximo de entre ellos. Esto no proporciona realmente el máximo de la función, pero si los puntos test son suficientemente

densos proporcionan una aproximación precisa. La otra opción es aprovecharse de que la función es conocida y encontrar los extremos analíticamente. Para ello es necesario tomar la segunda derivada de S en el caso de la velocidad (la cual ha de ser calculada de todos modos), y la tercera en el caso de la aceleración (que supone una derivada adicional); a continuación, hallar los ceros en esta mediante alguna función de MATLAB, como puede ser `fnzeros`; y, finalmente, tomar los valores absolutos de los valores de las splines de interés en los extremos hallados. El tiempo de ejecución en ambos casos es comparable y la precisión es suficientemente cercana como para no suponer una diferencia, especialmente teniendo en cuenta que las velocidades y aceleraciones nominales suelen tener un margen de error y, por lo tanto, los límites han de establecerse con un cierto margen de seguridad. Por consiguiente, se ha seleccionado la primera opción, que resulta más sencillo de implementar.

4.3.2. Duración máxima

Igual que la duración del trayecto ha actuado como factor de minimización, también ha de considerarse una restricción. En la práctica es importante que las misiones de recogida de muestras se ajusten a un periodo de tiempo prestablecido, ya que de lo contrario la batería o el combustible del USV podría acabarse, dejándolo aislado en medio de la masa de agua, o podría perjudicar otras misiones que pudieran haber sido planificadas al terminar esta. El cálculo de la duración se realiza de igual manera que en los objetivos, pero su uso dentro del algoritmo de optimización es diferente.

4.3.3. Zonas prohibidas

En modelos matemáticos no existe ninguna dificultad al definir un dominio infinito. Sin embargo, en la práctica, lo más normal es encontrar limitaciones; no solo en el dominio temporal, como se ha comentado en la sección anterior, sino también en el dominio espacial. En particular en este problema, el objetivo es encontrar trayectorias que puedan ser recorridas por un USV, lo cual implica que el vehículo ha de mantenerse dentro del agua. Además, pueden existir algunas regiones dentro del embalse o lago en cuestión en las cuales la navegación esté restringida. Por estas razones es importante implementar una restricción en el algoritmo de optimización que evite que las trayectorias recorran zonas no permitidas, las aquí llamadas zonas prohibidas.

Para que el algoritmo de optimización sea capaz de evitar las zonas prohibidas del dominio, una opción es restringir la distancia viajada dentro de tal zona. En algoritmos mono-objetivo, al tener una penalización proporcional a la distancia que se recorre dentro de la zona prohibida, se consigue que el algoritmo sea capaz de buscar una solución que cumpla las restricciones de manera progresiva. Podemos medir la longitud que recorren las trayectorias dentro de la zona prohibida midiendo la distancia entre intersecciones impares y pares consecutivas (en ese orden) de la spline con la superficie de la zona prohibida, asumiendo que ambos extremos de la trayectoria están en zona segura, lo cual se consigue haciendo que esos puntos no sean variables de decisión, sino puntos fijos. Esto tiene sentido ya que sería conveniente que el barco comenzara y terminara en unos puntos determinados.

4.3.3.1. Definición de las zonas prohibidas

Las zonas prohibidas son intrínsecamente volúmenes que cumplen la condición de no pertenencia a un dominio especificado. La manera más sencilla de definir este volumen es definir una función tal que su resultado escalar sea negativo a un lado de la frontera y positivo al otro, ya que esto permite resolver un sistema para las intersecciones entre la zona prohibida y la trayectoria con facilidad. Si las zonas prohibidas están definidas como funciones analíticas, el intervalo de

intersección $[a, b]$, en el cual la trayectoria se encuentra en el interior de la zona prohibida, puede ser hallado creando un sistema y resolviéndolo para encontrar los instantes de intersección.

Para ello supondremos que $S(t)\colon \mathbb{R} \to \mathbb{R}^3$ es la spline que representa la trayectoria, y $F(x, y, z)\colon \mathbb{R}^3 \to \mathbb{R}$ es la función que define la superficie de la zona prohibida,

$$S(t) = \begin{bmatrix} S_x(t) \\ S_y(t) \\ S_z(t) \end{bmatrix}, \tag{98}$$

$$F(x, y, z) \leq 0\,, \tag{99}$$

y se ha de hallar t en el siguiente sistema:

$$\begin{cases} S_x(t) = x\,, \\ S_y(t) = y\,, \\ S_z(t) = z\,, \\ F(x, y, z) \leq 0\,, \end{cases} \tag{100}$$

que simplemente implica

$$F\big(S_x(t), S_y(t), S_z(t)\big) = F(t) \leq 0\,. \tag{101}$$

Entonces, las intersecciones de la trayectoria S con la zona prohibida F pueden encontrarse en lugares donde $F(t)$ cambia de signo, es decir, en los puntos donde (101) deja de cumplirse. Estos puntos definen los intervalos de intersección $[a, b]$ comentados con anterioridad. En la Figura 36 se muestra un ejemplo de las intersecciones de una spline con una zona prohibida, que en este caso se trata del exterior de la esfera.

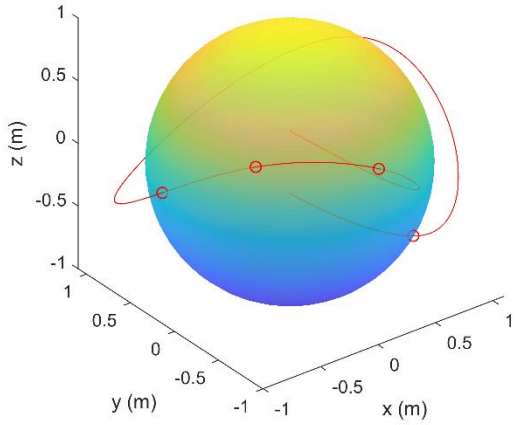


Figura 36: Ejemplo de intersecciones de una spline con una zona prohibida.

4.3.3.2. Intersección de la trayectoria con las zonas prohibidas

La ecuación (101) indica, mediante una desigualdad, cómo determinar si un punto dado está dentro de la zona prohibida o no. Sin embargo, para calcular las longitudes de estos segmentos, es necesario conocer los puntos en los cuales se entra o se sale de la región prohibida. Por ello, el objetivo es hallar todas las soluciones de $F(t) = 0$ en el intervalo $\left[t_0, t_{N_n}\right]$. Dado que la función

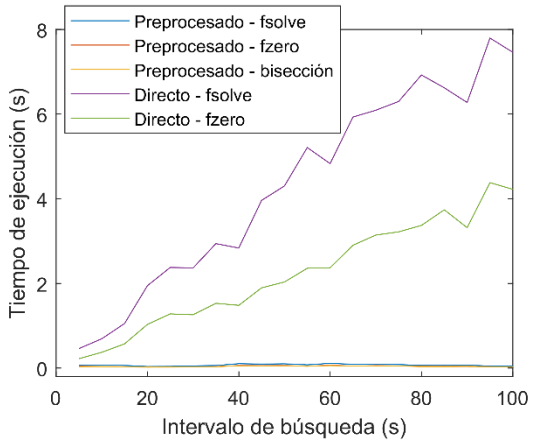
es no lineal y no es conocida, se necesita un método numérico robusto que sea capaz de encontrar las soluciones de manera consistente y, como ha de resolverse un gran número de veces para su optimización, también ha de ser lo más rápido posible.

La función de la zona prohibida no se da por conocida de forma que pueda adaptarse a diferentes masas de agua en la práctica, de modo que encontrar una solución analítica queda descartado. La mayor parte de métodos numéricos, especialmente aquellos que no necesitan demasiado tiempo o memoria, solamente proporcionan una única solución, por lo que es necesario ejecutar los algoritmos desde diversas condiciones iniciales para capturar todas las soluciones. Por ello, uno de los mayores problemas será seleccionar condiciones iniciales adecuadas.

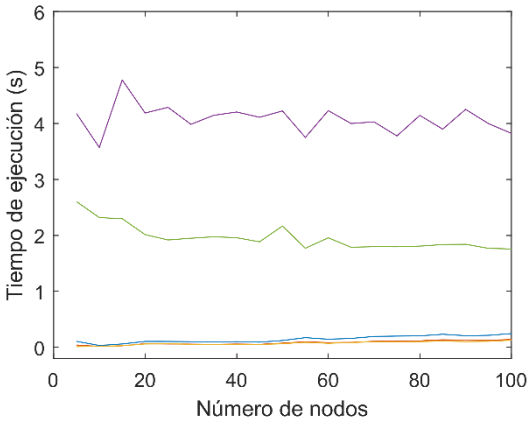
La forma más obvia de hallar todas las soluciones es resolver la ecuación desde un gran número de puntos a lo largo del dominio. Se denominará a esta estrategia la estrategia directa. Generalmente esto es ineficiente, dado que la mayor parte de las soluciones encontradas se encuentran repetidas, y la densidad de condiciones iniciales ha de ser significativa para poder capturar soluciones con zonas de atracción pequeñas.

Otra estrategia es comenzar con todas esas soluciones iniciales, pero en lugar de resolver la ecuación desde todas ellas, determinar dónde es posible encontrar una solución observando los cambios de signo en $F(t_0)$ para condiciones iniciales adyacentes. Entonces, solamente las soluciones de condiciones iniciales donde se cambie de signo son calculadas. Denominaremos a esta estrategia la estrategia preprocesada. A pesar de que esto reduce la cantidad de veces que se ejecuta el algoritmo de resolución y, con ello, la cantidad de tiempo necesaria para la ejecución, esta estrategia puede no encontrar algunas soluciones si el espacio entre condiciones iniciales es demasiado grande para capturar un cambio de signo, por lo que generalmente es una estrategia menos precisa que la directa.

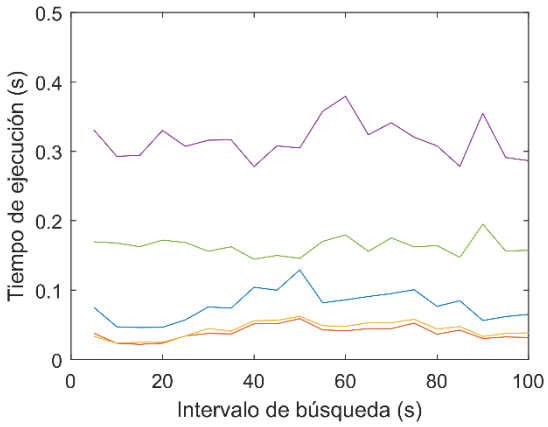
Comparando las dos estrategias, con un intervalo de tiempo fijo entre condiciones iniciales (dt), el tiempo de ejecución de la estrategia directa se incrementa a medida que el intervalo de búsqueda $[0, T]$ se extiende, como se observa en la Figura 37(a), pero se mantiene constante cuando el número de soluciones reales (que en este caso es directamente proporcional al número de nodos, N_n) se incrementa, como puede comprobarse en la Figura 37(b). En estos dos casos la estrategia preprocesada hace lo contrario que la directa. Sin embargo, elegir un paso de tiempo dinámico, de modo que la variación entre condiciones iniciales sea inversamente proporcional al número de soluciones reales, resulta en ambas estrategias manteniendo el tiempo de ejecución estático a medida que se incrementa el intervalo de búsqueda, como se exhibe en la Figura 37(c), e incrementando el tiempo de ejecución al incrementar el número de soluciones, como muestra la Figura 37(d). Los nodos para estas pruebas han sido generados con desviaciones aleatorias sobre una trayectoria que se adhiere a la superficie de una esfera de radio 1 m, que define la zona prohibida, y con los extremos fijos en su interior, al igual que se muestra en la Figura 36.



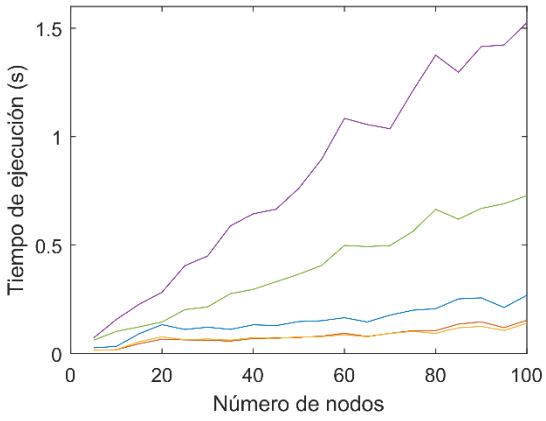
(a) Variando intervalo de búsqueda con paso de tiempo estático ($N = 20, dt = 0.2$).



(b) Variando número de soluciones con paso de tiempo estático ($T = 50, dt = 0.2$).



(c) Variando intervalo de búsqueda con paso de tiempo dinámico ($N = 20$).



(d) Variando número de soluciones con paso de tiempo dinámico ($T = 50$).

Figura 37: Tiempos de ejecución de los algoritmos de búsqueda de intersecciones.

MATLAB ofrece varios algoritmos de resolución, siendo para ecuaciones no lineales `fzero` y `fsolve` los más comunes. Tras algunas pruebas, se puede concluir que en general `fsolve` consigue encontrar más soluciones que `fzero`, pero también tiende a considerar como soluciones algunos mínimos locales que no representan una solución (falsos positivos). Podrían fácilmente filtrarse las soluciones no convergentes (lo cual se indica mediante un parámetro de salida de la función), pero en ese caso podríamos descartar soluciones en la estrategia preprocesada. No encontrar algunas soluciones no es un gran problema, a no ser que el número de soluciones sea muy alto, que en este problema está relacionado con el número de nodos de la spline. Podría evitarse esto asegurándose de que los nodos no se encuentran concentrados en zonas demasiado densas, pero entonces se perdería la flexibilidad de la que sería conveniente disponer en el proceso de optimización. Por ello, se decide hacer el proceso de encontrar soluciones lo más robusto posible.

Dado que para la estrategia preprocesada ya se están encontrando cambios de signo, es posible obtener intervalos en los cuales es seguro encontrar una solución (por el teorema del valor medio) y puede ser resuelta utilizando un método tan sencillo como el método de bisección.

En la Figura 38 se puede observar lo que se comentaba con anterioridad, que `fsolve` proporciona soluciones no válidas en mínimos locales sobre la $F(t)$ que se obtiene utilizando una spline de 20 nodos con variaciones aleatorias sobre la superficie de la esfera de radio 1 m que define la zona prohibida, y con los extremos fijos en puntos interiores, como en la Figura 36. El hecho de converger a mínimos locales es más común en la estrategia directa pero también sucede en la estrategia preprocesada en ocasiones. También se observa que los métodos preprocesados no son capaces de encontrar dos de las soluciones cerca de $t = 2$, dado que el intervalo entre condiciones iniciales no incluye ese cruce tan breve.

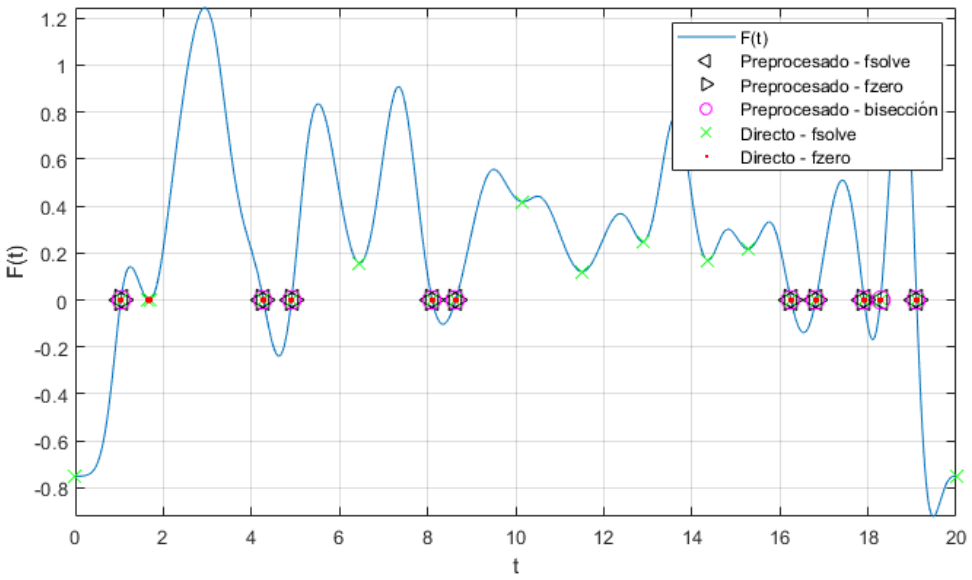


Figura 38: Ejemplo de comparación de precisión de los métodos y estrategias.

La velocidad de cómputo es un factor importante a considerar, y través de diversas pruebas se comprueba que la estrategia preprocesada es mucho más rápida que la directa, lo cual es de esperar, y que `fsolve` es más lento, pero más preciso, que `fzero`. El método de bisección se encuentra en términos de velocidad entre los dos anteriores (aunque no consistentemente, dado que la diferencia no es muy grande), y su precisión parece ser impecable siempre que el intervalo entre condiciones iniciales sea suficientemente pequeño, de modo que la mejor opción parece ser el método de bisección preprocesado.

Otros métodos para resolver la ecuación numéricamente también fueron considerados. En particular, como puede comprobarse en la Tabla 5, el método de Newton fue probado contra el de la bisección, y en un problema típico de este tipo, el método de bisección resulta consistentemente más veloz. No resulta más veloz, sin embargo, que el muestreo selectivo, que es esencialmente un método de la bisección modificado. Consiste en realizar el paso de preprocesado de manera progresiva en intervalos más pequeños hasta obtener todas las soluciones de manera simultánea. A diferencia del método de la bisección, no se toma un intervalo con una única solución, sino que se toma el intervalo completo. El primer paso del muestreo selectivo coincide con el de preprocesado. Se divide el intervalo de búsqueda completo en sub-intervalos más pequeños, y de entre estos se descartan aquellos que no contengan un cambio de signo. En el siguiente paso cada uno de los intervalos restantes se vuelve a subdividir en múltiples partes, a diferencia de la bisección que subdivide en solo dos partes. Los datos estadísticos han sido obtenidos de la resolución de las intersecciones de 100 splines, cuyos nodos se generan de igual manera que en el caso anterior, y la zona prohibida definida con anterioridad.

Tabla 5: Comparación de los métodos de resolución.

	Tiempo medio de ejecución	Varianza del tiempo de ejecución
Newton	6.0076×10^{-1} s	1.6932×10^{-2} s^2
Bisección	2.9993×10^{-2} s	4.1908×10^{-5} s^2
Muestreo selectivo	1.0520×10^{-3} s	2.2539×10^{-8} s^2

Debido a que el método del muestreo selectivo es igual de preciso que el de la bisección o el de Newton, pero resulta más veloz, este es el utilizado en el algoritmo de optimización.

4.3.3.3. Longitud de las intersecciones

Una vez se han hallado las intersecciones y, por lo tanto, los intervalos de la spline contenidos en la zona prohibida, es necesario hallar su longitud para utilizarla en el algoritmo de optimización. Para ello, la distancia entre dos intersecciones consecutivas a y b con la superficie puede ser medida mediante la integral de la distancia (85), la misma utilizada para hallar la longitud total de la spline pero utilizando las intersecciones como límites de la integral.

Existen otras opciones más sencillas que permiten prescindir de la integración. En lugar de minimizar la longitud de los segmentos de la trayectoria que se encuentran en zona prohibida, es posible minimizar el tiempo que se emplea ahí. Esto tiene un efecto similar en la penalización pero es más rápido de calcular, simplemente $\tau_{i+1} - \tau_i$. Se corre el riesgo, sin embargo, de que en lugar de reducir la distancia en zonas prohibidas se incremente la velocidad, lo cual no siempre es apropiado.

4.3.3.4. Zonas prohibidas en geometrías irregulares

Todo lo que se ha comentado se ha desarrollado asumiendo que la función que define las zonas prohibidas, F, es suficientemente suave para aplicar los métodos numéricos de resolución. Sin embargo, en lugar de utilizar funciones suaves, resulta más preciso representar el dominio directamente con una geometría mallada irregular, posiblemente la misma utilizada para la simulación. En ese caso, al haber elegido usar el método de la bisección, lo único que necesita cumplirse es que en un lado de la superficie $F(t)$ sea negativo y en el otro sea positivo, no es necesario que la función sea ni siquiera continua.

De hecho, dado que es un método numérico que evalúa puntos individuales, es posible simplemente utilizar una función que determine si un punto particular se encuentra dentro o fuera del dominio y devuelva el resultado en forma de un valor negativo, que puede ser constante, si está fuera, y un valor positivo, si está dentro. Para ello pueden utilizarse funciones como `inpolyhedron` [16], que devuelve un 0 si está fuera o un 1 si está dentro de la geometría tridimensional dada, y restar al resultado 0.5 para dejar los marcadores a ambos lados del cero con valores en $\{-0.5, 0.5\}$, lo cual es suficiente para que el método de la bisección encuentre las intersecciones.

Sin embargo, es conveniente tener en cuenta ciertas particularidades. Para empezar, la sonda del USV no es capaz de extenderse a la misma profundidad a la que llega el lago, de modo que comprobar si un punto está dentro del agua no implica necesariamente que pueda ser alcanzado. Por otra parte, aproximarse demasiado a la costa puede suponer un problema para el USV, ya que corre el riesgo de quedar encallado, que se dañe la sonda o que el mapeado en la costa no sea completamente fiable debido a la variación natural del nivel del agua. Por otra parte, la utilización de funciones de pertenencia a espacios tridimensionales irregulares tiende a ser lenta, ya que métodos como el de los índices de una curva plana (winding number) [48] o el de la suma de

ángulos [49] no funcionan en 3D, y otros, como el de ray casting [50] [51], resultan mucho menos eficientes que en 2D. En la Figura 39 se muestra cómo se determina si un punto se encuentra dentro de la geometría utilizando ray casting. Si como en el punto A el número de intersecciones de un rayo con las fronteras es par, el punto se encuentra en el exterior, si es impar, como en el punto B, se encuentra en el interior.

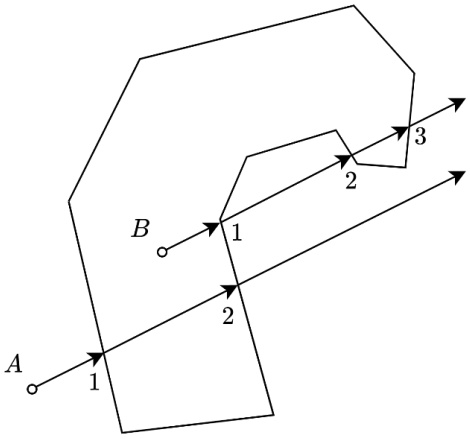


Figura 39: Ray casting para determinar la localización de un punto.

Por estas razones se ha optado por comprobar si las coordenadas horizontales de los puntos candidatos pertenecen al interior de un polígono formado escalando hacia abajo los límites de la superficie del agua del dominio tridimensional irregular, marcado con la línea roja en la Figura 40, y comprobar si la coordenada vertical z se encuentra entre los límites especificados por la sonda. De este modo se resuelven ambos problemas presentados por el método anterior. Por un lado, los límites de la sonda son implementados directamente en el dominio espacial y, por otro, se consigue una ejecución mucho más veloz. Además, la pertenencia a un polígono puede comprobarse mediante la función oficial de MATLAB `inpolygon`, sin tener que recurrir a soluciones de terceras partes. Evidentemente esta comprobación devuelve un 0 o un 1, a lo cual hay que restarle 0.5 como en el caso anterior.

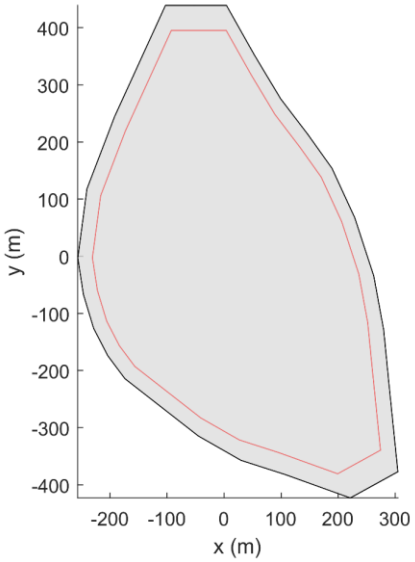


Figura 40: Dominio poligonal reducido en el plano horizontal.

Para implementar zonas prohibidas por partes o con subdominios interiores, la utilización de funciones binarias resulta muy conveniente, ya que pueden aplicarse con facilidad toda clase de funciones lógicas de manera que se cumplan los requerimientos necesarios. En este proyecto se ha tomado siempre la zona prohibida definida por la geometría, como se acaba de describir, y además se han añadido, en algunos escenarios, hasta tres zonas prohibidas adicionales definidas mediante elipses en el plano horizontal, aunque el número concreto no está limitado de manera alguna. A pesar de que resulta más sencillo definirlas de forma continua, estas zonas prohibidas adicionales han sido formuladas de manera booleana porque de esta manera es suficiente con aplicar el operador "and" recursivamente para definir la función de zona prohibida final.

4.4. Función objetivo

El problema de optimización que se tiene es intrínsecamente multi-objetivo. Existen dos maneras de resolver tales problemas. Por un lado, se pueden optimizar cada uno de los objetivos de manera individual y simplemente no considerar soluciones que no cumplan las restricciones (estrategia multi-objetivo) y, por otro lado, se pueden combinar todos los objetivos y las restricciones en una sola función objetivo mediante su suma (estrategia mono-objetivo). Debido a la complejidad del TFM se ha optado por dar un tratamiento mono-objetivo al problema de optimización y dejar el planteamiento multi-objetivo como parte del trabajo futuro.

Analizando los objetivos propuestos en secciones anteriores, se observa que los dos primeros, duración y distancia total de la trayectoria, son objetivos de minimización, mientras que el tercero, la concentración de partículas, se trata de un objetivo de maximización. Para consolidar ambos tipos de problemas en uno solo es necesario invertir el signo de uno de los grupos. Lo más común es formular problemas de optimización en forma de problemas de minimización, de modo que la contribución de la concentración de partículas se multiplica por -1.

Las restricciones se imponen en forma de fuertes penalizaciones sobre la función objetivo, lo cual consigue que sea un problema de optimización sin restricciones, que hace la implementación de un algoritmo más sencilla. Todas las restricciones de este problema guardan una relación directa con el objetivo, de modo que no es necesario invertir el signo de ninguna de ellas al aplicarlas a la función objetivo.

Con cada una de las partes calculadas el usuario ha de decidir cuáles son las prioridades del problema, es decir, cuál de los objetivos es más importante, si es que hay alguno más importante que otros. Esta importancia se ve reflejada en la función objetivo en forma de coeficientes que acompañan a cada término. La función objetivo de este problema es

$$\min Z = c_t\left(t_{N_n} - t_0\right) + c_l l - c_\rho \sum_{i=1}^{N_p} \varrho_p^i + c_P Z_P \,, \tag{102}$$

donde

$$\begin{aligned} Z_P = b_v\left(v_{\max} - \sup_{i\in\{1,\dots,N_s\}}\{v_i\}\right) + b_a\left(a_{\max} - \sup_{i\in\{1,\dots,N_s\}}\{a_i\}\right) \\ + b_t\left(t_{\max} - t_{N_n} + t_0\right) + \sum_{i=1}^{N_{\text{inter}}} l_{\text{inter}}^i \end{aligned} \tag{103}$$

es la función que combina las penalizaciones; c_t, c_l, c_ρ y $c_P \in \mathbb{R}^+$ son los coeficientes que regulan la importancia de cada parte de la función objetivo (102), cuyos valores y variables relacionadas se muestran en la Tabla 6; $[t_0, t_{N_n}]$ es el intervalo temporal en el cual se desarrolla la misión de

medición de la concentración de cianobacterias; $l \in \mathbb{R}^+$ es la longitud de la trayectoria calculada mediante (85); $\boldsymbol{\varrho}_p = \left(\varrho_p^1, \dots, \varrho_p^{N_p}\right)^T \in \{0,1\}^{N_p}$ es el vector de marcadores que indica si una partícula ha sido vista; $\boldsymbol{v} = \left(v_1, \dots, v_{N_s}\right)^T$ y $\boldsymbol{a} = \left(a_1, \dots, a_{N_s}\right)^T \in \mathbb{R}^+$ son los vectores de las normas de la velocidad y la aceleración en los puntos de muestreo de la spline correspondiente; $v_{\max}$ y $a_{\max} \in \mathbb{R}^+$ son las normas de la velocidad y la aceleración máximas permitidas, respectivamente; b_v, b_a y $b_t \in \{0,1\}$ son los marcadores booleanos que se activan si la velocidad, la aceleración o el tiempo, en ese orden, se encuentran por encima del límite permitido; y $\boldsymbol{l}_{\text{inter}} = \left(l_{\text{inter}}^1, \dots, l_{\text{inter}}^{N_{\text{inter}}}\right) \in (\mathbb{R}^+)^{N_{\text{inter}}}$ es el vector de las longitudes de las intersecciones, que se obtienen mediante (85) tras hallar las intersecciones de la spline con la zona prohibida como se describe en la Sección 4.3.3.

Tabla 6: Coeficientes de la función objetivo.

Duración	c_t	1
Longitud	c_l	1
Concentración de partículas	c_ρ	10^3
Penalización	c_P	10^{20}

Es de notar que de entre los objetivos, el de la concentración de partículas tiene el coeficiente más elevado. Además, este coincide con el único objetivo que ha sido multiplicado por -1 para maximizarlo, lo cual quiere decir que es común que la función objetivo adquiera valores tanto positivos como negativos.

Asumiendo que es posible encontrar una solución válida (ya que las trayectorias definidas por splines cambian suavemente y, por lo tanto, se trata de un espacio de búsqueda que se comporta bien), es posible concentrar los esfuerzos en proporcionar una formulación que incremente la flexibilidad. Esto significa que no solo las coordenadas espaciales de los nodos de la spline han de ser considerados como variables de decisión, sino también su coordenada temporal. Esto hace posible mantener los nodos en su emplazamiento espacial, pero cambiar la curvatura de la spline entre ellos. Dado que existen N_n nodos en la spline en cada una de las coordenadas (x, y, z y t), y los extremos espaciales y un extremo temporal son estacionarios, habrá $4N_n - 7$ variables de decisión, que definen la spline 4-3-4 correspondiente, y es a partir de esta spline y de los resultados de la simulación de las cianobacterias como se calculan todos los valores necesarios para obtener el valor de la función mono-objetivo (102).

5. Optimización Genética

En el capítulo anterior se ha presentado el problema de optimización a resolver, pero existen numerosos algoritmos capaces de realizar esta optimización; que incluyen, por un lado, métodos directos, como pueden ser los métodos de Newton-Raphson, Quasi-Newton, gradiente conjugado, punto interior, o descenso del gradiente; y, por otro lado, métodos heurísticos, como los métodos de la evolución diferencial, nubes de partículas, búsqueda tabú, temple simulado o algoritmos genéticos.

En este problema la función objetivo es irregular y no diferenciable, lo cual hace que se descarten la mayor parte de los métodos directos, ya que su funcionamiento suele depender del gradiente o del Hessiano, que en este caso son desconocidos y difíciles de aproximar. De entre los algoritmos heurísticos restantes se ha seleccionado por su versatilidad el algoritmo de optimización genética, cuyo principio de optimización es el de simular un proceso de selección natural, donde las variables de decisión quedan representadas como genes pertenecientes a cada solución candidata. Además, en los algoritmos genéticos se trabaja con una población de candidatos cuyos genes se combinan o se mutan hasta obtener individuos con funciones objetivo de bajo valor, y devolviendo como solución la mejor encontrada hasta el momento. Este es un algoritmo que suele ser bastante eficaz en problemas con un gran número de variables y un espacio de búsqueda de gran tamaño. A continuación, se presenta la implementación del algoritmo, incluyendo las particularidades seleccionadas en cada parte y los resultados obtenidos mediante su aplicación.

5.1. Implementación del algoritmo

Un algoritmo de optimización genética se divide en dos partes principales. En primer lugar, se tiene la inicialización, encargada de crear todas las variables necesarias y de establecer los valores iniciales; y, en segundo lugar, el bucle de optimización, donde cada iteración representa una generación del algoritmo. La implementación se ha realizado en MATLAB, y el código se incluye en el Apéndice 2.

5.1.1. Inicialización

Esta es la parte inicial del código. En ella se establecen los parámetros relacionados con el método, se crean las variables necesarias y se crea la población inicial. Es importante notar que es aquí donde se importa la geometría para crear los límites del espacio de búsqueda, el cual se define inmediatamente después en forma de una función, y que también se importa la solución de la simulación del transporte de las cianobacterias, que a su vez resulta necesaria para calcular los valores objetivo.

5.1.1.1. Función de generación

Cada uno de los miembros de la población se ha creado como una estructura que almacena no solo las variables de decisión, sino también el valor de la función objetivo, la velocidad y aceleración máxima y otras variables de interés. De este modo se logra que no sea necesario recalcular todos esos valores si el candidato no ha sido modificado, como ocurre con las élites. Por lo tanto, la función de generación está formada por dos partes: la generación de los valores de las variables de decisión y el cálculo de las variables asociadas.

En primer lugar, se generan los valores de las variables de decisión del candidato, lo cual se hace aleatoriamente de acuerdo a una distribución uniforme de un rango especificado en los

parámetros. En este caso el rango se especifica individualmente para cada coordenada y se toma de los extremos del dominio válido de búsqueda, es decir, las variables de decisión se encuentran entre los extremos del dominio especificado en la Sección 4.3.3.4 en cada una de las direcciones espaciales y entre $t = 0$ y la duración máxima especificada, $t = t_{\max}$. Recordando lo comentado en la Sección 4.2.1, para garantizar el buen condicionamiento del sistema lineal que calcula la spline, es necesario que el vector temporal esté ordenado y que $t_1 = 0$. Al seleccionar valores aleatorios, no se garantiza que el vector temporal comience en $t_1 = 0$, ni que los extremos de la spline se hallen donde se los necesita, de modo que se ordena el vector temporal y se desplaza el vector completo de acuerdo a

$$t_i = t_i - t_1, \qquad \forall i \in \{1, \dots, N_n\}, \tag{104}$$

consiguiendo de esta manera que se mantengan los intervalos temporales entre los puntos generados, pero que se comience la misión en el momento necesario; y se establecen los extremos de la spline en los lugares especificados en los parámetros. En la Figura 41 se muestra cómo se desplaza el vector temporal, de manera que se mantengan los intervalos temporales entre los nodos ($t_1, \dots, t_4$ en este ejemplo) pero se comience la misión al principio del intervalo permitido, $[t_0, T]$.

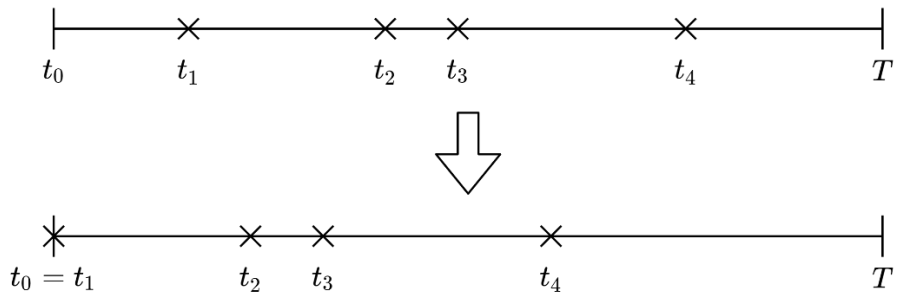


Figura 41: Ejemplo de desplazamiento del vector temporal.

En segundo lugar, han de calcularse las variables asociadas a cada individuo de la población. Estas incluyen: los coeficientes de la spline definida por los nodos, la spline de velocidades, la spline de aceleraciones, la longitud total de la trayectoria, la duración total de la trayectoria, los vectores de puntos test, las intersecciones con la zona prohibida, la longitud de cada una de las intersecciones, la velocidad máxima, la aceleración máxima y el vector de partículas vistas. Con el objetivo de ahorrar tiempo, este último solamente se calcula si se cumplen las restricciones.

5.1.2. Bucle de optimización

Una vez se han definido todas las variables necesarias y existe una población inicial, puede comenzarse el bucle, que se repite hasta que se cumplan las condiciones de convergencia o se alcancen las iteraciones máximas especificadas. El objetivo del bucle es mejorar la población de manera iterativa hasta lograr una solución que sea lo suficientemente buena, aunque también es importante lograr que se mantenga la diversidad genética para que la solución no quede estancada antes de tiempo. Para lograrlo se hace uso de varias técnicas que se describen a continuación: el elitismo, el cruce, la mutación y la inmigración. También se describen otras funciones de importancia: las funciones de selección, recombinación, refinamiento y parada.

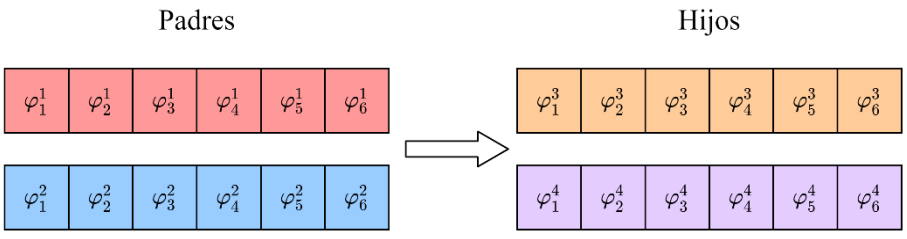
5.1.2.1. Función de selección

Tanto el operador de cruce como el operador de mutación requieren la utilización de uno o dos padres para crear uno o varios hijos, que a su vez son añadidos a la generación siguiente. Por ello, resulta necesario seleccionar esos padres. De nuevo se siguen los principios de la selección natural y se seleccionan de entre todos los candidatos de la generación anterior aquellos con mejores valores de la función objetivo. Sin embargo, elegir simplemente a los mejores trabaja en contra de la diversidad genética, ya que nuevas soluciones introducidas a través de los inmigrantes tienen una baja probabilidad de ser mejores que aquellas que ya han sido sometidas a varias iteraciones de optimización. Por ello, se utiliza una función de selección que preserve las posibilidades para todos los individuos, aunque ponga un mayor énfasis en los mejores. Se trata de una función de selección dinámica y preservativa, ya que la probabilidad de selección varía en función del valor objetivo.

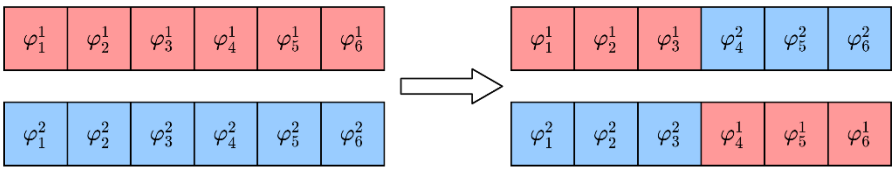
En particular se ha utilizado un método proporcional (también conocido como método de la ruleta) en el cual las proporciones se han hallado mapeando los valores objetivos de todos los candidatos al rango $[0.01, 0.5]$, de manera que el peor candidato de la población tenga, como máximo, un 1% de posibilidades de ser elegido y el mejor, como máximo, un 50%. Hay que tener en cuenta que no se trata de probabilidades conservativas, es decir, puede haber más de dos candidatos con probabilidades del 50%, haciendo que las probabilidades reales varíen en función de la composición de la población. La selección se realiza utilizando la función de MATLAB `randsample` y utilizando los valores mapeados como pesos en la selección.

5.1.2.2. Operador de cruce

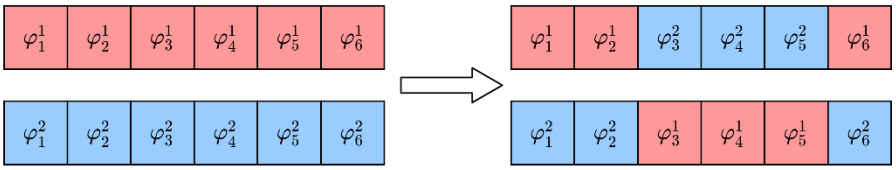
Uno de los factores que juegan un papel importante en la selección natural es la combinación de genes mediante el cruce. Este proceso consiste en combinar las variables de decisión de dos candidatos de la generación anterior, los padres, para obtener uno o más candidatos nuevos, los hijos, que compartan en cierta medida los genes de cada uno. Existen muchas maneras de realizar el cruce, que incluyen cruces aritméticos, cruces de un punto, cruces de varios puntos y cruces uniformes. La diferencia entre cada uno de ellos puede observarse en la Figura 42, donde φ_i representa cada uno de los genes. En este proyecto se ha decidido utilizar el cruce uniforme.



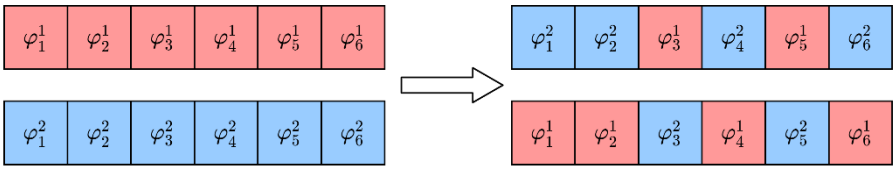
(a) Cruce aritmético.



(b) Cruce de un punto.



(c) Cruce de dos puntos.



(d) Cruce uniforme.

Figura 42: Tipos de cruces.

Una de las razones por las cuales no se ha utilizado el cruce aritmético es que cada nodo guarda un significado en sí mismo. El cruce aritmético combina los genes de acuerdo a

$$\varphi_i = s\varphi_i^1 + (1-s)\varphi_i^2, \qquad \forall i \in \{1, \dots, N_n\} \tag{105}$$

donde φ_i, φ_i^1 y $\varphi_i^2 \in \mathbb{R}$ es cada uno de los genes del hijo y cada padre, respectivamente; y $s \in [0,1]$ es un número aleatorio tomado de una distribución uniforme en el rango mencionado. De esta manera, se obtiene una solución intermedia entre los padres, lo cual no sería adecuado, ya que sería común combinar dos nodos, ambos situados en zonas de alta concentración de partículas, para obtener un nodo situado entre medias en una zona despoblada.

Para evitar esto, los cruces por uno o varios puntos y el cruce uniforme pertenecen todos ellos a la misma familia de cruces, en la cual se selecciona el gen de uno de los padres o del otro. De esta manera, al combinar los genomas de los padres, se conserva la posición de los nodos y, por lo tanto, la concentración de partículas en los mismos. El cruce por un punto consiste en seleccionar un valor aleatorio de una distribución uniforme en el rango $[0,1]$. Este número determina el punto de corte (en porcentaje del total de nodos N_s). Todos los genes anteriores al punto de corte provienen de uno de los padres, y los posteriores provienen del otro.

El cruce por dos o más puntos es similar al cruce por un punto, con la diferencia de que se generan tantos números aleatorios como cruces y los padres se alternan en cada intervalo resultante hasta alcanzar el último. De este modo, en un cruce por dos puntos se producen tres intervalos, donde los de los extremos provienen de un padre y el central del otro.

El cruce uniforme es similar, pero en lugar de tener un número de cortes fijo, la cantidad de intervalos suele ser más elevada, ya que se determina de manera individual el padre de cada uno de los genes. Para realizar el cruce uniforme se crea un vector de números aleatorios generados a partir de una distribución uniforme del rango $[0,1]$ de la misma longitud que el vector de nodos, N. Si el número aleatorio correspondiente es menor que 0.5 el gen se toma de un padre, si no, del otro. Se generan dos hijos de cada dos padres, de manera que un hijo sea complementario al otro. Los genes que en el primer hijo pertenecen al primer padre, pertenecen al segundo padre en el segundo hijo. Esto incrementa las posibilidades de aislar nodos problemáticos. Hay que tener en cuenta que para mantener la integridad de los nodos se considera que un gen está compuesto por las cuatro coordenadas de un nodo, de manera que si la coordenada x pertenece al primer padre, también lo hacen las coordenadas y, z y t.

En estos tres últimos métodos, al tomar nodos completos de distintos individuos, es posible que el vector temporal no esté ordenado. En ese caso se ordena sin modificar la colocación de los nodos en el resto de coordenadas, lo cual cambia el orden en el que se visitan, que es importante a la hora de realizar varias pasadas por la misma zona. También es posible que al combinar dos padres el hijo obtenga dos veces el mismo instante temporal, al haberlo heredado de antecesores más antiguos a través de las dos vías, pero en posiciones diferentes. Esto supone un problema grave, ya que implica que el sistema que ha de resolverse para hallar la spline es singular y no puede ser resuelto. En ese caso se realiza la media aritmética entre el valor repetido y el valor del nodo anterior, y se asigna como variable temporal a uno de los nodos repetidos, resolviendo así el problema. Una vez se han obtenido las variables de decisión es necesario calcular las variables auxiliares. Este grupo de candidatos se denomina cruces.

Los cruces por uno o varios puntos y el cruce uniforme pertenecen todos ellos a la misma familia de cruces, en la cual se selecciona el gen de uno de los padres o del otro. Cuanto mayor sea el intervalo de agrupamiento, mayor es la probabilidad de conservar la estructura de las soluciones, evitando así la ruptura de esquemas; pero menor es la probabilidad de mejorar el posicionamiento de uno o varios nodos de la spline, que puedan encontrarse desalineados con respecto a la verdadera solución óptima que se busca. Por ello, se opta por realizar un cruce de dos puntos en el cual los puntos de cruce son aleatorios, con el objetivo de que este tipo de cruce capture hasta cierto punto los beneficios de ambos extremos.

5.1.2.3. Operador de mutación

El operador de mutación es similar al de cruce. El objetivo principal de este operador es el de introducir variabilidad genética y realizar pequeños cambios en los valores de los candidatos. Consiste en tomar el genoma de un padre y realizar modificaciones a genes individualmente. En este caso las diferentes coordenadas sí que se ven disociadas de manera que las mutaciones puedan realizarse en un solo eje, independientemente del resto. Para evitar soluciones sin sentido se limita la distancia de mutación a un radio, especificado por los parámetros en cada eje. Tanto la probabilidad de mutación como la distancia a la que se desplaza el gen están determinados por números aleatorios tomados de distribuciones uniformes, con rango $[0,1]$ y $[-1,1]$ respectivamente, representando el segundo número aleatorio el porcentaje de desplazamiento dentro del rango permitido por la distancia máxima de mutación. Matemáticamente, siendo $\boldsymbol{s} = (s_1, \dots, s_{N_n}) \in [0,1]^{N_n}$ y $\boldsymbol{q} = (q_1, \dots, q_{N_n}) \in [-1,1]^{N_n}$ vectores de números aleatorios tomados de una distribución uniforme en sus respectivos rangos, la mutación se calcula de acuerdo a

$$\varphi_i = \varphi_i + r_m q_i, \qquad \forall i \in \{1, \dots, N_n \mid s_i < s_{\text{lim}}\} \tag{106}$$

donde φ_i es el gen a mutar, r_m es el radio de mutación establecido, s_{lim} es la probabilidad de mutación y N_n es el número de nodos de la spline, es decir, el número de genes de un individuo. Recordando lo comentado en la Sección 4.2.1, es necesario que el vector temporal comience en $t = 0$ y que esté ordenado. Sin embargo, al realizar mutaciones, es posible que esto no se cumpla, o que los extremos de la spline no sean los establecidos espacialmente, de modo que es necesario arreglar estos defectos de la misma manera que se ha hecho en la función de generación. Una vez se han obtenido las variables de decisión es necesario calcular las variables auxiliares. Este grupo de candidatos se denomina mutantes.

5.1.2.4. Inmigración

Al orientar los cruces y las mutaciones a individuos con mejores valores de la función objetivo se corre el riesgo de que la diversidad genética quede reducida. Para contrarrestar los efectos de los anteriores grupos se introducen soluciones nuevas, creadas de igual manera que la población inicial. Al tener genomas aleatorios y ser incluidos en el operador de cruce de la generación siguiente son capaces de transferir genes nuevos a una población que, de otro modo, sería demasiado igual entre sí. Este grupo de candidatos se denomina inmigrantes y su número exacto, como el del resto de grupos, se define en los parámetros de la optimización.

5.1.2.5. Función de recombinación

Siguiendo el lema de la supervivencia del más apto, resulta necesario preservar en la población a las mejores soluciones, de manera que estas no se vean degradadas a lo largo de las generaciones debido a mutaciones o cruces. Por ello se seleccionan las mejores soluciones (aquellas con menor valor de la función objetivo) y se añaden a la generación siguiente sin realizar modificaciones de ninguna clase. Al tener las variables auxiliares, comentadas en la inicialización, guardadas en cada individuo no resulta necesario calcularlas de nuevo para este grupo. Este grupo de candidatos se denomina élites.

Una vez se han obtenido todos los miembros de los diferentes grupos se procede a la recombinación. Al tener una formulación mono-objetivo, la opción más sencilla resulta la sustitución generacional aplicando el elitismo, de modo que esta es la recombinación utilizada.

5.1.2.6. Función de parada

Una vez se ha alcanzado el número máximo de nodos en las soluciones se comprueba la convergencia. Esencialmente, la condición de convergencia consiste en que el óptimo no ha de mejorar un porcentaje establecido durante un periodo de generaciones dado. Estos parámetros se establecen en la configuración del método. Por ejemplo, puede establecerse que si el óptimo no mejora un 1% durante 40 iteraciones, se considera que la solución es suficientemente buena y se finaliza la ejecución.

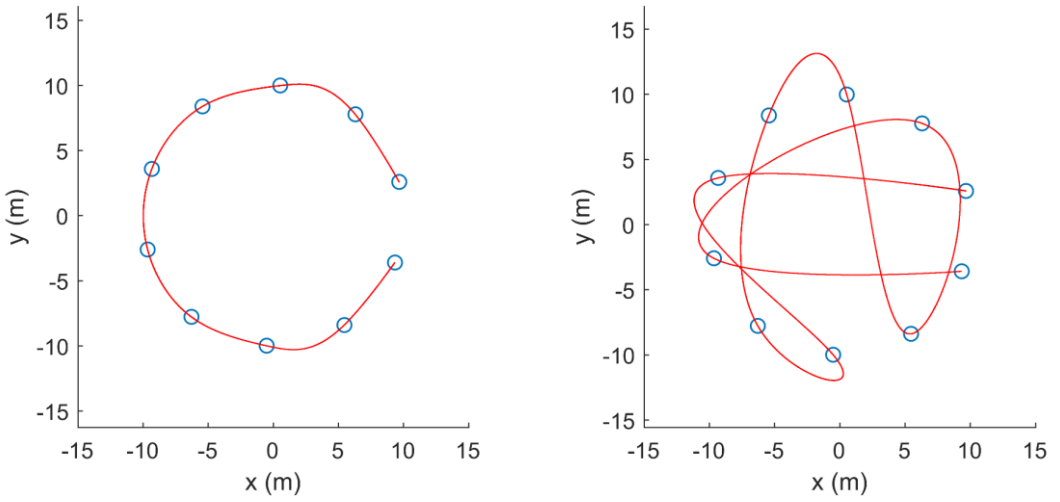
La otra posibilidad para alcanzar el fin de la ejecución del algoritmo es que se alcance el número máximo de iteraciones permitidas, en cuyo caso no se consideraría haber alcanzado la convergencia, sino que la ejecución estaba siendo demasiado larga. Esto no es algo que suela ocurrir, al menos con los parámetros usados en este proyecto.

5.1.3. Mejoras al algoritmo

Existen algunos métodos que diferencian al algoritmo genético implementado del algoritmo genético típico. Por un lado, parte de su código ha sido paralelizado a varios núcleos de computación con el objetivo de agilizar su ejecución, lo cual es una técnica habitual. Por otro lado, y por motivos que se explican a continuación, se introduce la técnica del refinamiento, cuya función es la de incrementar de manera gradual el número de nodos de las splines candidatas y, así, obtener trayectorias más óptimas con menor esfuerzo computacional.

5.1.3.1. Función de refinamiento

El hecho de haber codificado la trayectoria en forma de spline implica que el orden en que se visitan los nodos, o incluso los momentos en los que se visitan, tiene un efecto drástico en la trayectoria. Esto puede comprobarse en la Figura 43, donde los nodos, marcados con los círculos azules, se han mantenido iguales pero se ha cambiado el orden en el que se han visitado, manteniendo, sin embargo, el vector temporal.



(a) Trayectoria a través de nodos ordenados. (b) Trayectoria a través de nodos desordenados.

Figura 43: Efecto del orden de los nodos en la spline.

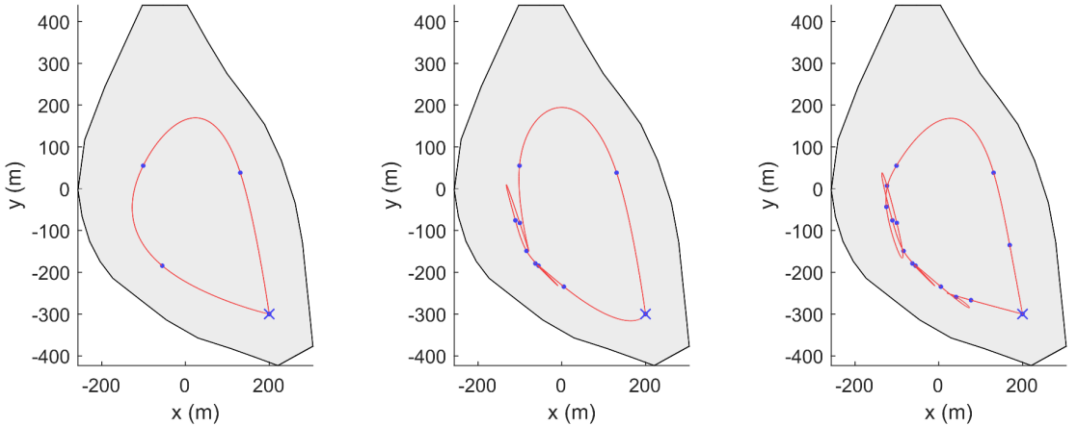
Por lo general la distribución de las cianobacterias, resultante de la simulación, tiende a ser regular debido a las variaciones suaves del flujo a lo largo del dominio. Por esta razón resulta conveniente influir positivamente en el algoritmo de modo que se alcancen trayectorias ordenadas con mayor facilidad. Para ello se opta por incluir los nodos de manera progresiva, comenzando tan solo con cuatro, que se encuentran intrínsecamente ordenados ya que los extremos son fijos, e ir añadiendo nuevos nodos de uno en uno hasta llegar a la cantidad deseada, la cual se especifica en los parámetros.

Para dar una oportunidad de encontrar la ubicación óptima de los nodos antes de añadir más, se establece una condición de convergencia, más suave que la de parada, de manera que cuando se alcance se añada un nodo adicional.

Esta condición de refinamiento es igual a la condición de convergencia, con la única diferencia de que los parámetros son más permisivos, obteniendo así un refinamiento más rápidamente que una convergencia. El porcentaje a mejorar para evitar la convergencia es mayor y la cantidad de generaciones en las cuales se mide el cambio es menor que en la condición de parada.

La cantidad de nuevos nodos a colocar no está restringida de manera alguna, pero para que los cambios resulten más graduales y que resulte más sencillo para el algoritmo de optimización obtener soluciones, se añaden de uno en uno. El nodo adicional se coloca a lo largo de la trayectoria existente en un lugar aleatorio, siguiendo una distribución uniforme entre los límites de la trayectoria. Para ofrecer algo más de variabilidad, en esa generación se incrementa el número de individuos en la población, de manera que cada individuo genere varios refinamientos, cuya cantidad exacta se especifica en los parámetros de la inicialización.

En la Figura 44 se muestra un ejemplo de progresión de refinamiento. A medida que se añaden nodos, la forma general de la trayectoria se mantiene, pero se obtiene más detalle en las zonas necesarias.



(a) 5 nodos. (b) 10 nodos. (c) 15 nodos.

Figura 44: Ejemplo de progresión de refinamiento.

5.1.3.2. Paralelización

Con el propósito de reducir el tiempo de cómputo se paralelizan algunas secciones del algoritmo. En particular, se utiliza el comando de MATLAB `parfor` en los bucles encargados de calcular los valores de la función objetivo y en aquellos encargados de calcular las variables de las cuales la función de optimización depende, como puede ser el cálculo de las splines de trayectoria, velocidad y aceleración, las intersecciones o la concentración de partículas. Se utiliza un perfil de paralelización local que distribuye las iteraciones entre los cuatro núcleos del ordenador (los cuatro trabajadores correspondientes). La creación de los trabajadores puede llevar un tiempo un poco más largo, pero una vez creados pueden ser reutilizados sin coste computacional adicional hasta que sean eliminados, ya sea por inactividad o deliberadamente. No se ha aplicado la paralelización a otros bucles, como los de cruce o mutación, debido a que es necesario compartir ciertas estructuras de datos entre los trabajadores, lo cual provoca que el tiempo de computación sea mayor que si se decide evitar la paralelización.

5.1.4. Pseudocódigo

A continuación, se presenta el pseudocódigo del algoritmo completo de modo muy superficial. El código completo se incluye en el Apéndice 2.

```
 1. Inicializar parámetros y variables.
 2. Generar la población inicial y calcular las variables asociadas.
 4. Calcular los valores de la función objetivo.
 5. Si alguno de ellos es menor que el óptimo, actualizar el óptimo.
 6. Si se cumplen las condiciones de convergencia, finalizar.
 7. Si se cumplen las condiciones de refinamiento, refinar.
 8. Seleccionar padres.
 9. Calcular cruces y sus variables asociadas.
10. Calcular mutantes y sus variables asociadas.
11. Crear inmigrantes y sus variables asociadas.
12. Salvar élites.
13. Combinar las poblaciones de 9-12 en la nueva generación. Volver a 4.
```

5.2. Resultados de la optimización

De igual manera que en los resultados de la simulación numérica, en los resultados de la optimización también se presentan diferentes escenarios (EO1-EO6). Utilizando diversos escenarios de la simulación como base (Sección 3.2), en cada validación se varía el instante inicial, la posición de los nodos de los extremos de la spline, y la cantidad, tamaño y forma de las zonas prohibidas adicionales, manteniendo siempre igual el dominio de búsqueda fundamental. Los parámetros del algoritmo genético utilizados se muestran en la Tabla 7. Además, la duración máxima de la misión es $t_{\max} = 3$ h, la velocidad máxima es $v_{\max} = 1\ \mathrm{m/s}$ y la aceleración máxima $a_{\max} = 1\ \mathrm{m/s^2}$.

Tabla 7: Parámetros de la optimización.

Tamaño de la población	`nPop`	50	
Número de nodos iniciales	`nNodesMin`	4	
Número de nodos máximos	`nNodesMax`	15	
Porcentaje de élites	`pElite`	10	%
Porcentaje de cruces	`pXOver`	30	%
Porcentaje de mutantes	`pMutate`	50	%
Porcentaje de inmigrantes	`pInmigrants`	10	%
Probabilidad de mutación	`mutationProb`	5	%
Radio de mutación (x)	`mutDist(1)`	20	m
Radio de mutación (y)	`mutDist(2)`	20	m
Radio de mutación (z)	`mutDist(3)`	2	m
Radio de mutación (t)	`mutDist(4)`	600	s
Iteraciones máximas	`maxIter`	1500	
Tolerancia de convergencia	`convTol`	10^{-6}	%
Iteraciones para la convergencia	`convIter`	30	
Tolerancia de refinamiento	`refineTol`	10^{-2}	%
Iteraciones para el refinamiento	`refineIter`	20	
Refinamientos por individuo	`nRefine`	5	

5.2.1. Escenario EO1

Los resultados obtenidos se han calculado basándose en la solución de la simulación EF1-EP1. El instante inicial de la misión se ha establecido en $t = 360$ h y el nodo incial y final coinciden en $\boldsymbol{x} = (200, -300, -0.25)^T$. Al tratarse de un periodo tan reducido y haber utilizado el flujo

lento, la posición horizontal de las partículas es prácticamente estática a lo largo de toda la misión. Sin embargo, la implementación es capaz de trabajar con flujos más veloces o misiones más largas, como se verá en el escenario EO6, donde el desplazamiento horizontal de las cianobacterias tiene un efecto importante.

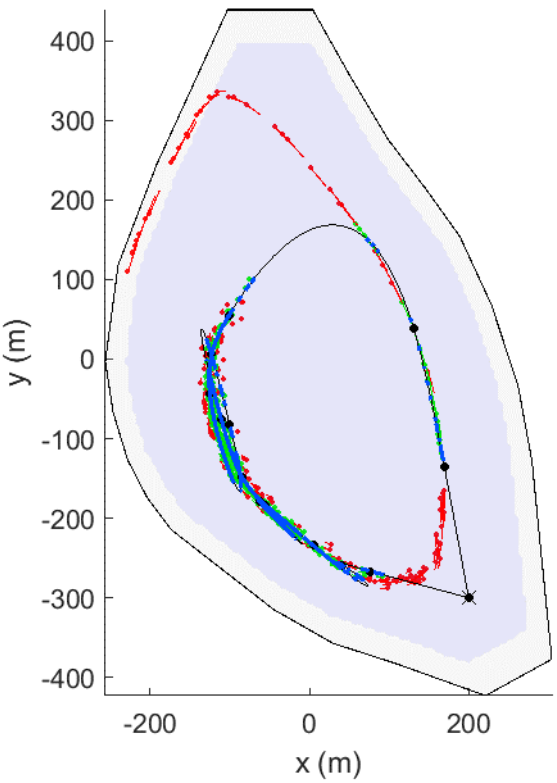
Los resultados obtenidos se reflejan en la Tabla 8. La trayectoria observa 353 partículas de las 500 partículas que hay dentro del dominio de la masa de agua en el último instante de la misión (todas las simuladas en este caso), lo cual representa un 70.60% del total de partículas en el dominio. Sin embargo, no todas las partículas que se hallan dentro del dominio del fluido se encuentran en el interior del espacio de búsqueda, ya que las zonas prohibidas son más restrictivas. Por lo tanto, realmente se observa un 72.93% de aquellas partículas que se están en el interior del espacio de búsqueda, 484 partículas en total. Estos son los porcentajes que se reflejan en la Tabla 8 y en las sucesivas. Hay que tener en cuenta que las soluciones iniciales por lo general no solo no cumplen las restricciones, sino que además es muy común que ni siquiera la mejor de las condiciones iniciales encuentre partículas a lo largo de su trayectoria.

Tabla 8: Resultados de EO1.

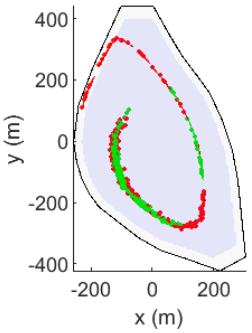
Valor de la función objetivo	-3.4131×10^5
Duración de la trayectoria	9730.8 s (2 h 42 min 10.8 s)
Longitud de la trayectoria	1961.6 m
Partículas vistas	353 (70.60% / 72.93%)
Generaciones	940
Tiempo de ejecución del algoritmo de optimización	7010.7 s (1 h 56 min 50.7 s)

Como se ha comentado en la Sección 5.1.3.2, la creación de los trabajadores paralelos puede incrementar ligeramente el tiempo de cómputo pero solo es necesario crearlos una vez, de modo que si se ejecutan varios escenarios sucesivamente, este tiempo no afecta a ninguna de las ejecuciones exceptuando la primera. El tiempo de cómputo, sin incluir la creación del grupo de trabajadores paralelo (parallel pool), es el que se refleja también en la Tabla 8.

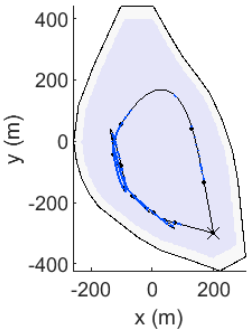
La trayectoria en el plano horizontal de la solución devuelta por el algoritmo en una ejecución del planificador se representa en la Figura 45, y el componente vertical a lo largo del tiempo en la misma trayectoria se muestra en la Figura 46. En ambos casos se han representado en verde las trayectorias de las partículas observadas a lo largo de la misión, en rojo las trayectorias de las partículas que no han sido vistas y en negro la trayectoria del USV obtenida como resultado óptimo, marcando con puntos los nodos de la spline y con una cruz el comienzo de la trayectoria. Los tramos azules de la spline representan los tramos de la trayectoria en los que se detectan partículas. Finalmente, en la Figura 46 se han incluido unas líneas de puntos que representan el rango vertical del sensor. Esto no se ha incluido en la solución en el plano horizontal debido a que los rangos no son suficientemente amplios para ser apreciados a las escalas representadas.



(a) Trayectoria y partículas superpuestas.



(b) Partículas.



(c) Trayectoria.

Figura 45: Solución de EO1 en el plano horizontal.

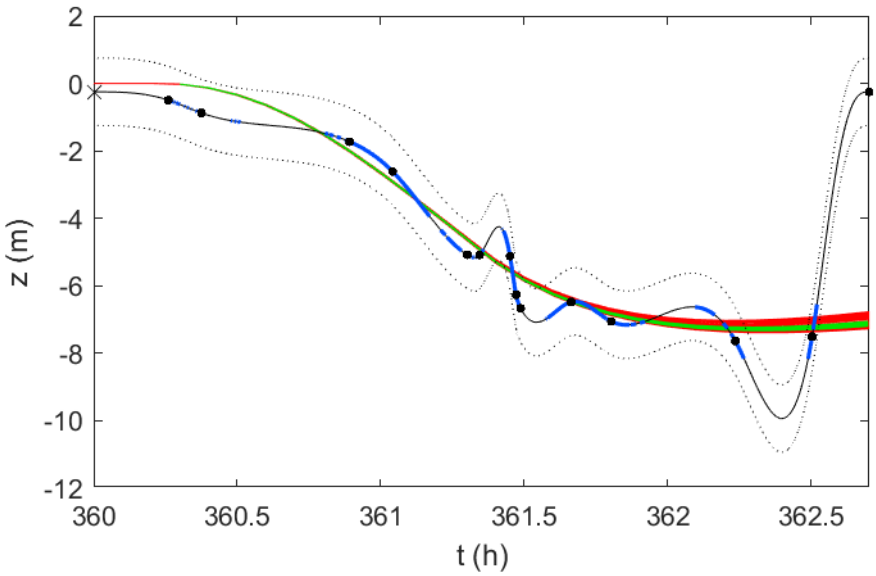
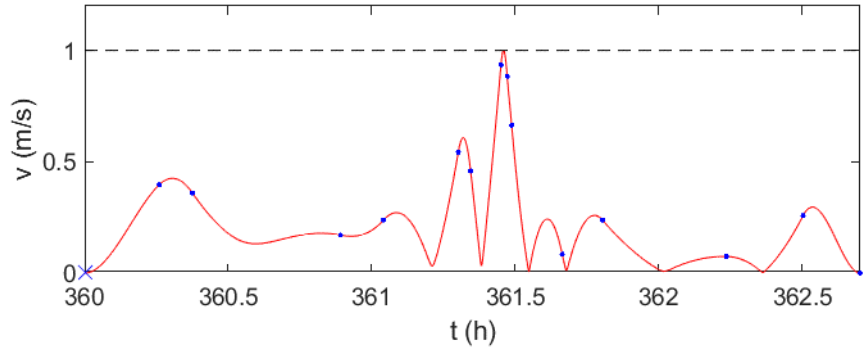


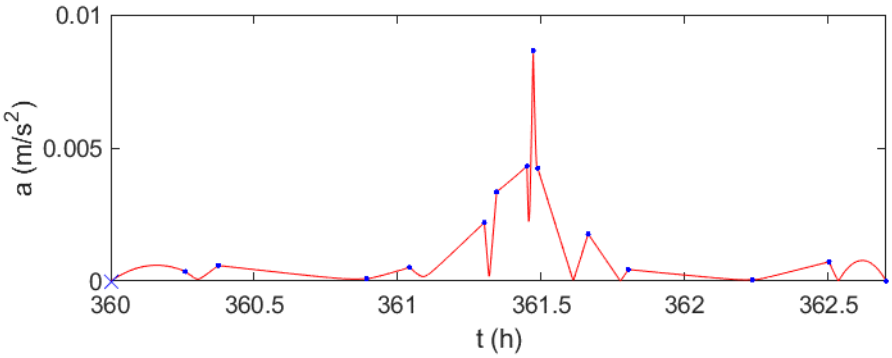
Figura 46: Solución de EO1 en el plano vertical.

En el plano horizontal las trayectorias de las partículas tienen una longitud prácticamente despreciable debido a que las aguas son de flujo lento, especialmente en el centro del lago, pero la migración causada por la flotabilidad sí que tiene un efecto importante en el desplazamiento vertical. Esto provoca que el ajuste óptimo en el plano vertical sea más complicado de obtener que en el plano horizontal. También es de resaltar la gran diferencia que existe en la escala de ambos planos: el horizontal está en el orden de los cientos de metros, mientras que los desplazamientos verticales se limitan al orden de las unidades.

Se observa en la Figura 47, que muestra las magnitudes de S_v y S_a a lo largo del tiempo, que se cumplen las restricciones impuestas en la velocidad y la aceleración, que se han establecido en $1\ \mathrm{m/s}$ y $1\ \mathrm{m/s^2}$, respectivamente. Se marca el límite de velocidad con una línea discontinua, pero no el de la aceleración por estar los valores muy por debajo del límite permitido.



(a) Magnitud de la velocidad.



(b) Magnitud de la aceleración.

Figura 47: Magnitudes de las derivadas de la trayectoria solución de EO1.

La convergencia del método, no específicamente de esta solución particular, se ve reflejada en la Figura 48, la cual muestra la media del óptimo de cada una de las generaciones, en la línea continua, y el intervalo de confianza simétrico del 95% mediante las líneas discontinuas. Además, en la Tabla 9 se muestran las estadísticas del número de generaciones y del tiempo de cómputo necesarios para alcanzar la convergencia según las condiciones expuestas en la Sección 5.1.2.6. Estas estadísticas han sido obtenidas ejecutando el algoritmo 8 veces.

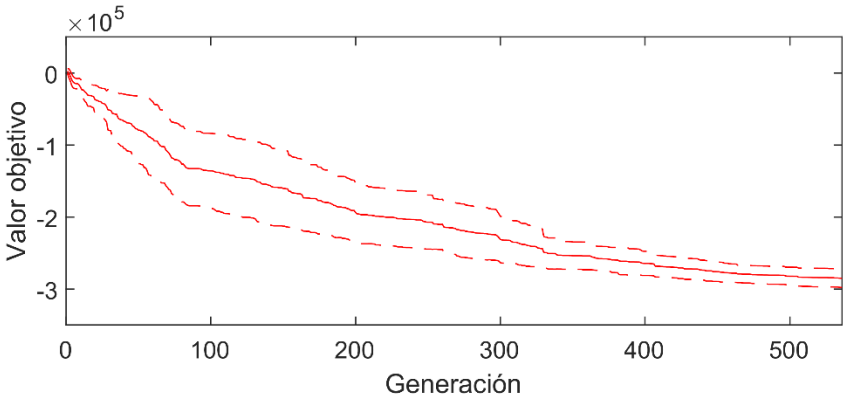


Figura 48: Progresión del óptimo de la solución de EO1.

Tabla 9: Estadísticas de cómputo de EO1.

	Media	Varianza
Generaciones	718.67	1.2595×10^4
Tiempo de cómputo	4067.1 s	3.8947×10^5 s^2

La curva de convergencia se vuelve más horizontal a medida que se incrementan las iteraciones. Esto es debido a que, conforme el algoritmo progresa, resulta más difícil encontrar soluciones cada vez mejores. También se aprecia que, en este caso, el rango del intervalo de confianza decrece al aproximarse a las últimas generaciones, lo cual indica que hallar una solución para este escenario es sencillo y la mayoría de las ejecuciones proporcionan un buen resultado. Nótese que solamente se representan aquellas generaciones de las que se disponen los datos de las 8 ejecuciones, pero las generaciones medias para alcanzar la convergencia, como se indica en la Tabla 9, son mayores. Esto explica la inclinación que se observa en la curva que, por lo general, termina siendo más plana en ejecuciones individuales, como se muestra en la curva de convergencia de la solución proporcionada, que se exhibe en la Figura 49.

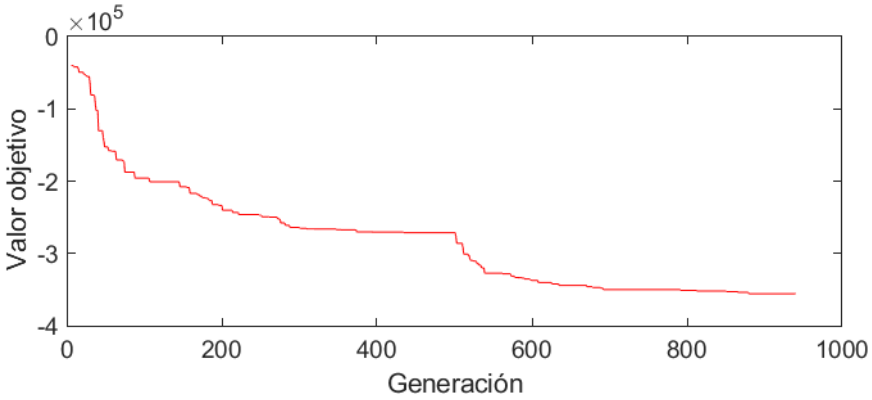


Figura 49: Progresión del óptimo de la solución expuesta.

5.2.2. Escenario EO2

El segundo escenario de optimización que se presenta comparte con EO1 la posición inicial y final de la trayectoria, el instante inicial y el escenario de simulación utilizado, pero introduce tres zonas prohibidas adicionales, modeladas con elipses en el plano horizontal, centradas en

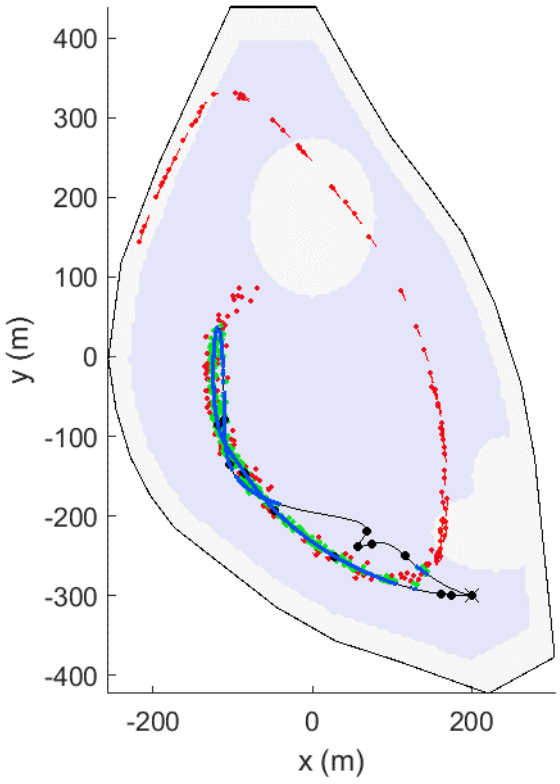
$\boldsymbol{x} = (230, -220, z)^T$, $\boldsymbol{x} = (0, 175, z)^T$ y $\boldsymbol{x} = (240, -150, z)^T$, con semiejes $(80, 50)^T$, $(80, 100)^T$ y $(40, 50)^T$ respectivamente en las direcciones de los ejes x e y.

El mejor de los resultados obtenidos se refleja en la Tabla 10. Resulta evidente que, al añadir restricciones en forma de zonas prohibidas, la cantidad de partículas encontradas sea menor. Además, la trayectoria resultante es más corta tanto en distancia como en duración, lo cual compensa la deficiencia de partículas observadas mediante los objetivos secundarios para lograr alcanzar una función objetivo de mayor calidad que en el escenario anterior.

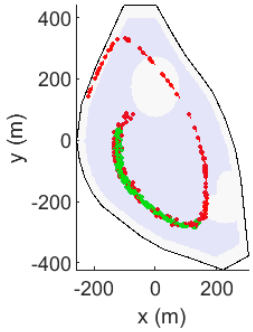
Tabla 10: Resultados de EO2.

Valor de la función objetivo	-3.4259×10^5
Duración de la trayectoria	4322.1 s (1 h 12 min 02.8 s)
Longitud de la trayectoria	1089.4 m
Partículas vistas	348 (69.60% / 74.52%)
Generaciones	1180
Tiempo de ejecución del algoritmo de optimización	6982.6 s (1 h 56 min 22.6 s)

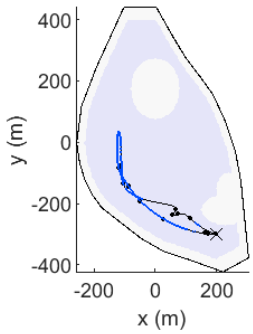
En la Figura 50 se evidencia el efecto que tiene incluir las zonas prohibidas. En lugar de trazar una trayectoria que tienda a la forma circular, se opta por recorrer solamente uno de los lados de la estela formada por las partículas y regresar al punto donde ha de finalizarse la trayectoria por el mismo lado, doblando las posibilidades de encontrar partículas en esa mitad. Se observa en la Figura 51 que la duración de la misión es menor, pero también se siguen las partículas a medida que descienden. Las magnitudes de las derivadas de la trayectoria se muestran en la Figura 52.



(a) Trayectoria y partículas superpuestas.



(b) Partículas.



(c) Trayectoria.

Figura 50: Solución de EO2 en el plano horizontal.

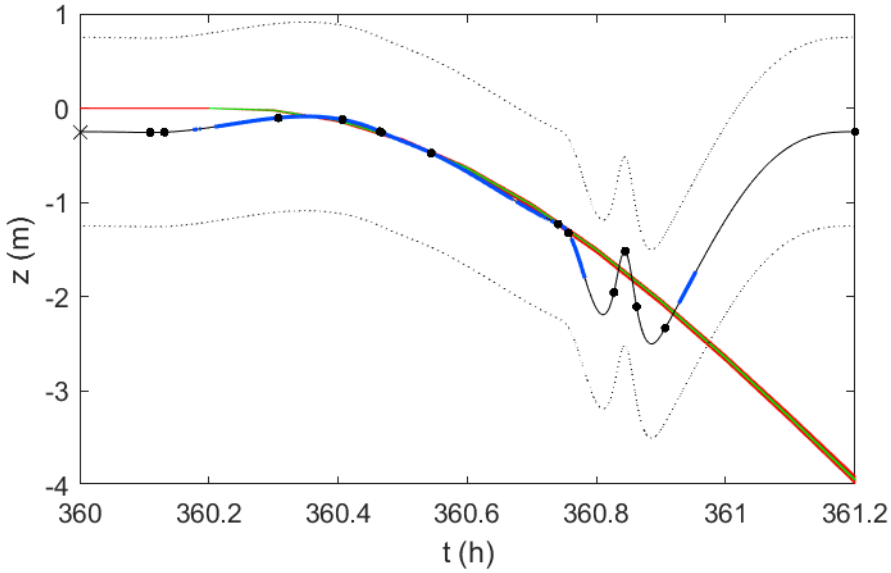
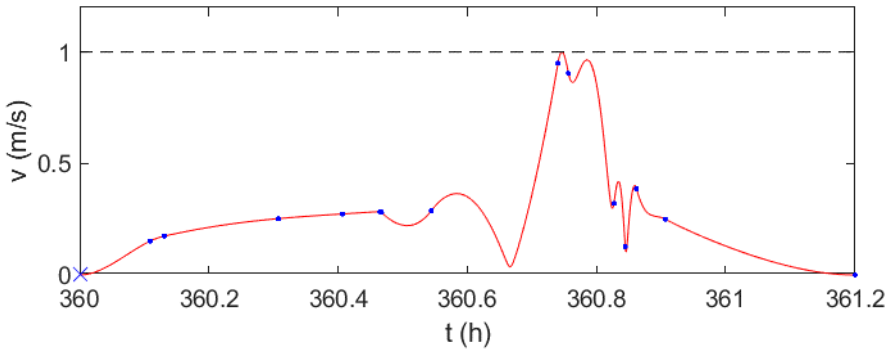
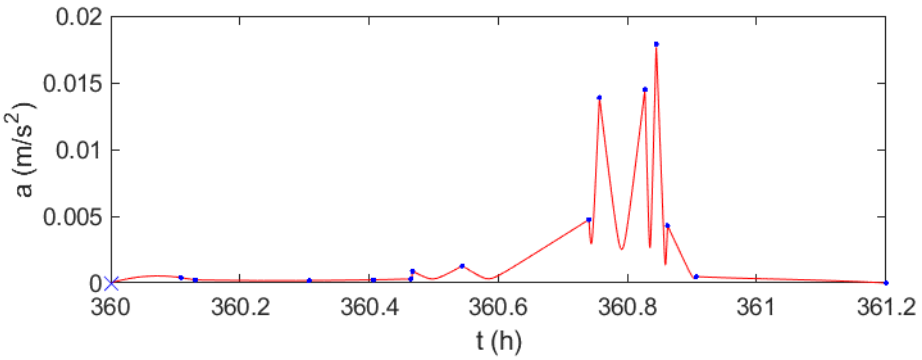


Figura 51: Solución de EO2 en el plano vertical.



(a) Magnitud de la velocidad.



(b) Magnitud de la aceleración.

Figura 52: Magnitudes de las derivadas de la trayectoria solución de EO2.

La convergencia se ve reflejada en la Figura 53, que muestra la media del óptimo de cada una de las generaciones y el intervalo de confianza simétrico del 95%. En la Tabla 11 se exponen las estadísticas de cómputo de la convergencia. En ambos casos las estadísticas han sido obtenidas a partir de 8 ejecuciones del algoritmo.

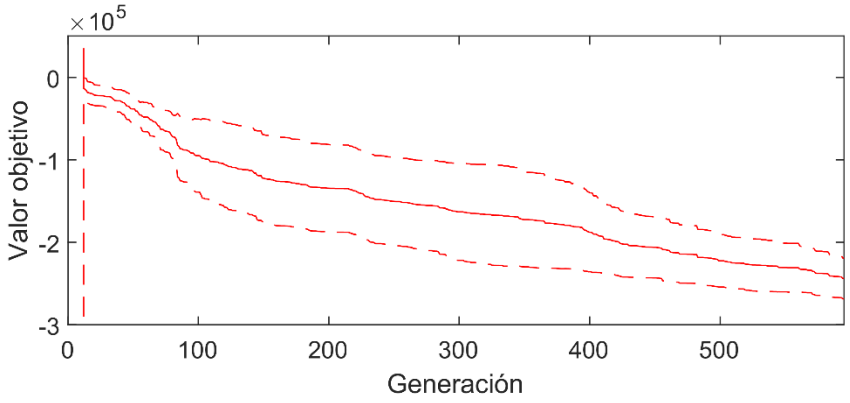


Figura 53: Progresión del óptimo de la solución de EO2.

Tabla 11: Estadísticas de cómputo de EO2.

	Media	Varianza
Generaciones	854.22	1.8942×10^4
Tiempo de cómputo	4842.9 s	5.5731×10^5 s^2

5.2.3. Escenario EO3

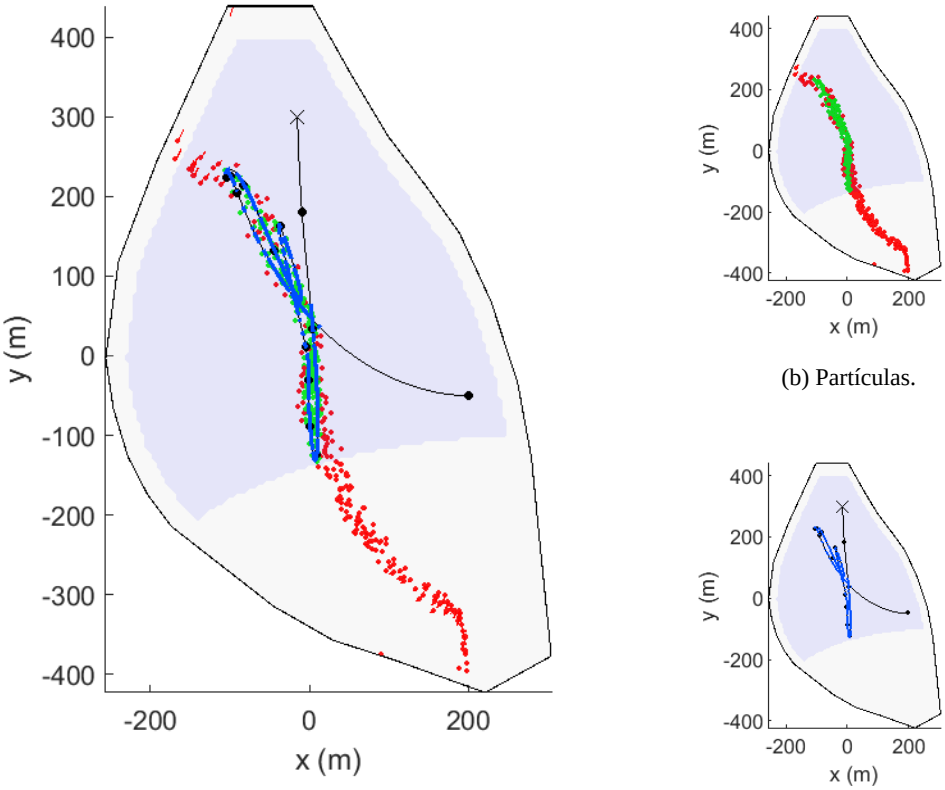
El tercer escenario de optimización presenta una estela de partículas lineal a lo largo del dominio. En lugar de empezar y terminar la trayectoria desde el mismo punto, estos se tratan de puntos diferentes: $\boldsymbol{x} = (-15, 300, -1)^T$ y $\boldsymbol{x} = (200, -50, -1.2)^T$, respectivamente. Se presenta una zona prohibida muy amplia situada en la parte baja del lago, una elipse con semiejes $(500, 200)^T$ en los ejes x e y respectivamente y centrada en $\boldsymbol{x} = (300, -300, z)^T$. El instante inicial de la misión es $t = 33$ h y se ha utilizado una simulación EF1-EP2. Los resultados obtenidos se reflejan en la Tabla 12.

Tabla 12: Resultados de EO3.

Valor de la función objetivo	-2.5219×10^5
Duración de la trayectoria	5325.3 s (1 h 28 min 45.3 s)
Longitud de la trayectoria	1484.7 m
Partículas vistas	259 (55.11% / 77.54%)
Generaciones	889
Tiempo de ejecución del algoritmo de optimización	5787.8 s (1 h 36 min 27.8 s)

En la Figura 54 se muestra cómo la trayectoria del USV se ajusta a la estela linear de partículas, y cómo incluso la recorre más de una vez por lugares ligeramente distintos para maximizar la cantidad de partículas vistas. También se aprecia cómo se aproxima a la zona prohibida sin llegar a entrar en ella, a pesar de que la estela conduzca en esa dirección. Al tratarse de una zona prohibida tan amplia, se limita en gran medida la cantidad de partículas a las que se puede acceder.

Por otra parte, la trayectoria vertical de la sonda del USV se mantiene al nivel de las partículas, que en este escenario no sufre grandes cambios, como puede comprobarse en la Figura 55.



(a) Trayectoria y partículas superpuestas.

(b) Partículas.

(c) Trayectoria.

Figura 54: Solución de EO3 en el plano horizontal.

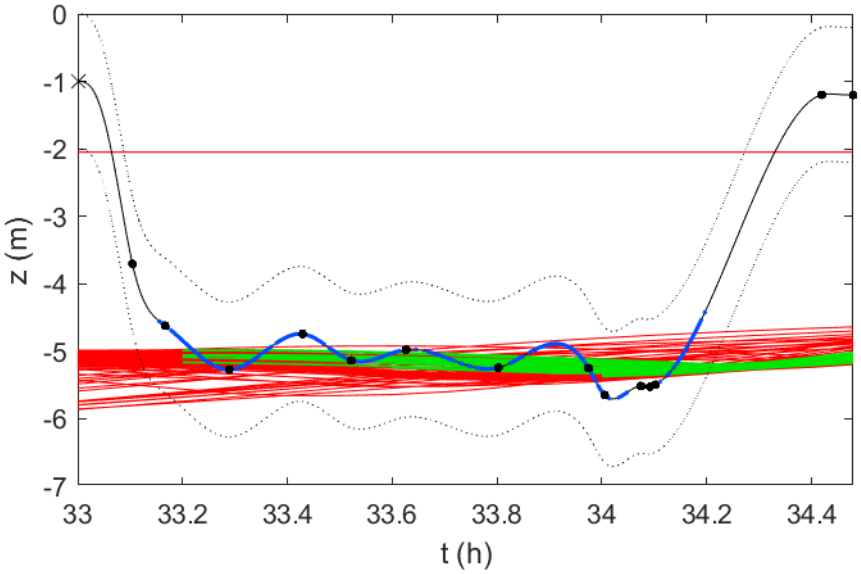
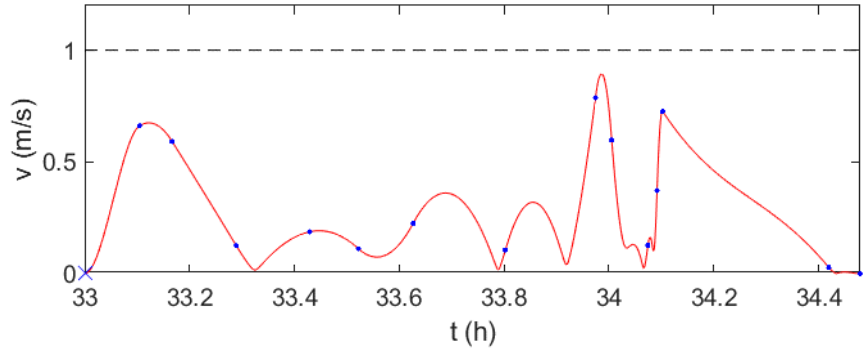
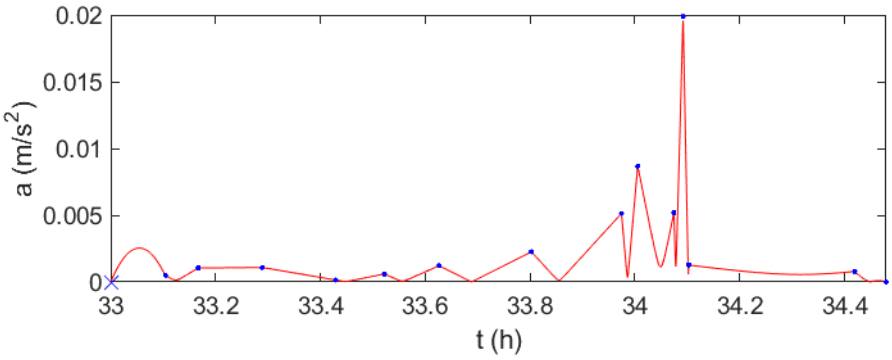


Figura 55: Solución de EO3 en el plano vertical.



(a) Magnitud de la velocidad.



(b) Magnitud de la aceleración.

Figura 56: Magnitudes de las derivadas de la trayectoria solución de EO3.

La convergencia se muestra en la Figura 57, que representa la media del óptimo de cada una de las generaciones y el intervalo de confianza simétrico del 95%. En la Tabla 13 se exponen las estadísticas de cómputo de la convergencia. En ambos casos las estadísticas han sido obtenidas a partir de 8 ejecuciones del algoritmo.

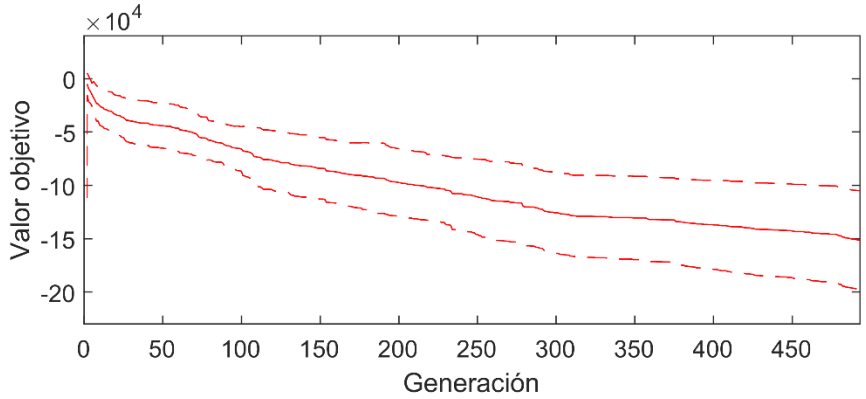


Figura 57: Progresión del óptimo de la solución de EO3.

Tabla 13: Estadísticas de cómputo de EO3.

	Media	Varianza
Generaciones	835.78	4.3747×10^4
Tiempo de cómputo	4696.7 s	1.8744×10^6 s^2

5.2.4. Escenario EO4

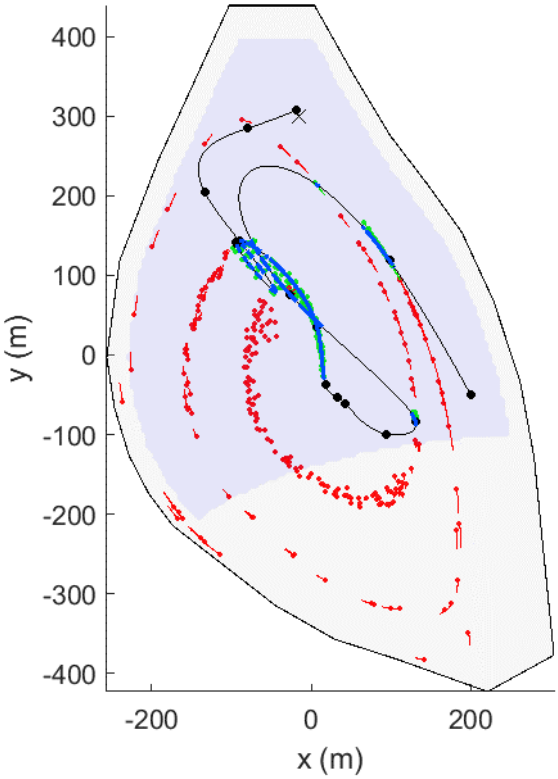
El cuarto escenario de optimización solamente difiere del anterior en que el tiempo de inicio es distinto. En lugar de comenzar tan pronto como en el caso anterior ($t = 33$ h), se retrasa el inicio de la misión hasta $t = 291$ h.

Los resultados obtenidos se reflejan en la Tabla 14. Al haberse producido una mayor dispersión de las partículas en el lago, debido al retraso del tiempo de inicio, encontrar partículas presenta un problema mayor, de modo que el valor de la función objetivo obtenido es algo peor que en el caso anterior.

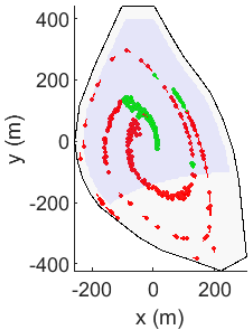
Tabla 14: Resultados de EO4.

Valor de la función objetivo	-2.1754×10^5
Duración de la trayectoria	8833.9 s (2 h 27 min 13.9 s)
Longitud de la trayectoria	1623.4 m
Partículas vistas	228 (48.51% / 58.76%)
Generaciones	1027
Tiempo de ejecución del algoritmo de optimización	5714.2 s (1 h 35 min 14.2 s)

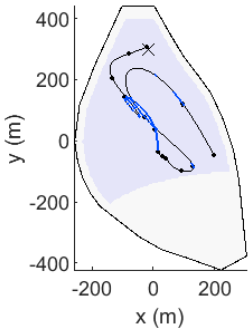
En la Figura 58 se muestra que la dispersión de las partículas ya no es linear, lo cual, por un lado, dificulta su seguimiento y, por otro, provoca que un mayor número de partículas hayan abandonado el dominio. La trayectoria obtenida trata de seguir una de las estelas más densas y parte de otras estelas menos concentradas, pero evita siempre de forma efectiva la gran zona prohibida de la parte inferior del dominio. Puede comprobarse en la Figura 59 que de nuevo las variaciones verticales de las partículas en este escenario son leves. La Figura 60 muestra que se cumplen las restricciones impuestas en la velocidad y la aceleración.



(a) Trayectoria y partículas superpuestas.



(b) Partículas.



(c) Trayectoria.

Figura 58: Solución de EO4 en el plano horizontal.

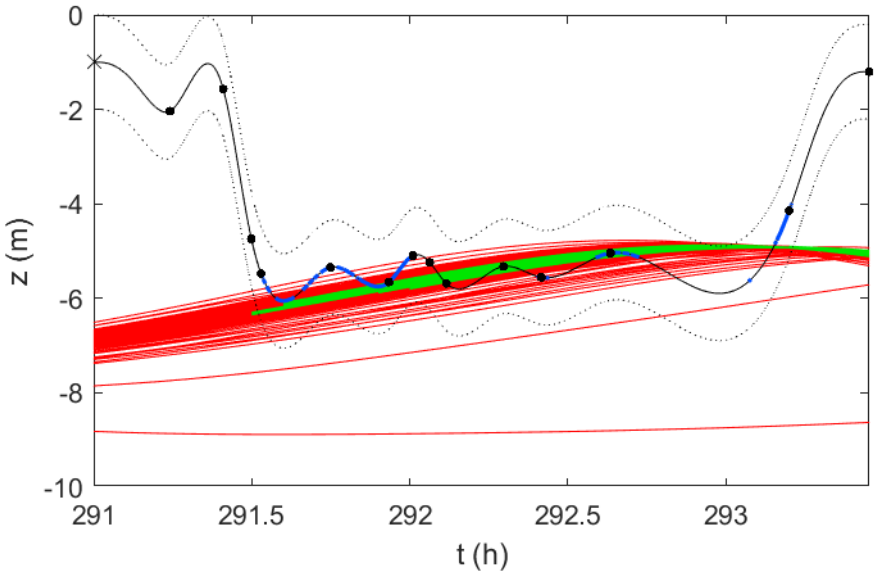
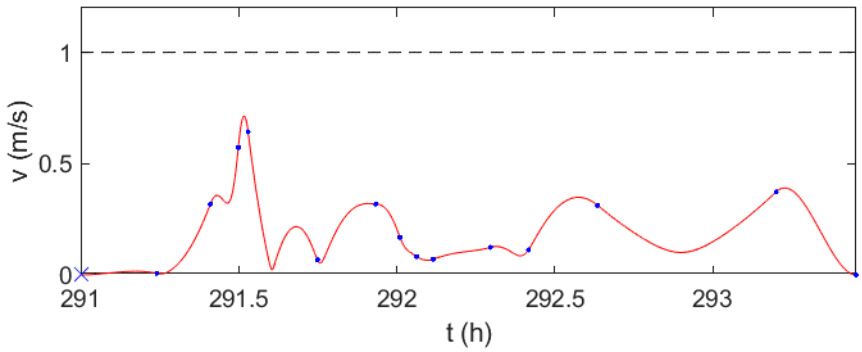
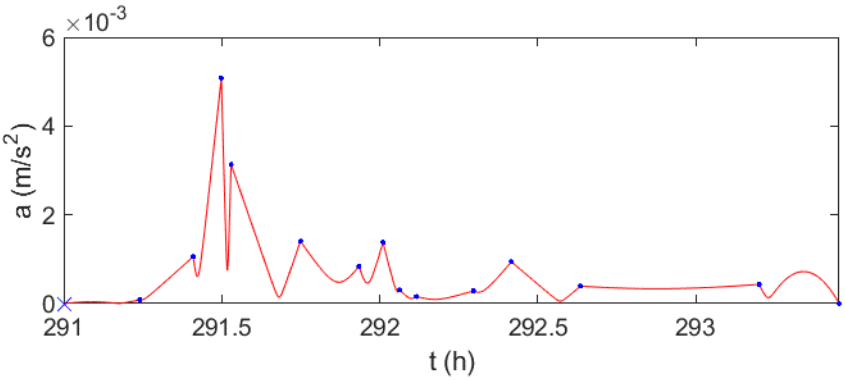


Figura 59: Solución de EO4 en el plano vertical.



(a) Magnitud de la velocidad.



(b) Magnitud de la aceleración.

Figura 60: Magnitudes de las derivadas de la trayectoria solución de EO4.

La convergencia se ve reflejada en la Figura 61, que representa la media del óptimo de cada una de las generaciones y el intervalo de confianza simétrico del 95%. En la Tabla 15 se muestran las estadísticas de cómputo de la convergencia. En ambos casos las estadísticas han sido obtenidas a partir de 8 ejecuciones del algoritmo.

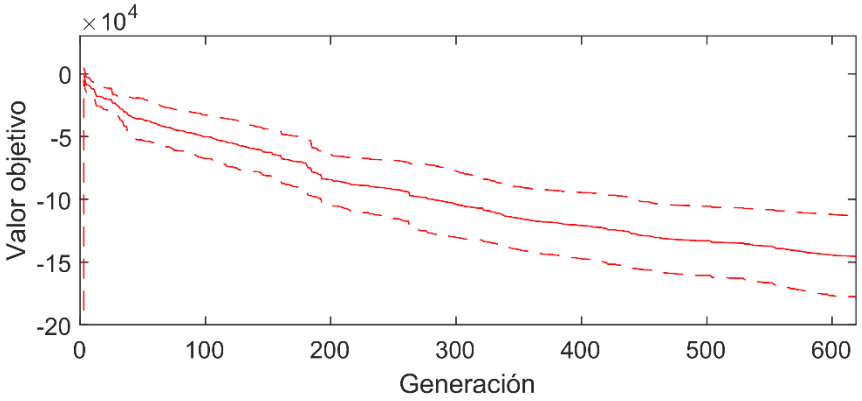


Figura 61: Progresión del óptimo de la solución de EO4.

Tabla 15: Estadísticas de cómputo de EO4.

	Media	Varianza
Generaciones	903.89	2.7445×10^4
Tiempo de cómputo	5216.4 s	1.1630×10^6 s^2

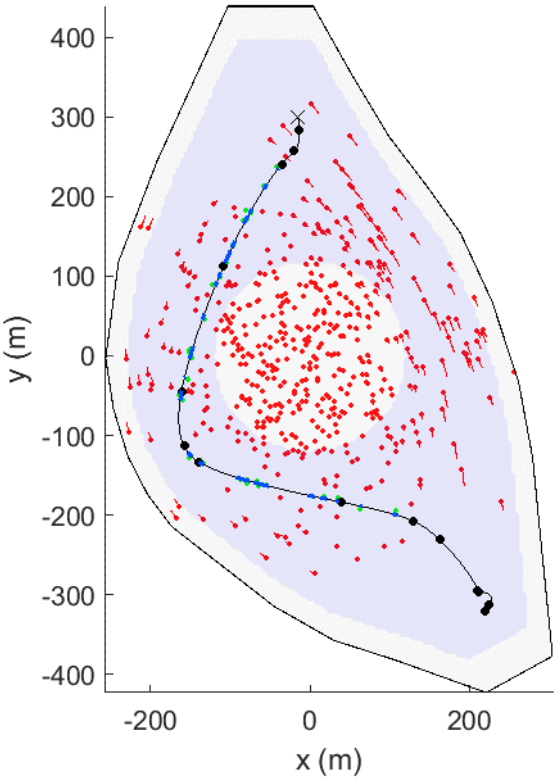
5.2.5. Escenario EO5

El escenario número cinco presenta una distribución muy dispersa de las partículas, de modo que es de esperar una mayor dificultad a la hora de alcanzar buenos valores de la función objetivo. Existe una zona prohibida circular, de radio 120 m situada en $\boldsymbol{x} = (0, 0, z)^T$, centrada en la media de la dispersión inicial de las partículas, resultado de la simulación EF1-EP3. Los puntos inicial y final de la trayectoria son $\boldsymbol{x} = (-15, 300, -1)^T$ y $\boldsymbol{x} = (220, -320, -2.2)^T$, respectivamente. El instante de inicio de la misión es $t = 2$ h. Los resultados obtenidos se reflejan en la Tabla 16.

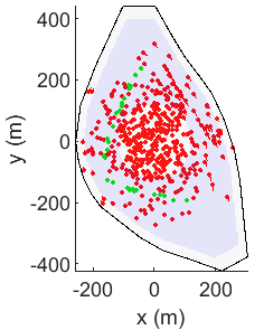
Tabla 16: Resultados de EO5.

Valor de la función objetivo	-2.6577×10^4
Duración de la trayectoria	4502.9 s (1 h 15 min 02.9 s)
Longitud de la trayectoria	919.7 m
Partículas vistas	32 (6.41% / 13.39%)
Generaciones	537
Tiempo de ejecución del algoritmo de optimización	3095.5 s (0 h 51 min 35.5 s)

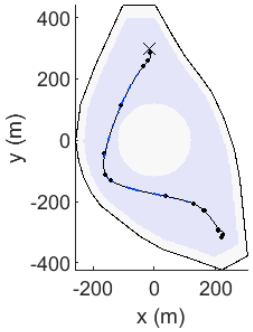
Como puede observarse en la Figura 62, existe una zona prohibida en el centro del lago que fuerza al USV a tomar trayectorias que la rodeen por los laterales, al tener situado el principio y el final en extremos opuestos de la zona prohibida. Debido a que las partículas no se encuentran alineadas en estelas evidentes, sino dispersas por el dominio entero, la trayectoria óptima tampoco sigue una trayectoria clara, más que la impuesta por los puntos inicial y final. La mayor parte de las partículas se encuentran en la zona prohibida del centro del lago, lo cual dificulta la toma de muestras. La Figura 63 representa la trayectoria en el plano vertical y la Figura 64 las magnitudes de las derivadas de la spline, que cumplen las restricciones.



(a) Trayectoria y partículas superpuestas.



(b) Partículas.



(c) Trayectoria.

Figura 62: Solución de EO5 en el plano horizontal.

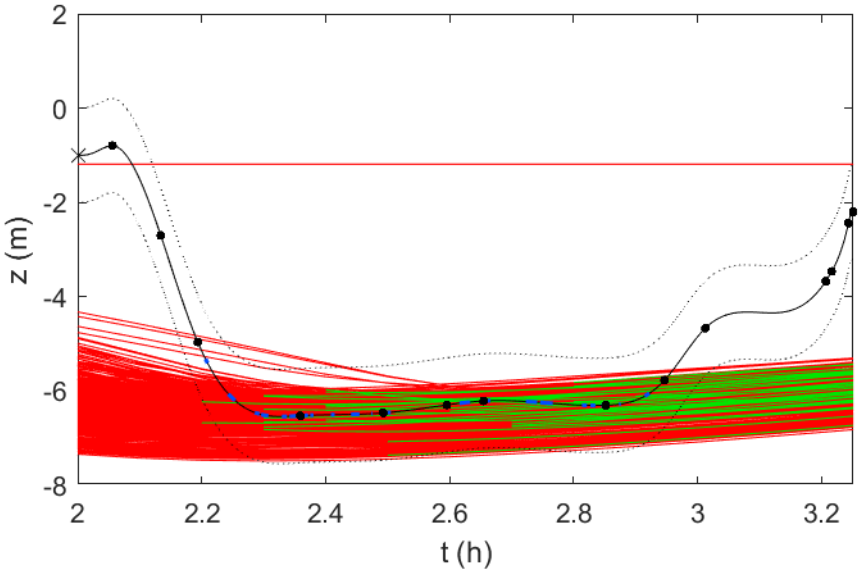
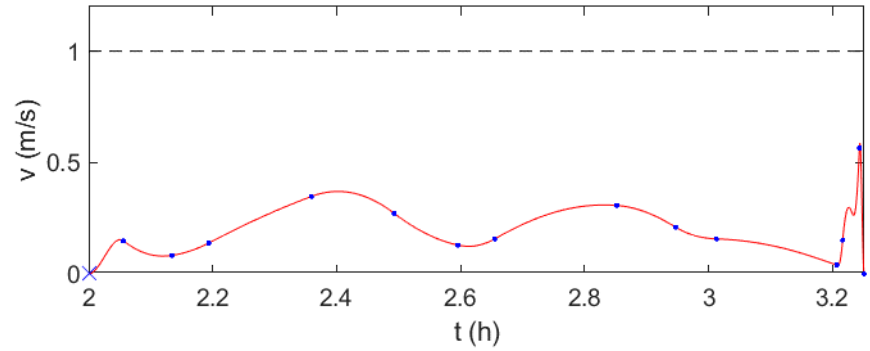
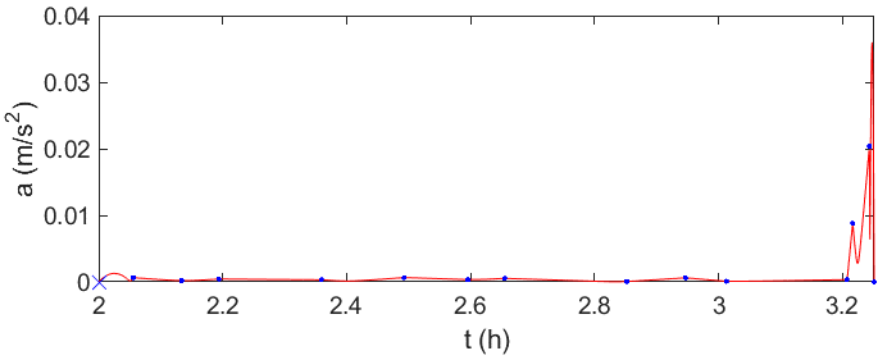


Figura 63: Solución de EO5 en el plano vertical.



(a) Magnitud de la velocidad.



(b) Magnitud de la aceleración.

Figura 64: Magnitudes de las derivadas de la trayectoria solución de EO5.

La convergencia se muestra en la Figura 65, que representa la media del óptimo de cada una de las generaciones y el intervalo de confianza simétrico del 95%. En la Tabla 17 se exponen las estadísticas de cómputo de la convergencia. En ambos casos las estadísticas han sido obtenidas a partir de 8 ejecuciones del algoritmo.

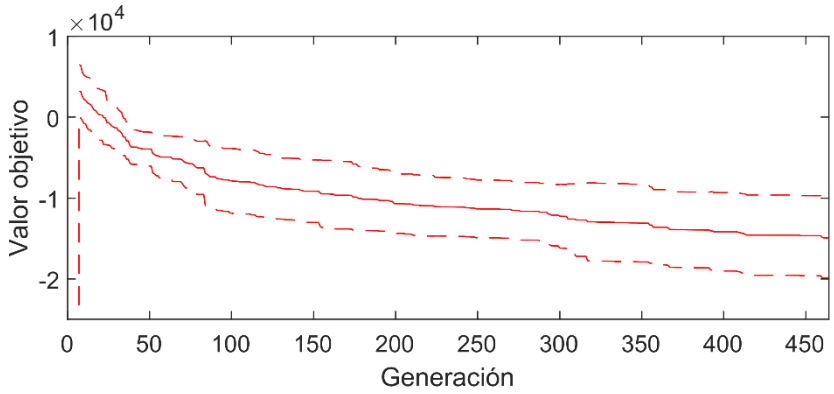


Figura 65: Progresión del óptimo de la solución de EO5.

Tabla 17: Estadísticas de cómputo de EO5.

	Media	Varianza
Generaciones	622.00	6.8540×10^3
Tiempo de cómputo	3436.9 s	2.9866×10^5 s^2

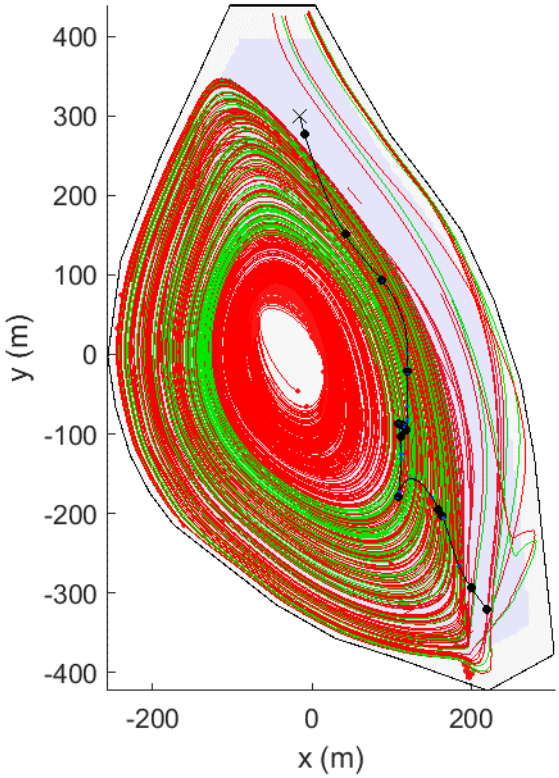
5.2.6. Escenario EO6

Finalmente se presentan los resultados obtenidos utilizando el último escenario de optimización. Esencialmente es igual a EO5, con la única diferencia de que, en lugar de utilizar el flujo lento (EF1), se utiliza el flujo rápido (EF2). Esto pone de manifiesto una situación que difiere de lo realista debido a una limitación del método propuesto. La velocidad del flujo, despreciable en los casos anteriores en cuanto al control del USV, en este caso sí que debería jugar un papel importante y no lo hace, debido a que la función objetivo no tiene en cuenta la velocidad del fluido subyacente y en su lugar entiende la velocidad del USV solamente en un marco absoluto. Por ello, esta velocidad mantiene su máximo de $1\ \mathrm{m/s}$ en el proceso de optimización al mismo tiempo que atraviesa zonas donde el fluido en la superficie tiene una velocidad de cerca de $5\ \mathrm{m/s}$. Esto implica que al atravesar dichas zonas, el USV ha de moverse, relativo al agua, a una velocidad de entre 4 y $6\ \mathrm{m/s}$, lo cual no se encuentra a su alcance. Los resultados obtenidos se reflejan en la Tabla 18.

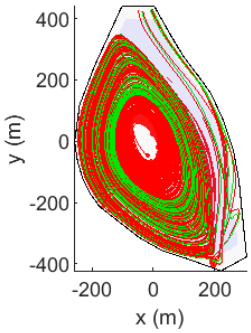
Tabla 18: Resultados de EO6.

Valor de la función objetivo	-3.2900×10^4
Duración de la trayectoria	3331.6 s (0 h 55 min 31.6 s)
Longitud de la trayectoria	768.0 m
Partículas vistas	37 (14.57% / 57.81%)
Generaciones	739
Tiempo de ejecución del algoritmo de optimización	5754.5 s (1 h 35 min 54.5 s)

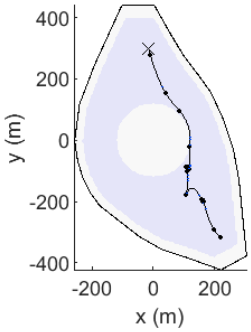
La Figura 66 muestra los resultados obtenidos. Se observa que las partículas vistas tienen tiempo de dar toda la vuelta al vórtice, mientras que el USV permanece exclusivamente en uno de los lados. Se aprecia que, a pesar de que las trayectorias de las partículas hacen parecer que hay una concentración alta en todo el dominio, esto solamente se debe a su rápido movimiento, ya que en realidad están más dispersas incluso que en el escenario anterior. Unos resultados de la simulación más detallados pueden observarse en la Sección 3.2.2.3. Debido a las similitudes entre este escenario y el anterior, las trayectorias óptimas halladas se asemejan mucho en que ambas se mantienen a un lado del vórtice y la trayectoria se dirige sin demasiadas desviaciones hasta el punto final establecido, a pesar de la diferencia en la velocidad de flujo.



(a) Trayectoria y partículas superpuestas.



(b) Partículas.



(c) Trayectoria.

Figura 66: Solución de EO6 en el plano horizontal.

También se observa una cantidad comparable de partículas encontradas. Esto a causa de que, por un lado, hay una reducción en la cantidad de partículas que permanecen en el dominio, debido a que un flujo más veloz es capaz de expulsar las partículas del dominio fluido con más rapidez, y a que el flujo vertical descendente logra hundir a las cianobacterias que antes flotaban; mientras que, por otro lado, la mayor movilidad de las partículas proporciona más posibilidades de ser halladas. Esto puede comprobarse en la Figura 67, donde las partículas alcanzan profundidades que se encuentran fuera del alcance de la sonda del USV, que se ha establecido en el intervalo $[-10, 0]$ m (y dentro de la restricción vertical de las zonas prohibidas). En esta figura se observan algunas trayectorias que terminan antes de llegar al final del intervalo temporal. Esto se debe a que esas partículas salen del dominio del fluido antes de completar la simulación.

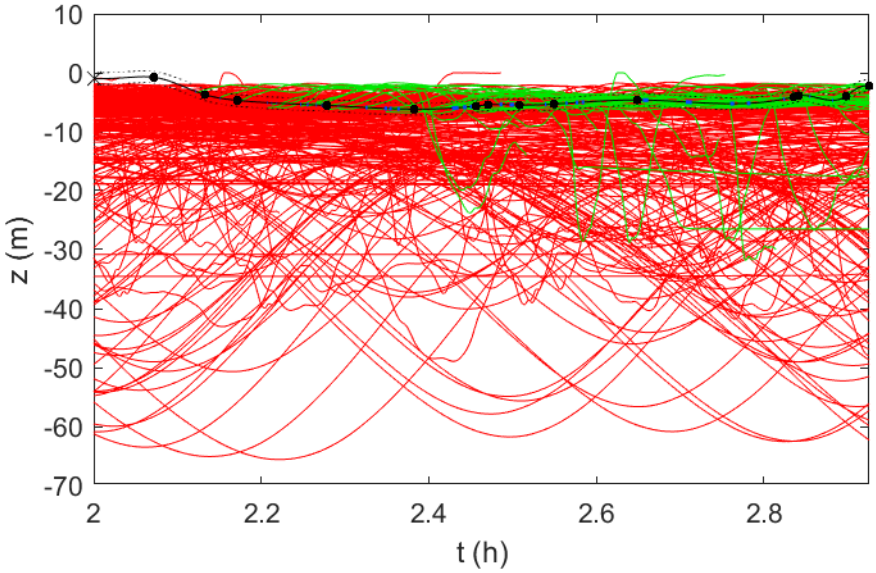
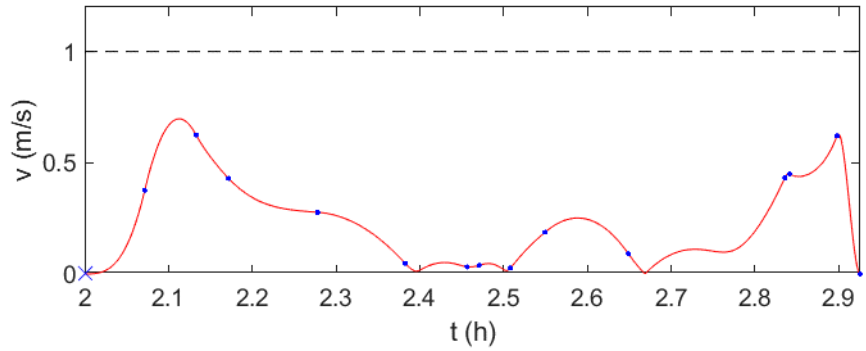
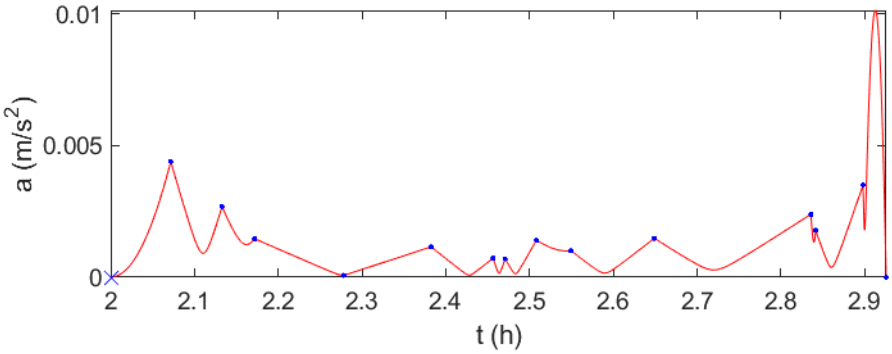


Figura 67: Solución de EO6 en el plano vertical.



(a) Magnitud de la velocidad.



(b) Magnitud de la aceleración.

Figura 68: Magnitudes de las derivadas de la trayectoria solución de EO6.

La convergencia se muestra en la Figura 69, que representa la media del óptimo de cada una de las generaciones y el intervalo de confianza simétrico del 95%. En la Tabla 19 se exponen las

estadísticas de cómputo de la convergencia. En ambos casos las estadísticas han sido obtenidas a partir de 8 ejecuciones del algoritmo.

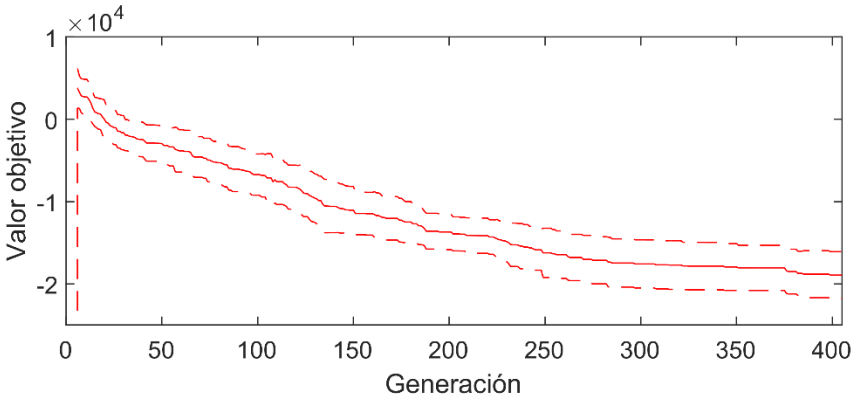


Figura 69: Progresión del óptimo de la solución de EO6.

Tabla 19: Estadísticas de cómputo de EO6.

	Media	Varianza
Generaciones	578.11	1.5503×10^4
Tiempo de cómputo	4899.5 s	4.6388×10^5 s^2

6. Conclusiones

Este proyecto ha sido estructurado en dos partes diferentes y complementarias. En primer lugar, se ha realizado el modelado y simulación del transporte de cianobacterias en aguas lénticas y, en segundo lugar, empleando los resultados de la simulación, se ha optimizado el desplazamiento que debería realizar un USV para poder monitorizar las zonas con una elevada concentración de cianobacterias, utilizando un algoritmo genético para este fin, y codificando la trayectoria del USV mediante splines.

A continuación, se describen las conclusiones que se pueden extraer del trabajo realizado en cada parte y algunas líneas de trabajo que se podrían explorar en un futuro.

6.1. Modelado y simulación del transporte de cianobacterias

El modelo del transporte de cianobacterias se ha realizado en un dominio tridimensional e irregular, que permite definirlo mediante archivos CAD. El flujo del agua en este modelo está limitado a flujo laminar y estacionario, y la posición de las colonias de cianobacterias se calcula determinísticamente. Esto quiere decir que depende en gran medida de su estado inicial y que cualquier estado inicial dentro del dominio produce una solución válida. El modelo tiene en cuenta el transporte proveniente del flujo del fluido, así como la migración vertical de las colonias, que se produce debido a variaciones en la densidad celular al recibir diferentes cantidades de luz. También se consideran las variaciones en la cantidad de luz en diversos momentos del día y a diferentes profundidades.

Además, para comprobar el funcionamiento de los modelos, se ha simulado el comportamiento de las cianobacterias sobre una masa de agua ficticia, con dos condiciones de velocidad de flujo de entrada diferente y múltiples localizaciones iniciales de las partículas. Los resultados de estas simulaciones muestran un comportamiento adecuado y explicable, asociado al flujo de fluido en aguas lénticas esperable, las condiciones lumínicas utilizadas, y a las distribuciones iniciales de las cianobacterias propuestas.

6.1.1. Trabajo futuro

Existen algunas mejoras en el modelo que pueden realizarse en el futuro. En este proyecto se ha seleccionado el uso de flujo laminar y estático debido a que simplifica el modelo y es una aproximación razonable en grandes masas de agua. Sin embargo, la inclusión de turbulencias en el modelo mejoraría la difusión de las partículas haciendo el modelo más realista. También habría que implementar otros comportamientos de las cianobacterias de las que este modelo carece, como puede ser el crecimiento o decrecimiento de las colonias a lo largo del tiempo. Para ello, puede ser conveniente incluir también el transporte de nutrientes, ya que estos representan uno de los factores de crecimiento de la colonia. Además, este modelo es determinístico, lo cual quiere decir que las posiciones de las cianobacterias son puntuales y conocidas. Podría ser conveniente la implementación de un modelo probabilístico que dé más información sobre la posibilidad de encontrar cianobacterias en un punto dado y modele su distribución en forma de un campo continuo. Esto mejoraría tanto la eficiencia como la significancia de la optimización posterior.

En este proyecto se ha aplicado el modelo sobre un lago ficticio, con niveles de luminosidad ficticios y con distribuciones de cianobacterias ficticias. Otra línea de trabajo futuro sería aplicar el modelo de transporte a un caso real y comprobar la precisión del mismo. Los parámetros utilizados se han tomado de la literatura, lo cual implica que puede haber variaciones en diferentes masas de agua o para diferentes tipos de cianobacterias, de modo que será necesario realizar un estudio de carácter biológico específico para cada caso particular.

6.2. Optimización de las trayectorias de un USV

Habiendo obtenido los resultados de la simulación numérica y, por lo tanto, conociendo la posición simulada de las colonias en cualquier momento del dominio temporal simulado, se procede a la optimización de las trayectorias de un USV, cuyo objetivo es el de recoger muestras de cianobacterias. Las trayectorias se han codificado mediante un tipo de spline especifico desarrollado para este proyecto, la spline 4-3-4, y la optimización se ha realizado tomando los nodos de esta como variables de decisión. El problema planteado también considera la existencia de zonas prohibidas que, por un lado, sirven para definir el dominio físicamente accesible por el USV y, por otro, para incluir zonas de la masa de agua donde la navegación pueda estar vedada.

Para la optimización se ha utilizado un algoritmo genético que minimiza la duración y la distancia total de la trayectoria y maximiza la cantidad de partículas vistas, que es análogo a maximizar la densidad de partículas a lo largo de la trayectoria evitando la repetición de muestras. Como restricciones se impone una duración máxima, que la trayectoria ha de encontrarse por completo dentro de la zona segura, y que la velocidad y aceleración del USV deben hallarse dentro de los límites permitidos. Todo ello implica que la trayectoria puede ser utilizada como señal de referencia en el control de un USV real, ya que respeta su velocidad y aceleración máxima, limita su recorrido a zonas permitidas y alcanzables, y respeta la duración máxima establecida para la misión.

Basándose en los resultados de las simulaciones que se han realizado en la primera parte, se plantean seis escenarios diferentes. En ellos se destacan los efectos que tienen varios parámetros en la trayectoria del USV encontrada, como son la inclusión de zonas prohibidas en el espacio de búsqueda, la variación del instante inicial de la misión de recogida de muestras, o el efecto del flujo del fluido en la trayectoria. En todos los casos se comprueba que el algoritmo es capaz de encontrar soluciones que cumplan las restricciones impuestas, eviten las zonas prohibidas y tengan sentido. Sin embargo, se pone de manifiesto una limitación del método propuesto en simulaciones donde la velocidad del fluido es demasiado alta, ya que un USV real no sería capaz de seguir la trayectoria propuesta por el algoritmo. Esto es debido a que las restricciones de velocidad del USV en este Trabajo de Fin de Máster han sido impuestas en coordenadas absolutas, mientras que las restricciones reales se miden en referencia a la velocidad del flujo por el que se mueve el vehículo.

6.2.1. Trabajo futuro

Existen numerosas líneas de investigación futuras para esta parte del proyecto, ya que hay muchas posibilidades en cada parte de la misma. Por un lado, la codificación utilizada se ha realizado mediante splines 4-3-4, que son de clase C^2 y cumplen las restricciones impuestas en las derivadas de primer y segundo orden en los extremos de la spline. Sin embargo, tienen el inconveniente de ser splines globales, lo cual quiere decir que una variación en uno de los nodos afecta a toda la spline. Existen otro tipo de splines, las splines locales, que al modificar un nodo limitan los cambios a su entorno inmediato. Este tipo de curvas tiene otra serie de inconvenientes, ya que se requiere un mayor número de parámetros para caracterizarlas. Además, las utilizadas comúnmente no cumplen los requisitos de continuidad, de modo que se propone como trabajo futuro el desarrollo de splines de un orden mayor a 3 que cumplan todas las restricciones impuestas siendo al mismo tiempo locales.

Además, existen otras codificaciones de la trayectoria como, por ejemplo, mediante consignas de referencias (velocidad, orientación) que podrían mejorar los resultados. Otras opciones pueden consistir en mantener la codificación en splines, pero realizar la optimización en dos partes: por un lado, el plano horizontal, donde existe menor variación de la localización de las partículas a lo largo del tiempo y, por otro, optimizar el plano vertical, en el cual la migración vertical de las cianobacterias juega un papel importante. En cualquier caso, sería de utilidad añadir unas

restricciones de velocidad que tengan en cuenta el flujo del agua en la que se mueve el USV, ya que de este modo se podrían obtener resultados más realistas en condiciones de flujo rápido.

Adicionalmente, en este proyecto solamente se ha probado un algoritmo de optimización, el algoritmo genético, y únicamente en su variante mono-objetivo. Queda como trabajo futuro realizar una comparación entre diferentes tipos de algoritmos y explorar los efectos de su implementación como algoritmo multi-objetivo, así como un análisis estadístico del efecto que tiene cada uno de los parámetros del algoritmo en la optimización.

Apéndice 1: Código de MATLAB de la Simulación Numérica

```
clear; clc; tic

%=========================================================================
% SETUP
%=========================================================================

% Fluid flow solution (from COMSOL)
dataName = "NavierStokesSolutionFine.txt";

Tday  = 24;     % Day duration (h)
Ilim  = 5.75;   % Irradiance thresshold for light/dark conditions (Wm-2)
zTol  = 1e-3;   % Depth to which the z-level is considered top boundary
cellN = 500;    % Number of particles to be generated

% Physical Parameters % | Units     | Description
                      % +-----------+-------------------------
nu      = 3e-6   ;    % | m2s-1     | Viscosity
rhoF    = 1000   ;    % | kgm-3     | Fluid Density
g       = 9.81   ;    % | ms-2      | Gravity acceleration
dp      = 8e-4   ;    % | m         | Particle diameter
betta   = 1.5e-3 ;    % | s2m-3     | Normative factor in cell density
I0      = 146.43 ;    % | Wm-2      | Photo-inhibition limit
gamma   = -8.3e-3;    % | kgm-3s-1  | Rate of change in cell density (dark)
alpha   = 4.7e-4 ;    % | s-1       | Decay rate of cell density
Cp      = 1037   ;    % | kgm-3     | Minimum cell density
nCell   = 0.10   ;    % | %         | Cell content in colony
nGas    = 0.05   ;    % | %         | Gas content in colony
rhoMuc  = 998    ;    % | kgm-3     | Mucilage density
Imax    = 800    ;    % | Wm-2      | Maximum irradiance
Ki      = 0.8    ;    % | m-1       | Light atenuation coefficient

% Form gravity vector
g = [0,0,-g];

% Time stepping (h)
dt = 0.0005;
tVec = 0:dt:6;
tN = length(tVec);

% Import mesh, fluid velocity (u) and fluid pressure (p)
dataImport = load(dataName);
nodes = dataImport(:,1:3);
u     = dataImport(:,4:6);
N     = size(nodes,1);
clear dataImport;

% Import geometry (to use in boundaries)
model = createpde(1);
importGeometry(model,'lake5.stl');
[geomF, geomV] = model.Geometry.allDisplayFaces();

% Interpolation function for u
interpUx = scatteredInterpolant(nodes(:,1), nodes(:,2),nodes(:,3),u(:,1));
interpUy = scatteredInterpolant(nodes(:,1), nodes(:,2),nodes(:,3),u(:,2));
interpUz = scatteredInterpolant(nodes(:,1), nodes(:,2),nodes(:,3),u(:,3));
interpU = @(x,y,z) [interpUx(x,y,z), interpUy(x,y,z), interpUz(x,y,z)];

% Light irradiance at surface level
Is = @(t) max(Imax*sin(2*pi/Tday*t),0);
```

```
% Heaviside step function
Heaviside = @(x) max(0, x./abs(x));

% Initial cell position (Gaussian dist.)
mu = [0,-200,-5];
sigma = [25,25,5];
xCell = []; rndGenIter = 0;
while size(xCell,1) < cellN
    rndGenIter = rndGenIter + 1;
    sz = cellN-size(xCell,1);
    points = [normrnd(mu(1),sigma(1),sz,1),normrnd(mu(2),sigma(2),sz,1),...
        normrnd(mu(3),sigma(3),sz,1)];
    points = points(inpolyhedron(geomF, geomV, points),:);
    xCell(end+1:end+(size(points,1)),:) = points;
end

% Initial cell density (uniform dist.)
rhoCellFun = @(z) ones(size(z,1),1)*Cp+50*rand(size(z,1),1);
rhoCell = rhoCellFun(xCell(:,3));

% Initialize logs
xCellPrev = xCell;
xCellLog  = zeros(tN, cellN, 3);
upLog     = zeros(tN, cellN, 3);
rhoCLog   = zeros(tN,cellN);

clear nodes u;

%=========================================================================
% TIME LOOP
%=========================================================================
fprintf('Time steps:   0%%');

for n = 1:tN

    if mod(n, round(tN/100))==0
        fprintf('\b\b\b\b%3i%%', round(n/tN*100));
    end

    t = tVec(n);

    % Light irradiance at each cell
    I = Is(t)*exp(Ki*xCell(:,3));

    % Cell density
    parfor i = 1:cellN
        % Light conditions
        if I(i) > Ilim
            dRhoCell = betta*I(i)*exp(-I(i)/I0)+gamma;
        % Dark conditions
        else
            delta    = rhoCell(i) - Cp;
            dRhoCell = -alpha * delta * Heaviside(delta);
        end
        rhoCell(i)  = rhoCell(i) + 3600*dt*dRhoCell;
    end

    % Colony density
    rhoC = rhoCell*nCell*(1-nGas)+rhoMuc*(1-nCell);

    % Interpolate fluid velocity field
    uCell = interpU(xCell(:,1), xCell(:,2), xCell(:,3));
```

```
    % Transport velocity
    uBuoy = dp^2.*rhoC./(18*nu*rhoF).*(1-rhoF./rhoC).*ones(cellN,1)*g;
    up = uCell + uBuoy;

    % Integrate up to get position (explicit Euler)
    xCell = xCell + 3600*dt * up;

    % Impose limits
    % Upper boundary
    xCell(:,3) = min(xCell(:,3),-zTol);

    % Other boundaries
    inDom = inpolyhedron(geomF, geomV, xCell);
    xCell = xCell.*inDom + xCellPrev.*(1-inDom);
    xCellPrev = xCell;

    % Store logs
    xCellLog(n,:,:) = xCell;
    upLog(n,:,:)  = up;
    rhoCLog(n,:)  = rhoC;

end
SimulationTime=toc;

%==========================================================================
% SAVE SOLUTION FOR OPTIMIZATION
%==========================================================================
save('SimulationSolution', 'xCellLog', 'tVec');
```

Apéndice 2: Código de MATLAB de la Optimización

```
clear; clc; tic

%=========================================================================
% PARAMETERS
%=========================================================================

maxV = 1;                   % Max velocity of the boat (m/s)
maxA = 1;                   % Max acceleration of the boat (m/s^2)
maxT = 3*3600;              % Max duration of the path (s)
tStart = 360;               % Start time (h)
node0 = [200,-300,-0.25];   % Initial USV position (m)
nodeF = [200,-300,-0.25];   % Final USV position (m)

forbPoint = [ 230, -220;    % Centers of ellipses forbiden zones (m) (x,y)
                0,  175;
              240, -150];

forbRad   = [  80,   50;    % Semiaxis of ellipses forbiden zones (m) (x,y)
               80,  100;
               40,   50];

% Method's Parameters
nPop = 50 ;                 % Population size
nNodesMin = 4;              % Min number of nodes in the spline
nNodesMax = 15;             % Max number of nodes in the spline
pElite = 0.1;               % Proportion of elite children
pXOver = 0.3;               % Proportion of cross over children
pMutate = 0.5;              % Proportion of mutation children
mutationProb = 0.05;        % Probability of mutation
mutDist = [20,20,2,600];    % Max distance of mutation [x,y,z,t]
convTol = 1e-8;             % Tolerance for convergence (relative error)
convIter = 30;              % Min iterations w/o improvent for convergence
maxIter = 1500;             % Max iterations of the method
refineTol = 1e-4;           % Tolerance for refinement (relative error)
refineIter = 20;            % Min iterations w/o improvement for refinement
nRefine = 5;                % Number of refinements for each member of pop


%=========================================================================
% SETUP
%=========================================================================

% Calculate number of each kind of children
nElite = round(pElite*nPop);
nXOver = round(pXOver*nPop);
nMutate = round(pMutate*nPop);
nInmigrants = nPop - nElite - nXOver - nMutate;

% Import geometry (to use in boundaries)
model = createpde(1);
importGeometry(model,'lake5.stl');
[~, geomV] = model.Geometry.allDisplayFaces();

% 2D boundaries and range in height (search space)
surfPoints = 0.90*geomV(geomV(:,3) > -1e-3, 1:2);
bound = boundary(surfPoints(:,1),surfPoints(:,2));
```

```
% Forbiden zone function (if inside safe zone: -0.5, if outside: 0.5)
forbAdd = @(x,y,z) 1;
for i = 1:size(forbPoint,1)
    p = forbPoint(i,:);
    r = forbRad(i,:);
    forbAdd = @(x,y,z) forbAdd(x,y,z) &...
        (1/r(1)^2*(x-p(1)).^2 + 1/r(2)^2*(y-p(2)).^2) > 1;
end
forbDom = @(x,y,z) (inpolygon(x, y, surfPoints(bound, 1),...
    surfPoints(bound, 2))& z > -10 & z < 0);
forb = @(x,y,z) (forbDom(x,y,z) & forbAdd(x,y,z)) - 0.5;

% Interval for point generation (initial solutions)
intervals = [ min(surfPoints(:,1)) , max(surfPoints(:,1)) ;
              min(surfPoints(:,2)) , max(surfPoints(:,2)) ;
              -10                  , 0 ;
              0                    , maxT ];

% Restrictions
restr = [maxV, maxA, maxT];

% Load simulation solution
load('SimulationSolution')

% Initial population
nNodes = nNodesMin;
pop = generateRandPop(nPop, nNodes, intervals, node0, nodeF);
parfor i = 1:nPop
    pop{i} = setPathFields(pop{i}, forb, xCellLog, tVec, tStart, restr);
end

% Initial values
optim = inf;
refineLastIter = 0;
minObj = zeros(maxIter,1);
meanObj = zeros(maxIter,1);

%================================================================================
% OPTIMIZATION LOOP
%================================================================================
fprintf('\nGeneration:        ')

for iter = 1:maxIter

    fprintf('\b\b\b\b\b\b\b%7d',iter)

    % Get objective function values
    parfor i = 1:nPop
        [pop{i}.obj, pop{i}.parts] = objFun(pop{i}, restr);
    end

    % Get generation's optimum
    objs = cellfun( @(x) x.obj, pop );
    [minObj(iter), minIndex] = min(objs);
    meanObj(iter) = mean(objs(1:end-nInmigrants));

    % Update global optimum
    if minObj(iter) < optim
        optim = minObj(iter);
        elemSol = pop{minIndex};
    end
```

```
% Convergence
if (iter > refineLastIter + convIter) &&...
       abs((minObj(iter-convIter)-optim)/minObj(iter-convIter))<convTol
    break
end

% Refinement
if (iter > refineLastIter + refineIter) &&...
       abs((minObj(iter-refineIter)-optim)/minObj(iter-refineIter))...
       <refineTol &&...
       nNodes + 1 <= nNodesMax

    refineLastIter = iter;
    tempPop = cell(nPop*nRefine, 1);
    for i = 1:nPop
        for j = 1:nRefine
            tempPop{nRefine*i-j+1} =...
                refineNodes(pop{i},forb,xCellLog,tVec,tStart,restr);
        end
    end
    nNodes = nNodes + 1;
    pop = tempPop; clear tempPop
end

% Parent Selection
% (Probability of selection inversely proportional to objFun)
objs = cellfun( @(x) x.obj, pop );
minObjs = min(objs);
maxObjs = max(objs);
normalizedObjs = (objs - minObjs)/(maxObjs - minObjs)*(100-2) + 2;
weights = 1./normalizedObjs; % Range [1/100, 1/2]
parentIndex = randsample(1:length(pop),2*nXOver+nMutate,true,weights);

% Elite Children (Save the best solutions)
[~,order] = sort(cellfun( @(x) x.obj, pop ));
elite = cell(nElite,1);
for i = 1:nElite
    elite{i} = pop{order(i)};
end


% Cross Over
r = rand(ceil(nXOver/2),2);
xOver = cell(nXOver,1);
for i = 1:nXOver/2
   p1 = pop{parentIndex(2*i-1)};
   p2 = pop{parentIndex(2*i)};

   cutoff = sort(round(nNodes*r(i,:)));
   xOver{2*i-1}.x = [p1.x(1:cutoff(1)), p2.x(cutoff(1)+1:cutoff(2)),...
        p1.x(cutoff(2)+1:end)];
   xOver{2*i-1}.y = [p1.y(1:cutoff(1)), p2.y(cutoff(1)+1:cutoff(2)),...
        p1.y(cutoff(2)+1:end)];
   xOver{2*i-1}.z = [p1.z(1:cutoff(1)), p2.z(cutoff(1)+1:cutoff(2)),...
        p1.z(cutoff(2)+1:end)];
   xOver{2*i-1}.t = [p1.t(1:cutoff(1)), p2.t(cutoff(1)+1:cutoff(2)),...
        p1.t(cutoff(2)+1:end)];
   xOver{2*i-1}.t = sort(xOver{2*i-1}.t);
```

```
        xOver{2*i}.x = [p2.x(1:cutoff(1)), p1.x(cutoff(1)+1:cutoff(2)),...
             p2.x(cutoff(2)+1:end)];
        xOver{2*i}.y = [p2.y(1:cutoff(1)), p1.y(cutoff(1)+1:cutoff(2)),...
             p2.y(cutoff(2)+1:end)];
        xOver{2*i}.z = [p2.z(1:cutoff(1)), p1.z(cutoff(1)+1:cutoff(2)),...
             p2.z(cutoff(2)+1:end)];
        xOver{2*i}.t = [p2.t(1:cutoff(1)), p1.t(cutoff(1)+1:cutoff(2)),...
             p2.t(cutoff(2)+1:end)];
        xOver{2*i}.t = sort(xOver{2*i}.t);

        % Fix repeated times by averaging one with previous time
        cs = xOver{2*i-1}.t(1:end-1) == xOver{2*i-1}.t(2:end);
        bracketind = find(cs);
        aux = [0,xOver{2*i-1}.t];
        xOver{2*i-1}.t(bracketind) = (xOver{2*i-1}.t(bracketind) +...
             aux(bracketind))/2;

        cs = xOver{2*i}.t(1:end-1) == xOver{2*i}.t(2:end);
        bracketind = find(cs);
        aux = [0,xOver{2*i}.t];
        xOver{2*i}.t(bracketind) = (xOver{2*i}.t(bracketind) +...
             aux(bracketind))/2;

    end

    % Add one child if nXOver is odd
    if mod((nXOver/2),2)
        p1 = pop{parentIndex(2*nXOver-1)};
        p2 = pop{parentIndex(2*nXOver)};

        cutoff = sort(round(nNodes*r(i+1,:)));
        xOver{nXOver}.x = [p1.x(1:cutoff(1)),p2.x(cutoff(1)+1:cutoff(2)),...
             p1.x(cutoff(2)+1:end)];
        xOver{nXOver}.y = [p1.y(1:cutoff(1)),p2.y(cutoff(1)+1:cutoff(2)),...
             p1.y(cutoff(2)+1:end)];
        xOver{nXOver}.z = [p1.z(1:cutoff(1)),p2.z(cutoff(1)+1:cutoff(2)),...
             p1.z(cutoff(2)+1:end)];
        xOver{nXOver}.t = [p1.t(1:cutoff(1)),p2.t(cutoff(1)+1:cutoff(2)),...
             p1.t(cutoff(2)+1:end)];
        xOver{nXOver}.t = sort(xOver{nXOver}.t);

        % Fix repeated times by averaging one with previous time
        cs = xOver{nXOver}.t(1:end-1) == xOver{nXOver}.t(2:end);
        bracketind = find(cs);
        aux = [0,xOver{nXOver}.t];
        xOver{nXOver}.t(bracketind) = (xOver{nXOver}.t(bracketind) +...
             aux(bracketind))/2;
    end

    % Calculate associated variables
    parfor i = 1:nXOver
        xOver{i} = setPathFields(xOver{i},forb,xCellLog,tVec,tStart,restr);
    end

    % Mutation (Uniform mutation within restricted range)
    mutate = cell(nMutate,1);
    for i = 1:nMutate
        r = rand(4,nNodes);

        tempMutateX = pop{parentIndex(2*nXOver+i)}.x;
        tempMutateY = pop{parentIndex(2*nXOver+i)}.y;
        tempMutateZ = pop{parentIndex(2*nXOver+i)}.z;
        tempMutateT = pop{parentIndex(2*nXOver+i)}.t;
```

```
        % Mutate
        tempMutateX(r(1,:) < mutationProb) = tempMutateX(r(1,:)...
            < mutationProb) + mutDist(1)*2*(rand()-0.5);
        tempMutateY(r(2,:) < mutationProb) = tempMutateY(r(2,:)...
            < mutationProb) + mutDist(2)*2*(rand()-0.5);
        tempMutateZ(r(3,:) < mutationProb) = tempMutateZ(r(3,:)...
            < mutationProb) + mutDist(3)*2*(rand()-0.5);
        tempMutateT(r(4,:) < mutationProb) = tempMutateT(r(4,:)...
            < mutationProb) + mutDist(4)*2*(rand()-0.5);

        % Asssign
        mutate{i}.x = tempMutateX;
        mutate{i}.y = tempMutateY;
        mutate{i}.z = tempMutateZ;
        mutate{i}.t = tempMutateT;

        % Fix time vector (Increasing and start in 0)
        mutate{i}.t = sort(mutate{i}.t);
        mutate{i}.t = mutate{i}.t - mutate{i}.t(1);

        % Clamp the edge positions
        mutate{i}.x(1)   = node0(1);
        mutate{i}.y(1)   = node0(2);
        mutate{i}.z(1)   = node0(3);
        mutate{i}.x(end) = nodeF(1);
        mutate{i}.y(end) = nodeF(2);
        mutate{i}.z(end) = nodeF(3);
    end

    % Calculate associated variables
    parfor i = 1:nMutate
        mutate{i} =...
            setPathFields(mutate{i}, forb, xCellLog, tVec, tStart, restr);
    end

    % Add inmigrants (Random solutions)
    inmigrants = generateRandPop(nInmigrants,nNodes,intervals,node0,nodeF);
    parfor i = 1:nInmigrants
        inmigrants{i} =...
            setPathFields(inmigrants{i},forb,xCellLog,tVec,tStart,restr);
    end

    % Recombination
    pop = [elite;xOver;mutate;inmigrants];

end
OptimizationTime = toc;

%=========================================================================
% SHOW SOLUTION
%=========================================================================

fprintf('\n \toptim = %f',optim)
fprintf('\n \tnForb = %f',nForb)
fprintf('\n \tparts:')
for i = 1:length(elemSol.parts)
    fprintf('\n\t\t%i: %f',i,elemSol.parts(i))
end
```

```
%=========================================================================
% FUNCTIONS (alphabetical order)
%=========================================================================

function [in] = concentrInPath(elem, xCellLog, tVec, tStart)

    % Detection ellipsoid semiaxis
    radius = [5,5,1];

    cellN = size(xCellLog,2);

    % Sample point loop
    in = zeros(cellN,1);
    for n = 1:length(elem.tt)

        t = elem.tt(n);

        % Get nearest time from simulation solution (different time vector)
        [~, nearInd] = min(abs(tVec - t/3600 - tStart));
        sample = reshape(xCellLog(nearInd, :, :),size(xCellLog,2),3);

        % Particles inside detection ellipsoid
        inVol = ((sample(:,1)-elem.xx(n)).^2./radius(1).^2 ...
            + (sample(:,2)-elem.yy(n)).^2./radius(2).^2 ...
            + (sample(:,3)-elem.zz(n)).^2./radius(3).^2-1) < 0;
        in = in|inVol;

    end

end

function [pop] = generateRandPop(nPop, nNodes, inter, node0, nodeF)

    % Initialize population's cell vector
    pop = cell(nPop,1);

    % Create each member of population
    for i = 1:nPop
        pop{i}.x = rand(1,nNodes)*(inter(1,2) - inter(1,1)) + inter(1,1);
        pop{i}.y = rand(1,nNodes)*(inter(2,2) - inter(2,1)) + inter(2,1);
        pop{i}.z = rand(1,nNodes)*(inter(3,2) - inter(3,1)) + inter(3,1);
        pop{i}.t = sort(rand(1,nNodes)*(inter(4,2)-inter(4,1))...
            + inter(4,1));
        pop{i}.t = pop{i}.t - pop{i}.t(1);
    end

    % Clamp start and end points to node0 and nodeF
    for i = 1:nPop
        pop{i}.x(1)   = node0(1);
        pop{i}.y(1)   = node0(2);
        pop{i}.z(1)   = node0(3);
        pop{i}.x(end) = nodeF(1);
        pop{i}.y(end) = nodeF(2);
        pop{i}.z(end) = nodeF(3);
    end
end
```

```
function [inters] = intersForbidden(elem, forb)

    % Intersection function (Intersection at g(t) = 0)
    g = @(t) forb( ppval(t, elem.pp.x), ppval(t, elem.pp.y),...
        ppval(t, elem.pp.z));

    % Apply selective sampling
    inters = selectiveSampling(g,elem.t(1), elem.t(end), 20, 5,1e-2);

end

function len = lenInter(elem,inters)

    % Norm of the derivative function
    derNorm = @(t) sqrt( (ppval(t,elem.dpp.x)).^2 ...
        + (ppval(t,elem.dpp.y)).^2 + (ppval(t,elem.dpp.z)).^2 );

    % Check there is an even amount of intersections
    % (initial and final waypoints fixed in safe zone)
    if mod(length(inters),2) ~= 0
        warning('Missing intersections');
        len = 20*ones(20,1);
        return
    end

    % Initialize solution vector
    len = zeros(length(inters)/2, 1);

    % Lengths between odd and even intersections
    for i = 1:length(inters)/2
        len(i) = integral(derNorm, inters(2*i-1), inters(2*i));
    end

end

function [val, parts] = objFun(elem, maxVar)

    % Restrictions
    maxV = maxVar(1);
    maxA = maxVar(2);
    maxT = maxVar(3);

    % Coefficeinets Objective Function
    coefTime    = 1;      % Total time
    coefLength  = 1;      % Total length
    coefConcen  = 1e3;    % Particle count
    coefForbLen = 1;      % Length in forbidden zone
    coefVel     = 1;      % Max velocity
    coefAcel    = 1;      % Max acceleration
    coefPenal   = 1e20;   % Penalization

    % Objective function
    parts(1) =  coefTime*(elem.T);
    parts(2) =  coefLength*elem.len;
    parts(3) = -coefConcen*sum(elem.rho);
    parts(4) =  coefPenal*(coefForbLen*sum(elem.lenInters));
    parts(5) =  coefPenal*(coefVel *(elem.maxV-maxV > 0)*(elem.maxV-maxV));
    parts(6) =  coefPenal*(coefAcel*(elem.maxA-maxA > 0)*(elem.maxA-maxA));
    parts(7) =  coefPenal*(coefTime*(elem.T-maxT > 0)*(elem.T - maxT));
```

```
    % Objective value
    val = sum(parts);

end

function [dpp] = ppDer(pp)

    % Polynomial derivative operator matrix
    D = diag(4:-1:1,1);

    % Spline derivatives
    dpp = pp;
    dpp.x.coefs = pp.x.coefs*D;
    dpp.y.coefs = pp.y.coefs*D;
    dpp.z.coefs = pp.z.coefs*D;

end

function [elem] = refineNodes(elem, forb, xCellLog, tVec, tStart,restr)

    % Add one random node in the range of the time vector
    t = sort([elem.t, rand()*(elem.t(end)-elem.t(1)) + elem.t(1)]);

    % Reset spline
    elem.x = ppval(elem.pp.x, t);
    elem.y = ppval(elem.pp.y, t);
    elem.z = ppval(elem.pp.z, t);
    elem.t = t;
    elem = setPathFields(elem, forb, xCellLog, tVec, tStart,restr);
end

function [tt] = selectiveSampling(fun, xL, xH, divd, sampTol, tol)

    % Default parameters
    if nargin < 6
        tol = 1e-6;
    end
    if nargin < 5
        sampTol = (xH - xL)/20;
    end
    if nargin < 4
        divd = 10;
    end

    % Create sampling vector
    tt = xL:sampTol*divd:xH-sampTol;
    dx=sampTol;

    % Solution loop
    while dx > tol

        % Sub-partitions
        prog = 0:dx:divd*dx;

        % New sampling vector
        ttNew = [];
        for i = 1:length(tt)
            ttNew = [ttNew, tt(i) + prog];
        end
        tt = ttNew(ttNew <= xH);
```

```
        % Find solution intervals
        gt = fun(tt);
        cs = gt(1:end-1).*gt(2:end);
        tt = tt(cs <= 0);
        dx = dx/divd;
    end

end

function [elem] = setPathFields(elem, forb, xCellLog, tVec, tStart, restr)

    % Splines
    elem.pp = spline434_3D(elem);   % Position
    elem.dpp = ppDer(elem.pp);      % Velocity
    elem.ddpp = ppDer(elem.dpp);    % Acceletation

    % Total length of path
    elem.len = lenInter(elem, [elem.t(1) elem.t(end)]);

    % Total duration of path
    elem.T = elem.t(end) - elem.t(1);

    % Test point vectors
    elem.tt = linspace(elem.t(1),elem.t(end),min(round(elem.len*2),5000));
    elem.xx = ppval(elem.pp.x,elem.tt);
    elem.yy = ppval(elem.pp.y,elem.tt);
    elem.zz = ppval(elem.pp.z,elem.tt);

    % Itersections with forbidden zone
    elem.inters = intersForbidden(elem, forb);    % Time points
    elem.lenInters = lenInter(elem, elem.inters); % Length of intersections

    % Max velocity and acceleration
    elem.maxV = max( sqrt( ppval(elem.dpp.x,elem.tt).^2 ...
        + ppval(elem.dpp.y,elem.tt).^2 + ppval(elem.dpp.z,elem.tt).^2 ));
    elem.maxA = max( sqrt( ppval(elem.ddpp.x,elem.tt).^2 ...
        + ppval(elem.ddpp.y,elem.tt).^2 + ppval(elem.ddpp.z,elem.tt).^2 ));

    % Seen particles count (only of restrictions are met to save time)
    if isempty(elem.inters) && elem.maxV < restr(1) &&...
            elem.maxA < restr(2) && elem.T < restr(3)
        elem.rho = concentrInPath(elem, xCellLog, tVec, tStart);
    else
        elem.rho = zeros(size(xCellLog,2),1);
    end
end

function [pp] = spline434_1D(t,x, v0, a0, vN, aN)

    if length(x)~=length(t)
        error('t and x not of the same size')
    end

    N = length(t);
    dim = 4*N-2;

    A = zeros(dim);
```

```
    % FORM RHS
    b = zeros(dim, 1);
    for i = 1:N-1
        b(4*i-3) = x(i);
        b(4*i) = x(i+1);
    end
    b(2) = v0; b(3) = a0; b(dim-1) = vN; b(dim) = aN;

    h(1:N-1) = t(2:N) - t(1:N-1);

    % FORM MATRIX

    % First block
    A(1:3,3:5) = flip(diag([1,1,2]))';

    % First block iterative part
    A(4:7,1) = [h(1)^4; 0; 4*h(1)^3; 12*h(1)^2];

    % Iterative block
    for i = 1:N-2
        m = 4*i;
        A(m+1:m+3, m+3:m+5) = flip(diag([1,-1,-2]))';
        A(m:m+3,m-2:m+1) = [
              h(i)^3     h(i)^2   h(i)    1 ;
              0          0        0       0 ;
            3*h(i)^2   2*h(i)     1       0 ;
            6*h(i)       2        0       0 ];
    end

    % Last block iterative part
    A(dim-5:dim-3, dim-3:dim) = A(dim-5:dim-3, dim-4:dim-1);

    % Last block
    A(dim-2:dim, dim-4:dim) = [
           h(N-1)^4     h(N-1)^3     h(N-1)^2   h(N-1)    1 ;
         4*h(N-1)^3   3*h(N-1)^2   2*h(N-1)     1         0 ;
        12*h(N-1)^2   6*h(N-1)       2          0         0 ];

    % Solve system
    coefsV = sparse(A) \ b;

    % Form piecewise polynomial object
    coefs(1,:) = coefsV(1:5)';
    for i = 1:N-2
        coefs(i+1,:) = [0 coefsV(4*i+2:4*i+5)'];
    end
    coefs(N-1,:) = coefsV(dim-4:dim);

    pp = mkpp(t,coefs);

end

function [pp] = spline434_3D(elem)

    pp.x = spline434_1D(elem.t,elem.x, 0,0,0,0);
    pp.y = spline434_1D(elem.t,elem.y, 0,0,0,0);
    pp.z = spline434_1D(elem.t,elem.z, 0,0,0,0);

end
```

Bibliografía

[1] United Nations, «Resolution 64/292: The human right to water and sanitation,» de *General Assembly*, 2010. Available: https://www.un.org/ga/search/view_doc.asp?symbol=A/RES/64/292&Lang=E.

[2] UNESCO, The United Nations World Water Development Report 2019: Leaving No One Behind, París, France: United Nations Educational, Scientific and Cultural Organization, 2019. ISBN: 978-92-3-100309-7.

[3] UNESCO, The United Nations world water development report 2015: Water for a Sustainable World, Paris, France: United Nations Educational, Scientific and Cultural Organization, 2015. ISBN: 978-92-3-100071-3.

[4] L. L. Canto de Sá, J. M. dos Santos Vieira, R. de Alcântara Mendes, S. C. Campelo Pinheiro, E. Rodrigues Vale, F. A. dos Santos Alves, I. Maura de Jesus, E. C. de Oliveira Santos y V. Bandeira da Costa, «Occurrence of toxic cyanobacterial bloom in the left margin of the Tapajós river, in the Municipality of Santarém (Pará State, Brazil),» *Rev Pan-Amaz Saude*, vol. 1, nº 1, pp. 159-166, 2010. doi: 10.5123/S2176-62232010000100022.

[5] C. Fischer, *Bloom of cyanobacteria in a freshwater pond*, North-eastern Lower Saxony, Germany, 2014.

[6] E. A. Cantoral Uriza, A. D. Asencio Martínez y M. Aboal Sanjurjo, «Cianotoxinas: efectos ambientales y sanitarios. Medidas de prevención,» *Hidrobiológica*, vol. 27, nº 2, pp. 241-251, 2017. ISSN: 0188-8897.Available: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0188-88972017000200241.

[7] I. Chorus y J. Bartram, Toxic cyanobacteria in water : a guide to their public health consequences, monitoring and management, World Health Organization, 1999. ISBN: 0419239308.

[8] SAICA, «Red de alerta (SAICA),» [En línea]. Available: https://www.miteco.gob.es/es/agua/temas/estado-y-calidad-de-las-aguas/aguas-superficiales/programas-seguimiento/saica.aspx. [Último acceso: 31 Agosto 2020].

[9] J. Morón López, L. Nieto Reyes y R. El-Shehawy, «CIANOALERT: Alerta inteligente contra las floraciones nocivas de cianobacterias,» IMDEA Agua, 2017. [En línea]. Available: https://www.agua.imdea.org/noticias/2017/cianoalert-alerta-inteligente-contra-las-floraciones-nocivas-de-cianobacterias. [Último acceso: 01 Septiembre 2020].

[10] J. Germano, «After outbreaks, Rochester's on guard against blue-green algae,» Record Searchlight, 2019. [En línea]. Available: https://eu.redding.com/story/news/2017/09/01/after-outbreaks-rochesters-guard-against-blue-green-algae/597680001/. [Último acceso: 01 Septiembre 2020].

[11] C. Yan, «Government Department Utilizes USV to Monitor Water Quality,» OceanAlpha, 2019. [En línea]. Available: https://www.oceanalpha.com/news_list/government-department-utilizes-usv-to-monitor-water-quality/. [Último acceso: 01 Septiembre 2020].

[12] Blender Online Community, «Blender - a 3D modelling and rendering package,» Blender Foundation, [En línea]. Available: http://www.blender.org. [Último acceso: 04 Agosto 2020].

[13] COMSOL, «COMSOL Multiphysics,» [En línea]. Available: https://www.comsol.com. [Último acceso: 07 Agosto 2020].

[14] MathWorks, Inc., «MATLAB version R2018b (9.5.0.944444),» [En línea]. Available: https://www.mathworks.com/products/matlab.html. [Último acceso: 07 Agosto 2020].

[15] MathWorks, Inc., «Partial Differential Equation Toolbox version 3.1,» [En línea]. Available: https://www.mathworks.com/products/pde.html. [Último acceso: 07 Agosto 2020].

[16] S. Holcombe, «File Exchange: inpolyhedron - are points inside a triangulated volume? version 3.3,» 2016. [En línea]. Available: https://www.mathworks.com/matlabcentral/fileexchange/37856-inpolyhedron-are-points-inside-a-triangulated-volume. [Último acceso: 07 Agosto 2020].

[17] MIT, «Basics of turbulent flow,» 2006. [En línea]. Available: http://www.mit.edu/course/1/1.061/OldFiles/www/dream/SEVEN/SEVENTHEORY.PDF. [Último acceso: 09 Agosto 2020].

[18] E. A. Medrano, Physical Aspects Explaining Cyanobacteria Scum Formation in Natural Systems, Eindhoven, 2014. [PhD Thesis].

[19] M. R. Maxey y J. J. Riley, «Equation of motion for a small rigid sphere in a nonuniform flow,» *The Physics of Fluids*, vol. 26, nº 4, pp. 883-889, 1983. doi: 10.1063/1.864230.

[20] J. L. Guermond, P. Minev y J. Shen, «An overview of projection methods for incompressible flows,» *Computer Methods in Applied Mechanics and Engineering*, vol. 195, pp. 6011-6045, 2006. doi: 10.1016/j.cma.2005.10.010.

[21] W. Ee y J.-g. Liu, «Projection Method I: Convergence and Numerical Boundary Layers,» *SIAM Journal on Numerical Analysis*, vol. 32, nº 4, pp. 1017-1057, 1995. doi: 10.1137/0732047.

[22] W. Ee y J.-g. Liu, «Projection Method II: Godunov–Ryabenki Analysis,» *SIAM Journal on Numerical Analysis*, vol. 33, nº 17, pp. 1597-1621, 1996. doi: 10.1137/S003614299426450X.

[23] W. Ee y J.-g. Liu, «Projection Method III: Spatial discretization on the Staggered Grid,» *Mathematics of Computation*, vol. 71, nº 237, pp. 27-47, 2002. doi: 10.1090/S0025-5718-01-01313-8.

[24] A. J. Chorin, «The numerical solution of the Navier-Stokes equations for an incompressible fluid,» *Bulletin of the American Mathematical Society*, vol. 73, nº 6, pp. 928-931, 1967. Available: https://projecteuclid.org/euclid.bams/1183529112.

[25] A. J. Chorin, «Numerical solution of the Navier-Stokes equations,» *Mathematics of Computation*, vol. 22, pp. 745-762, 1968. doi: 10.1090/s0025-5718-1968-0242392-2.

[26] B. Delaunay, «Sur la sphère vide. A la mémoire de Georges Voronoï,» *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, nº 6, pp. 793-800, 1934. Available: http://mi.mathnet.ru/eng/izv4937.

[27] F. Moukalled, L. Mangani y M. Darwish, The Finite Volume Method in Computational Fluid Dynamics, Cham: Springer, 2016. ISBN: 78-3-319-16874-6.

[28] J. H. Ferziger y M. Peric, «Finite Difference Methods,» de *Computational Methods for Fluid Dynamics*, Berlín, Springer, 2002, pp. 39-69. ISBN: 3-540-42074-6.

[29] O. Rübenkönig, The Finite Difference Method (FDM) - An introduction, 2006.

[30] A. Tourin, «Introduction to Finite Differences Methods,» de *Optimal Stochastic Control, Stochastic Target Problems, and Backward SDE*, vol. 29, New York, NY: Springer, 2013. ISBN: 978-1-4614-4286-8.doi: 10.1007/978-1-4614-4286-8_13.

[31] T. A. Davis, Direct Methods for Sparse Linear Systems, Gainesville, FL: SIAM, 2006. ISBN: 978-0-89871-613-9.

[32] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd Ed., Philadelphia, PA: Society for Industrial and Applied Mathematics, 2000. ISBN: 978-0-89871-534-7.

[33] J. R. Bunch y J. E. Hopcroft, «Triangular factorization and inversion by fast matrix multiplication,» *Mathematics of Computation*, vol. 28, nº 125, pp. 231-236, 1974. doi: 10.2307/2005828.

[34] R. Rannacher, «Finite Element Methods for the Incompressible Navier-Stokes Equations,» de *Fundamental Directions in Mathematical Fluid Mechanics*, Birkhäuser, Basel, 2000, pp. 191-293. ISBN: 978-3-0348-8424-2.doi: 10.1007/978-3-0348-8424-2_6.

[35] B.-N. Jiang, T.-L. Lin, L.-J. Hou y L. A. Povinelli, «A least-squares finite element method for 3D incompressible Navier-Stokes equations,» *NASA Technical Reports Server*, 1993. doi: 10.2514/6.1993-338.

[36] F. Glaisner y T. E. Tezduyar, «Finite element techniques for the Navier-Stokes equations in the primitive variable formulation and the vorticity stream-function formulation,» *NASA Technical Reports Server*, 1987.

[37] The OpenFOAM Foundation, «OpenFOAM,» [En línea]. Available: https://openfoam.org. [Último acceso: 07 Agosto 2020].

[38] Gerris, «Gerris Flow Solver,» [En línea]. Available: http://gfs.sourceforge.net. [Último acceso: 07 Agosto 2020].

[39] ANSYS, «ANSYS Fluent,» [En línea]. Available: https://www.ansys.com/products/fluids/ansys-fluent. [Último acceso: 07 Agosto 2020].

[40] Autodesk, Inc., «Autodesk CFD,» [En línea]. Available: https://www.autodesk.com/products/cfd. [Último acceso: 07 Agosto 2020].

[41] J. Koko, «Efficient MATLAB codes for the 2D/3D Stokes equation with the mini-element,» *HAL*, 2019. Available: https://hal.archives-ouvertes.fr/hal-02047515.

[42] J. Burkardt, «navier_stokes_3d_exact,» 2020. [En línea]. Available: https://people.sc.fsu.edu/~jburkardt/m_src/navier_stokes_3d_exact/navier_stokes_3d_exact.html. [Último acceso: 2020 Septiembre 06].

[43] H. V. Thurman, Introductory Oceanography, New Jersey: Prentice Hall College, 1997. ISBN: 0-13-262072-3.

[44] C. C. Mae Overman, Modeling Vertical Migration of Cyanobacteria and Zooplankton, Portland, USA: Portland State University. Department of Civil & Environmental Engineering, 2019. [MSc Thesis].doi: 10.15760/etd.7054.

[45] S. Perez-Carabaza, E. Besada-Portas, J. A. Lopez-Orozco y J. M. de la Cruz, «A Real World Multi-UAV Evolutionary Planner for Minimum Time Target Detection,» *GECCO '16: Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 981-988, 2016. doi: 10.1145/2908812.2908876.

[46] E. Besada-Portas, J. L. Risco-Martín y J. A. López-Orozco, «Análisis y planificación de misiones de búsqueda y rescate en el entorno marítimo,» *XL Jornadas de Automática: libro de actas*, pp. 8-15, 2019. doi: 10.17979/spudc.9788497497169.008.

[47] M. Caro-Huertas, Diseño y Simulación de un Sistema para el Control de Contaminantes en Embalses, 2019. [Trabajo de Fin de Máster].

[48] D. Sunday, «Point in Polygon Inclusion,» 2001. [En línea]. Available: http://geomalgorithms.com/a03-_inclusion.html. [Último acceso: 20 Agosto 2020].

[49] K. Hormann y A. Agathos, «The point in polygon problem for arbitrary polygons,» *Computational Geometry*, vol. 20, nº 3, pp. 131-144, 2001. doi: 10.1016/S0925-7721(01)00012-8.

[50] M. Shimrat, «Algorithm 112: Position of point relative to polygon,» *Communications of the ACM*, vol. 5, nº 8, p. 434, 1962. doi: 10.1145/368637.368653.

[51] D. Horvat, «Ray-casting point-in-polyhedron test,» de *Proceedings of CESCG 2012: The 16th Central European Seminar on Computer Graphics (non-peer-reviewed)*, 2012.