UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Técnica Superior de Ingenieros Navales

MADRID

TRABAJO FIN DE MÁSTER EN
INGENIERÍA NAVAL Y OCEÁNICA

Nº TFM-377

Análisis tecno-económico de un granelero propulsado a vela de 100 m de eslora para el transporte en la industria con principios lean

Autor:
JAVIER SARMIENTO GIL

Tutor:
RAFAEL D'AMORE DOMENECH

SEPTIEMBRE DE 2025

Dedicado a todos aquellos que me han ayudado y han estado a mi lado a lo largo de mi carrera universitaria

Propuesta TFM

POLITÉCNICA "Ingeniamos el futuro" CAMPUS DE EXCELENCIA INTERNACIONAL

PROPUESTA DE TRABAJO FIN DE MÁSTER

UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS NAVALES
Avda. de la Memoria, 4. 28040 Madrid

D./Dª Rafael d'Amore Domenech profesor/a adscrito/a al departamento DACSON de la E.T.S. de Ingenieros Navales de la U.P.M. propone el siguiente TRABAJO FIN DE MÁSTER para alumnos/as del título de MuINyO, para su evaluación por la C.O.A. y posterior aprobación, si procede.

Cotutor/a D./Dª | Empresa

Correo electrónico profesional Cotutor/a

Avalista D./Dª (en caso de ser necesario)

Título: Análisis tecno-económico de un granelero propulsado a vela de 100 m de eslora para el transporte en la industria con principios lean

Descripción del TRABAJO:

(Indíquese seguidamente si esta información se encuentra en documento anexo claramente identificado)

Objetivo:

El objetivo general de este trabajo es estudiar la viabilidad económica de un buque granelero monocasco disponible en la librería ShipSim de Open Modelica de 100 m de eslora entre perpendiculares, 20 m de manga, 4 metros de calado y un coeficiente de bloque de 0.693, únicamente propulsado a vela para el transporte de carbón entre Estados Unidos y Europa en la industria con principios lean. Para ello se definen los siguientes objetivos específicos:
- Diseñar la disposición general a partir de las formas del buque disponible en la librería de ShipSim.
-Calcular la capacidad de carga mediante composición de volúmenes a partir de la DG obtenida del buque.
-Obtener los diagramas polares del movimiento del buque en función de la dirección del viento y su velocidad mediante modelización acausal.
-Definir un algoritmo de weather routing para calcular el tiempo de viaje de la ruta óptima en base a datos históricos de vientos.
-Realizar un análisis estadístico de los tiempos de viaje obtenidos a partir de datos históricos de viento con el algoritmo de wheather routing para estimar la duración promedio y la variabilidad del viaje.
-Discutir la viabilidad económica del transporte de carga a granel mediante el uso de graneleros propulsados únicamente mediante velas rígidas considerando el capital inmovilizado durante el tiempo de viaje + la variabilidad mediante el uso del Valor Actualizado Neto.

Método y fases del trabajo:

1º Busqueda de información, selección de ruta y valoración de diferentes herramientas para poder alcanzar el objetivo.

2º Diseño de la disposición general a partir de las formas del buque y calculo de la capacidad de carga mediante composición de volúmenes a partir de la DG obtenida

3º Planteamiento de embarcación con velas rígidas mediante sistemas de modelización acausal como OpenModelica y más específicamente la librería SHIP-Sim para obtener los datos necesarios que permitan generar los diagramas polares del mismo y estimar su capacidad de carga.

4º Programación de un algoritmo de wheather routing en lenguage python que permita calcular el tiempo de viaje en función de los diagramas polares y datos reales de viento del globo terraqueo, a partir de la base de datos de la NASA, MERRA-2.

5º Análisis estadístico del viaje en función de la fecha y hora de salida y calculo de la desviación.

6º Discutir la viabilidad económica del proyecto teniendo en cuenta la variabilidad del tiempo de viaje en el tiempo de capital inmobilizado, mediante el "Valor Actualizado Neto" y verificación de la opción óptima (vela o propulsión convencional) para realizar este viaje haciendo uso de hojas de cálculo en función de la cantidad anual a transportar de carbón.

POLITÉCNICA "Ingeniamos el futuro" | CAMPUS DE EXCELENCIA INTERNACIONAL

PROPUESTA DE TRABAJO FIN DE MÁSTER

UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS NAVALES
Avda. de la Memoria, 4. 28040 Madrid

Medios específicos necesarios:

Programa de simulación y modelización acausal de lenguaje abierto OpenModelica
Editor de código como VScode
Anaconda para generar entornos conda y usar las librerias convenientes
Lenguaje de programación python
Base de datos MERRA-2 para obtener datos de viento reales.
Rhinoceros
Autocad
Hojas de cálculo excel

Estructura de capítulos, planos o documentos que constituyen el cuerpo del trabajo:

Introducción
Estudio y selección de la ruta
Disposición general y capacidad de carga del buque
Obtención de diagramas polares
-Fundamentos
-OpenModelica
-Aspectos técnicos
-Simulaciones, procesamiento de datos y diagrama polar
Algoritmo de wheather routing
-Merra-2, datos de viento reales
-Revisión bibliográfica
-Flujograma del algoritmo
-Obtención de tiempos de viaje y estudio estadístico
Análisis económico
Conclusiones

☐ **se solicita confidencialidad del TFM, cuya justificación es la siguiente:**

Propuesta de Tribunal, si lo considera el Tutor/a:

Presidente:
Vocal:
Secretario:

Asignación propuesta (táchese lo que no proceda):

1. Para el/la alumno/a D./Dª Javier Sarmiento Gil
2. Libre

Madrid, a 17 de JUNIO de 2025

Firmado:

Tutor/a	Cotutor/a	Alumno/a	Avalista
D'AMORE DOMENECH RAFAEL FEDERICO - 53733433M Firmado digitalmente por D'AMORE DOMENECH RAFAEL FEDERICO - 53733433M Fecha: 2025.06.17 10:51:13 +02'00'		SARMIENTO GIL, JAVIER (FIRMA) Firmado digitalmente por SARMIENTO GIL, JAVIER (FIRMA) Fecha: 2025.06.17 10:42:09 +02'00'	

2

Resumen

El presente trabajo realiza un estudio de viabilidad tecno-económica para un buque granelero de 100 metros de eslora, cuyo modelo se ha obtenido de la librería ShipSim (Basilio Puente & M. Dolores Fernandez, 2021) para OpenModelica, propulsado íntegramente por velas rígidas y comparándolo con una alternativa de propulsión convencional. El objetivo es determinar las condiciones bajo las cuales la propulsión eólica moderna puede ser una solución económicamente rentable.

Para ello, el trabajo se centra en el transporte de una carga determinada a lo largo de una ruta definida de acuerdo con diferentes criterios La metodología se ha estructurado en tres fases principales. En primer lugar, se ha modelado el rendimiento del buque a vela en la mar frente a diferentes condiciones de viento mediante el software OpenModelica y la librería ShipSim, obteniendo como resultado sus diagramas polares de rendimiento.

A continuación, con los diagramas polares como dato de entrada fundamental, se ha desarrollado un algoritmo de *weather routing* a medida. Esta herramienta se nutre de los datos de rendimiento del buque y de datos históricos de viento para simular miles de viajes transoceánicos. A pesar del elevado coste computacional que implica, este proceso ha permitido generar un conjunto de datos estadísticamente robusto de los tiempos de viaje y, lo que es más importante, de su variabilidad.

Finalmente, estos resultados técnicos se han integrado en un modelo económico probabilístico basado en una Simulación de Montecarlo. Este modelo compara el rendimiento de flotas de ambos tipos de buque y calcula su viabilidad en miles de escenarios futuros, considerando la incertidumbre en variables clave como el precio del combustible o la tasa de descuento.

El principal resultado del análisis indica que, bajo las condiciones de mercado actuales, la inversión en el buque de propulsión a vela no empieza a ser rentable hasta alcanzar un horizonte temporal de 35 años. La viabilidad del proyecto es altamente sensible al precio del combustible y a los costes de las emisiones de CO_2. Se concluye que, si bien la tecnología es técnicamente viable, su competitividad económica queda condicionada a una inversión a muy largo plazo o a un entorno regulatorio y de mercado más favorable.

Abstract

This thesis presents a techno-economic feasibility study for a 100-meter-long bulk carrier, whose model has been obtained from the ShipSim library (Basilio Puente & M. Dolores Fernandez, 2021) for OpenModelica, propelled entirely by rigid sails and comparing it with a conventional propulsion alternative. The objective is to determine the conditions under which modern wind propulsion can be an economically viable solution.

To this end, the work focuses on the transport of a specific cargo along a defined route according to different criteria. The methodology is structured in three main phases. Firstly, the performance of the sailing vessel at sea has been modeled under different wind conditions using the OpenModelica software and the ShipSim library, resulting in its performance polar diagrams.

Subsequently, with the polar diagrams as a fundamental input, a custom weather routing algorithm has been developed. This tool utilizes the vessel's performance data and historical wind data to simulate thousands of transoceanic voyages. Despite the high computational cost involved, this process has allowed for the generation of a statistically robust dataset of voyage times and, most importantly, their variability.

Finally, these technical results have been integrated into a probabilistic economic model based on a Monte Carlo Simulation. This model compares the performance of fleets of both types of vessels and calculates their viability across thousands of future scenarios, considering the uncertainty in key variables such as fuel price or the discount rate.

The main result of the analysis indicates that, under current market conditions, the investment in the sail-propelled vessel does not start to become profitable until a 35-year time horizon is reached. The project's viability is highly sensitive to fuel prices and the costs of CO_2 emissions. It is concluded that, while the technology is technically feasible, its economic competitiveness is conditioned on a very long-term investment or a more favorable regulatory and market environment

Agradecimientos

This work was founded by the EU through the Horizon Europe Project POSEIDON (Grant Agreement ID: 101096457), funded by the European Union under the Horizon Europe framework, within the call for proposals HORIZON-CL5-2022-D5-01 (Innovative energy storage systems on-board vessels).

En este trabajo quiero expresar mi agradecimiento a la Escuela Técnica Superior de Ingenieros Navales (ETSIN) de la Universidad Politécnica de Madrid por su formación, apoyo y ayuda de parte de sus profesores y personal auxiliar a lo largo de toda mi etapa universitaria.

Quiero agradecer especialmente a mi tutor, Rafael D'Amore Domenech, que además de haber sido un gran profesor y tutor, me ha enseñado a pensar como un ingeniero y a tomar decisiones en base a datos y no a suposiciones, sin su ayuda y su confianza (muchas veces mayor que la mía) en mis habilidades, no habría conseguido terminar este trabajo. Gracias.

Por supuesto, dar las gracias a mi pareja, amigos, familia y compañeros de la escuela que han estado a mi lado a lo largo de este extenso y muchas veces tedioso trabajo, sin su ayuda y apoyo no hubiese sido lo mismo.

Índice

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

El transporte marítimo de mercancías dependió de la vela durante mucho tiempo. Con la llegada de los motores de vapor y combustión, esto cambió. Los motores ofrecían algo que la vela no podía, la capacidad de cumplir con un horario sin depender del tiempo que hiciera.

Hoy en día, la situación ha cambiado. El sector naval se enfrenta a una presión regulatoria y económica muy elevada. Organismos como la Organización Marítima Internacional (OMI) y la Unión Europea han establecido hojas de ruta para la descarbonización, con objetivos de reducción de emisiones que obligan a buscar alternativas en el modo en que se propulsan los buques. A esto se suma la volatilidad de los precios del combustible, que representa una parte muy importante de los costes de operación. Si bien se investigan combustibles alternativos como el amoníaco o el metanol, estos presentan sus propios desafíos en cuanto a producción, almacenamiento y coste. La idea es aprovechar una fuente de energía gratuita, inagotable y sin emisiones como el viento

Para volver a aprovechar el viento, la industria está desarrollando diferentes tecnologías, conocidas como WAPS (Sistemas de Propulsión Asistida por Viento). Existen varias opciones principales: los rotores Flettner, que son cilindros giratorios que aprovechan el efecto Magnus para generar empuje; las velas de succión, que usan ventiladores para mejorar la aerodinámica y la sustentación; o las cometas de gran tamaño que tiran del buque desde una gran altura. (Laursen, 2023)

Este proyecto se centra en el uso de velas rígidas (*hard sails*). Este tipo de vela funciona de manera similar al ala de un avión. Se trata de una estructura rígida con un perfil aerodinámico, fabricada con materiales compuestos, que se instala verticalmente sobre la cubierta. Un sistema automático se encarga de orientarla en cada momento para obtener el máximo empuje del viento. Se han elegido para este estudio por su alta eficiencia aerodinámica y su robustez estructural, características importantes para un buque que depende completamente de ellas para su propulsión.

Sin embargo, volver a depender del viento, aunque sea con tecnología moderna, presenta desafíos importantes. El principal problema es la variabilidad del viento, no es constante, lo que se traduce en una incertidumbre sobre la duración de los viajes. Para la logística que depende de la puntualidad, esta falta de previsibilidad es un problema importante. Además, la inversión inicial para instalar estos sistemas es alta, y la rentabilidad depende de factores muy variables, como el precio que alcanzarán los combustibles convencionales o el coste de las tasas por emisiones de CO_2, así como las penalizaciones del FuelEU.

Es precisamente para abordar estos problemas que se realiza este trabajo. No basta con saber que una tecnología es viable técnicamente; es fundamental calcular su rendimiento en condiciones reales y comprobar si es una alternativa económicamente viable. Un análisis de viabilidad tecno-económica ofrece una base para una futura toma de decisiones.

Una característica fundamental del buque analizado en este proyecto es la ausencia de un motor de propulsión principal. Esta decisión de diseño se ampara directamente en el marco normativo de la Organización Marítima Internacional (OMI). Los convenios internacionales más importantes, como el SOLAS, eximen de gran parte de sus regulaciones a los "buques no propulsados por medios mecánicos", por lo que no existe una prohibición a nivel global que impida la operación de un buque mercante exclusivamente a vela en travesías oceánicas. Las posibles exigencias de motores auxiliares para maniobras en puerto dependen de administraciones nacionales o sociedades de clasificación, pero no son una imposición de la

OMI para la navegación. Este marco normativo es, por tanto, el que permite centrar este estudio en el diseño y la viabilidad de un buque sin propulsión mecánica convencional.

El objetivo principal de este Trabajo de Fin de Máster es, por tanto, analizar la viabilidad tecno-económica de un buque granelero de 100 metros de eslora propulsado únicamente por velas rígidas. El análisis se lleva a cabo mediante el modelado del rendimiento del buque, la simulación de rutas oceánicas con datos históricos de viento y, finalmente, una evaluación económica probabilística que permite manejar la incertidumbre de las variables clave.

El documento se organiza de la siguiente manera. Primero, se describen la ruta seleccionada y las bases del modelo económico utilizado para la comparación. A continuación, se detalla el proceso técnico seguido para el cálculo de las travesías a vela. Finalmente, se presentan y discuten los resultados del análisis, ofreciendo unas conclusiones sobre la viabilidad del modelo de buque propuesto.

Capítulo 2
Estudio y selección de la ruta

La identificación de la ruta y la mercancía transportados en este caso de estudio se seleccionará siguiendo diferentes criterios tanto técnicos como económicos. Este enfoque permite la selección de la opción de transporte más eficiente y económicamente viable para las condiciones consideradas. Los criterios de selección, así como la solución propuesta se definirán a lo largo de este capítulo.

1. Carga transportada

El primer criterio de selección en este proceso es el bien transportado. La carga debe de ser no perecedera y de un bajo valor intrínseco. No son criterios aleatorios, estos aspectos afectan en el coste debido a la variabilidad y al incremento del tiempo de viaje que implica el uso de propulsión a vela, comparado con buque de propulsión convencional. Además, en el caso de cargas perecederas, estos incrementos de tiempo podrían hacer al producto inservible, lo que descartaría lo para este tipo de transporte.

Por otro lado, el capital invertido en la carga y el tiempo que es inmovilizado impactan al coste del usuario final. La variabilidad del tiempo de tránsito asociada a los viajes de ida y vuelta del buque también penaliza económicamente la operación del usuario final de la compañía, ya que la variabilidad forzará a disponer de un inventario de seguridad mayor del mismo bien para evitar cuellos de botella e interrupciones en la cadena de suministro, debido a viajes que se retrasen significativamente. Todo este capital inmovilizado representa recursos financieros que dejan de estar disponibles para otras oportunidades de inversión que podrían generar rendimientos adicionales, un concepto conocido como coste de oportunidad. Por lo tanto, resulta esencial optimizar tanto el monto de la inversión como el tiempo de entrega para maximizar la rentabilidad global de la operación.

En la Tabla 1 se muestra los costes específicos y las densidades de diferentes bienes en los buques de carga sega a granel (Clarksons, 2020). Tras evaluar diferentes opciones se ha optado por el carbón como la carga más adecuada para el buque granelero utilizado en este estudio. El carbón cumple con los criterios clave de no ser perecedero y de poseer un valor intrínseco relativamente bajo, lo cual da una mayor probabilidad de que los costes globales se mantengan dentro de parámetros razonables y que los recursos financieros del comprador se utilicen de la manera más eficiente posible.

Tabla 1: Coste y densidad de productos secos a granel

Producto	Coste por unidad de masa (USD/kg)	Coste por unidad de volumen (USD/m³)	Densidad (kg/m³)
Mineral de hierro	0,10 – 0,15	250 - 600	2500 - 4000
Carbón	0,05 – 0,10	40 - 150	800 - 1500
Grano (trigo)	0,20 – 0,30	160 - 246	800 - 820
Fertilizantes	0,50 – 1,00	500 - 1500	1000 - 1500
Cemento	0,10 – 0,20	120 - 320	1200 - 1600
Bauxita	0,15 – 0,25	195 - 425	1300 - 1700
Productos sigerúrgicos	0,30 – 0,50	2340 - 4000	7800 - 8000
Productos forestales	0,25 – 0,40	125 - 280	500 - 700

2. Ruta marítima

Una vez seleccionada la carga, el siguiente paso es definir la zona geográfica de navegación. Esta decisión se orienta por el análisis de datos globales de viento, priorizando aquellas áreas que presentan además de una baja variabilidad en los patrones de viento, velocidades medias elevadas (d'Amore-Domenech et al., 2021), y que permitan tanto viajes de ida como el de retorno bajo condiciones favorables para el transporte de carbón. Por ello se selecciona la ruta del atlántico Norte, ya que consta de los vientos alisios durante la travesía desde Europa hacia América, y de los vientos del oeste para el trayecto de regreso desde América del Norte a Europa. Estos patrones de viento estables son preferibles frente a otras alternativas, como el Océano Pacífico, el Océano Índico o el Mar Mediterráneo, donde los vientos son más débiles o presentan una alta variabilidad.

Una vez definida de forma preliminar la zona de transporte para la carga de carbón, el siguiente paso es estudiar los principales países exportadores e importadores de este producto en el Atlántico Norte. Dado que Estados Unidos es actualmente el cuarto mayor exportador mundial de carbón y que varios países europeos figuran entre los diez principales importadores (IEA, 2024), resulta previsible que la ruta seleccionada conecte estas dos regiones, cargando en Estados Unidos y descargando en Europa.

La Tabla 2 muestra los principales puertos de exportación e importación de carbón de Estados Unidos (Parker Host, 2022) y Europa (IEA, 2025). De esta tabla se entiende que Hampton Roads, en Estados Unidos, constituye un punto clave para la exportación de carbón, mientras que Róterdam, en los Países Bajos, se consolida como uno de los principales puntos de entrada de este producto en Europa (Grootendorst et al., 2022).

Tabla 2: Principales puertos de exportación e importación de carbón del Atlántico Norte

Puerto de exportación	País	Puerto de importación	País
Hampton Roads	Estados Unidos	Róterdam	Países Bajos
Houston	Estados Unidos	Hamburgo	Alemania
Nueva Orleans	Estados Unidos	Amberes	Bélgica
Mobile	Estados Unidos	Le Havre	Francia
Baltimore	Estados Unidos	Londres	Reino Unido

Como resultado de este proceso de selección, la mercancía elegida para el caso de estudio es el carbón y la ruta seleccionada es desde el puerto de Hampton Roads, en Estados Unidos, hasta el puerto de Róterdam, en los Países Bajos. El buque aprovechará los vientos del oeste para el viaje de ida, asegurando un trayecto más rápido, y pudiendo regresar, en principio, a Hampton Roads utilizando los vientos alisios.

Capítulo 3
Obtención de diagramas polares

En este capítulo se aborda el proceso de obtención de los diagramas polares del buque objeto de estudio, los cuales representan gráficamente su velocidad y orientación en función de las condiciones de viento (velocidad y dirección). Dada la relevancia de estos diagramas en el análisis del rendimiento de la propulsión con velas, se presenta primero una revisión de los fundamentos teóricos necesarios para su comprensión. Después, se describe la implementación del modelo en OpenModelica, considerando los aspectos técnicos que afectan a la simulación. Finalmente, se detalla el procesamiento de los datos obtenidos y la generación de los diagramas polares finales.

1. Fundamentos

El estudio del comportamiento de un buque bajo distintas condiciones de viento y mar es una parte esencial dentro de este trabajo, especialmente cuando se busca evaluar tecnologías alternativas de propulsión, como es el caso de la propulsión por velas. En este contexto, los diagramas polares representan una herramienta gráfica que permite predecir la respuesta del buque frente a diferentes combinaciones de velocidad e incidencia del viento.

El concepto de diagrama polar tiene un gran uso en el campo de la aerodinámica y la navegación a vela, donde tradicionalmente se ha utilizado para describir la velocidad de embarcaciones deportivas y de recreo. No obstante, su aplicación en el ámbito del transporte marítimo comercial cobra cada vez más interés, como consecuencia de las políticas internacionales orientadas a la descarbonización del sector y la búsqueda de soluciones más eficientes desde el punto de vista energético y medioambiental.

Desde el punto de vista técnico, un diagrama polar es una representación gráfica en coordenadas polares donde el eje angular refleja el ángulo relativo del viento respecto al eje longitudinal del buque, mientras que el eje radial muestra la velocidad máxima que la embarcación puede alcanzar hacia el frente bajo esas condiciones específicas. De este modo, el diagrama permite visualizar de forma rápida y efectiva las zonas del espectro de ángulos de viento donde el buque presenta un mejor rendimiento, así como identificar aquellas situaciones donde su velocidad se ve limitada por razones aerodinámicas o hidrodinámicas. En el caso de los veleros, ángulos de viento a menos de 30º de la proa no genera velocidades hacia delante, como se puede apreciar en la siguiente imagen.

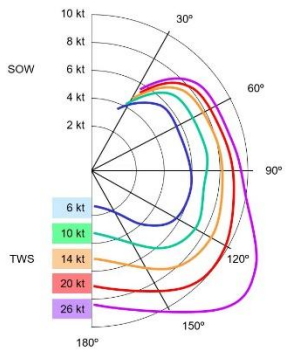

Figura 1: : Ejemplo de diagrama polar (EVOLUTION & Yatch Racing System, 2025)

La elaboración de un diagrama polar requiere previamente la obtención de un conjunto de datos representativos, los cuales se obtienen mediante simulaciones numéricas o ensayos experimentales. En el caso de los estudios numéricos, como el que aquí se presenta, se parte

de un modelo matemático que describe la dinámica del buque, incluyendo aspectos como la resistencia al avance, el empuje generado por las velas, la respuesta del timón y las fuerzas laterales inducidas por el viento. Además, se deben considerar las condiciones de estabilidad, las características geométricas del casco y las propiedades aerodinámicas de las velas.

Otro aspecto relevante en la generación de diagramas polares es la definición de los regímenes operativos del buque. Estos regímenes dependen no solo de la intensidad y dirección del viento, sino también de las decisiones tácticas del operador, como el ajuste de las velas y la orientación del timón. Por ello, en muchos casos se desarrollan campañas de simulación que abarcan un amplio rango de ángulos de vela y timón, con el objetivo de mapear completamente la respuesta del buque frente a todas las condiciones posibles.

A nivel metodológico, existen diferentes enfoques para obtener diagramas polares. Uno de ellos es el método experimental, que requiere la realización de pruebas en canales de experiencias hidrodinámicas o en condiciones de mar abierto. Estas pruebas permiten medir el rendimiento real del buque bajo condiciones controladas, aunque su elevado coste y la complejidad limitan su aplicación. La alternativa es el uso de modelos computacionales de simulación, que permiten reproducir el comportamiento del buque bajo una amplia gama de escenarios operativos de forma rápida y económica. Este es precisamente el enfoque adoptado en este trabajo, haciendo uso de la herramienta de simulación acausal OpenModelica y de la librería ShipSim (Basilio Puente & M. Dolores Fernandez, 2021).

Cabe destacar que los diagramas polares no solo aportan información valiosa para la evaluación del rendimiento del buque, sino que también constituyen una base fundamental en la optimización de rutas, especialmente en aquellos casos donde la velocidad de avance del buque depende de las condiciones meteorológicas.

En el contexto de este trabajo, los diagramas polares generados servirán como elemento clave para la posterior evaluación económica y operativa de la propulsión asistida por velas, completando la primera parte del flujograma de este trabajo (Figura 2). La información derivada de estos diagramas permitirá, entre otros aspectos, determinar el impacto de las condiciones de viento sobre los tiempos de viaje y sobre el consumo de combustible, elementos fundamentales para el cálculo de la rentabilidad de las distintas alternativas de transporte a analizar.

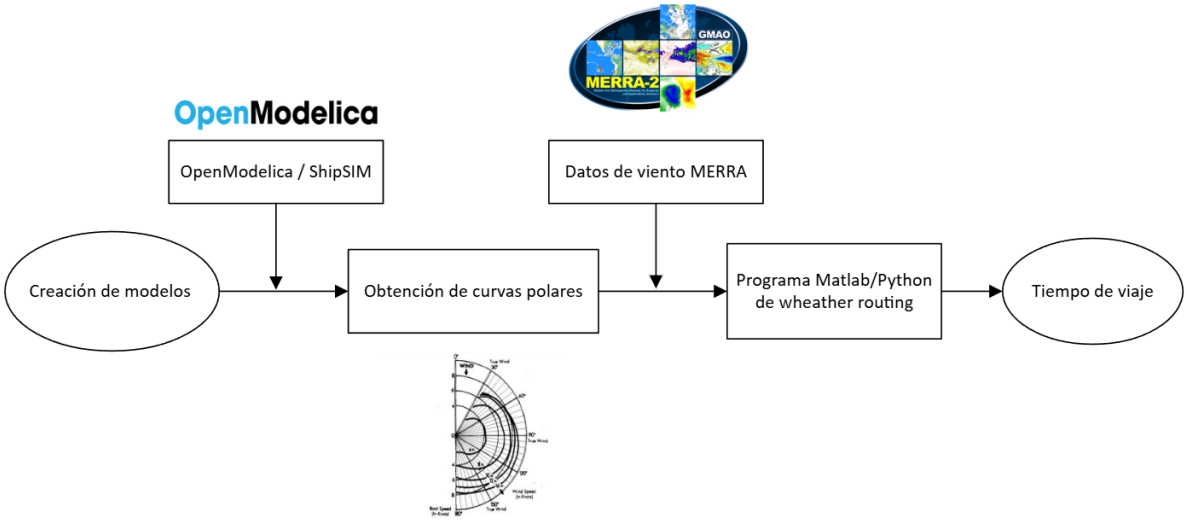

Figura 2: Flujograma de los métodos seguidos para obtener el tiempo de viaje

2. OpenModelica

En el presente trabajo, se ha hecho uso de la herramienta de simulación OpenModelica como pilar fundamental para el análisis del comportamiento dinámico de un buque propulsado por velas rígidas. Esta plataforma, basada en el lenguaje de modelado modélica, permite representar de sistemas físicos complejos, lo que la convierte en una opción adecuada para abordar las múltiples interacciones presentes en la navegación a vela. Gracias a la incorporación de la librería especializada ShipSim, ha sido posible simular la respuesta del buque frente a distintas condiciones de viento y obtener los diagramas polares que servirán como base para el análisis económico posterior. A lo largo de los siguientes apartados se detallan los motivos de esta elección, las características del entorno de simulación y el proceso seguido para obtener los resultados necesarios.

2.1. OpenModelica como herramienta de simulación

Uno de los principales motivos para seleccionar OpenModelica ha sido su naturaleza de software libre y de código abierto. Esta característica permite su uso sin coste económico y también facilita la posibilidad de adaptar o incluso desarrollar nuevas bibliotecas según las necesidades específicas del proyecto. En este caso concreto, el uso de la librería ShipSim (desarrollada por Basilio Puente y M. Dolores Fernández) ha sido clave para modelar el comportamiento del buque y obtener los diagramas polares necesarios para el análisis.

Frente a otras herramientas comerciales ampliamente utilizadas en el sector como MATLAB/Simulink o ANSYS, OpenModelica presenta una curva de aprendizaje algo más empinada en algunos aspectos, pero compensa esta limitación con una flexibilidad, transparencia y comunidad activa de desarrollo. Además, su integración con modélica, un lenguaje de modelado ampliamente reconocido permite estructurar el sistema de forma jerárquica y modular, algo muy útil cuando se trabaja con modelos complejos como los de sistemas de propulsión con viento.

En definitiva, la decisión de utilizar OpenModelica en este trabajo responde a una combinación de razones técnicas, prácticas y estratégicas, ya que permite abordar el estudio de la dinámica del buque de forma precisa y con un nivel de personalización muy alto.

2.2. ¿Qué es OpenModelica?

OpenModelica es una plataforma de simulación basada en el lenguaje de modelado modélica, desarrollado para representar sistemas físicos complejos lo más cerca posible al comportamiento real. A diferencia de las herramientas tradicionales de simulación que requieren definir explícitamente qué variable depende de cuál (es decir, causalidad impuesta), en OpenModelica los modelos se construyen de forma acausal. Esto significa que no se fuerza una dirección única en la relación entre las variables del sistema: son las propias ecuaciones físicas las que determinan las dependencias durante el proceso de simulación.

Esto resulta especialmente interesante, ya que muchos sistemas físicos no tienen relaciones unidireccionales estrictas entre sus variables. Por ejemplo, en el caso de un buque propulsado por velas, la interacción entre el viento, las fuerzas generadas por las velas, el timón y la velocidad del casco no sigue una secuencia lineal. Modelar este tipo de interacciones en un entorno causal implicaría numerosas simplificaciones o ecuaciones auxiliares artificiales, lo que se evita en un entorno acausal como el de OpenModelica.

Las simulaciones en OpenModelica comienzan con la definición de un modelo, construido a partir de bloques que representan los distintos elementos del sistema.

En resumen, OpenModelica permite simular sistemas físicos complejos sin imponer una dirección explícita en las relaciones entre variables, lo cual es esencial cuando se trabaja con sistemas como el de un buque a vela, donde las condiciones de contorno (como el ángulo del viento) cambian constantemente y afectan de forma no lineal.

2.3. Librería SipSim

Uno de los grandes valores añadidos de OpenModelica en el contexto de este TFM ha sido la posibilidad de utilizar bibliotecas específicas que modelan fenómenos físicos concretos. En este caso, se ha empleado la librería ShipSim, desarrollada por Basilio Puente y M. Dolores Fernández, orientada al modelado y simulación del comportamiento de buques y sistemas relacionados.

ShipSim es una librería especializada dentro del entorno Modelica que permite simular de forma realista tanto el casco del buque como sus elementos de control, propulsión y maniobra. Esta librería cuenta con componentes ya preparados para representar la hidrodinámica del casco, las fuerzas externas debidas al viento y al oleaje, el comportamiento de las velas, la influencia del timón, y los efectos del motor propulsor en caso de que se emplee.

En el contexto específico de este TFM, la librería ShipSim ha sido empleada para generar los diagramas polares del buque. Estos diagramas representan gráficamente cómo varía la velocidad del buque en función de dos variables: el ángulo del viento y su intensidad. Para obtener estos resultados, ShipSim permite definir configuraciones con o sin propulsión, ajustar el tipo y disposición de las velas, e incluso simular maniobras a diferentes ángulos de timón.

Por tanto, esta librería es una parte esencial del trabajo, al proporcionar los medios necesarios para estudiar cómo un buque propulsado por velas se comporta bajo diferentes condiciones meteorológicas, y así estimar el impacto en los tiempos de tránsito y la viabilidad económica del proyecto.

2.4. Modelado del buque y disposición de las velas

El modelado del sistema buque vela ha sido realizado en el entorno de simulación acausal OpenModelica, utilizando como base la librería especializada *ShipSim*, desarrollada por Basilio Puente y M. Dolores Fernández (2021). Esta librería incluye una serie de componentes específicos para la simulación de sistemas navales, lo que la convierte en una herramienta idónea para el estudio de la dinámica de un buque propulsado por velas rígidas.

2.4.1. Configuración del buque base

Como punto de partida, se ha empleado uno de los modelos genéricos que la propia librería proporciona: un buque granelero de aproximadamente 100 metros de eslora, 20 metros de manga y 5 metros de calado.

Este modelo ha sido modificado para adaptarse a las condiciones específicas de este TFM, en la que se evalúa la viabilidad del uso exclusivo de propulsión eólica mediante velas rígidas. Por tanto, se ha suprimido todo sistema de propulsión convencional (hélice y motor), introduciendo a cambio cuatro velas rígidas del tipo NACA0015 y un timón de gobierno adaptado a este nuevo entorno de operación.

2.4.2. Velas: disposición y características

Las velas empleadas se modelan utilizando el bloque *RigidWingSail* de la librería *ShipSim*, que permite definir tanto las propiedades geométricas como el comportamiento aerodinámico del sistema. En este estudio se han dispuesto cuatro velas alineadas longitudinalmente sobre

la cubierta principal, con una separación de 20 metros entre cada una de ellas. Todas las velas utilizan un perfil NACA0015.

Todas las velas tienen la misma geometría y comenzarán con el mismo ángulo inicial, el cual podrá variar durante la simulación mediante el input correspondiente, lo que permite realizar estudios de control y maniobrabilidad. Estas velas cuentan con una altura de 20 m. una anchura de 7 metros y un perfil aerodinámico NACA0015 como se ha comentado anteriormente.

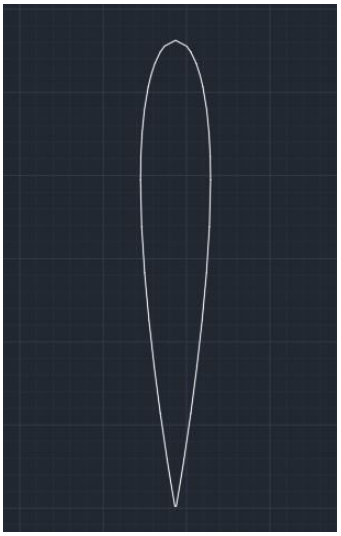

Figura 3: Perfil NACA0015

Para cada vela, se ha utilizado el bloque *FixedTranslation* para posicionarla en coordenadas específicas sobre la cubierta. Estas coordenadas, expresadas en el sistema de referencia del modelo, se recogen en la Tabla 3.

Tabla 3: Coordenadas de ubicación de velas y timón en el modelo

Elemento	Coordenada X [m]	Coordenada Y [m]	Coordenada Z [m]
Vela 1	20	0	15
Vela 2	40	0	15
Vela 3	60	0	15
Vela 4	80	0	15
Timón	-2	0	4,5

Cada vela incluye además los siguientes parámetros configurables dentro del modelo de simulación:

- Cuerda (c)
- Span, altura (s)
- Ángulo inicial respecto a crujía
- Perfil aerodinámico seleccionado
- Input de control para variar el ángulo durante la simulación
- Posición del eje de giro de las velas

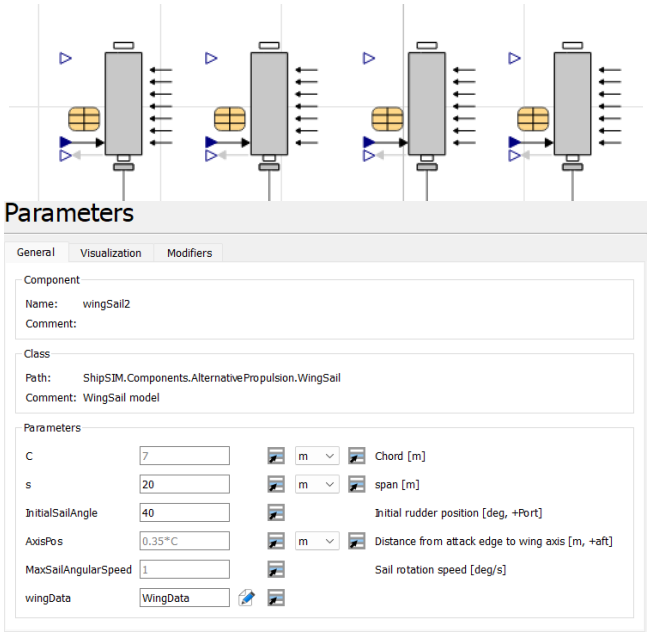

Figura 4: Configuración RigidWindSail en OpenModelica

2.4.3. Timón: adaptación al entorno sin hélice

En ausencia de un sistema de propulsión mecánica, el funcionamiento del timón ha requerido una modificación. Por defecto, *ShipSim* asume que el flujo sobre el timón es generado por una hélice con un diámetro y velocidad asociados. En este estudio, se ha modificado el código del modelo del buque para extraer directamente la velocidad real de avance del casco, que se emplea como input en el modelo del timón. Asimismo, se ha ajustado el diámetro efectivo del flujo, simulando un entorno con un volumen de agua mucho mayor que el generado por una hélice tradicional.

También se ha introducido un coeficiente de estela (wake fraction), que representa la interacción entre el flujo del agua y la popa del buque. Este parámetro es necesario para calcular correctamente la velocidad incidente sobre el timón, especialmente en simulaciones donde el régimen de operación depende únicamente del empuje aerodinámico de las velas.

El ángulo del timón, al igual que las velas, se define mediante un input externo, lo que permite controlar su valor dinámicamente durante cada simulación.

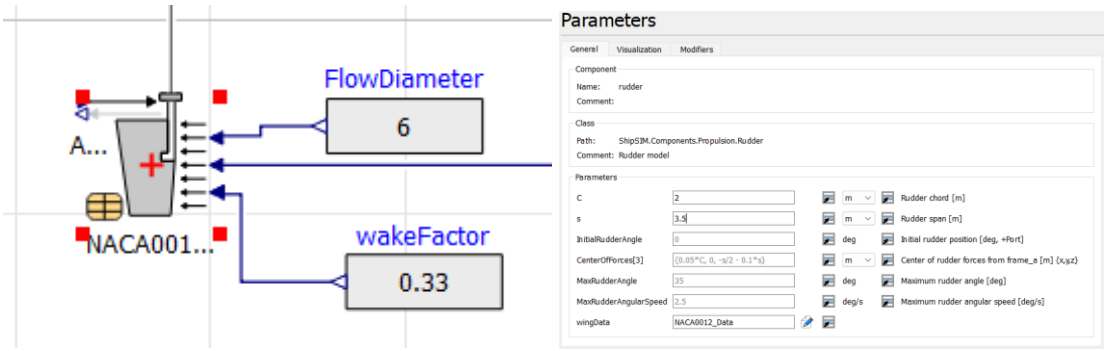

Figura 5: Configuración del timón en OpenModelica

2.5. Integración del modelo y estructura general

La integración del modelo se ha realizado conectando todos los componentes, que transmiten fuerzas, velocidades y momentos sin necesidad de definir explícitamente las direcciones de flujo. Esta característica, propia de la naturaleza acausal del lenguaje Modelica, permite que el sistema sea más flexible y escalable para futuras modificaciones o ampliaciones.

Los principales bloques utilizados en el modelo son:

- World: define la gravedad (dirección sentido y magnitud).
- ShipModelTH: modelo simplificado del buque.
- HydrodynamicXYY: cálculo de las fuerzas hidrodinámicas en las direcciones longitudinal, transversal y rotacional.
- ShipWind: modelo de viento que define la fuerza incidente sobre el casco y las velas.
- RigidWingSail: modelo de vela rígida, uno por cada vela instalada.
- RudderModel: modelo del timón.
- FixedTranslation: elemento de colocación tridimensional de los componentes.

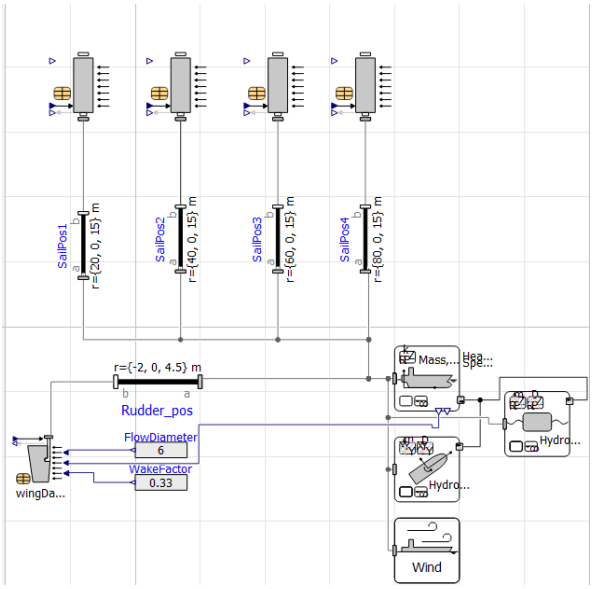

Figura 6: Sistema completo en OpenModelica con los principales bloques conectados

3. Aspectos técnicos

El presente apartado tiene como finalidad exponer y justificar los principales aspectos técnicos que han intervenido en el proceso de modelado, simulación y análisis del comportamiento del buque propulsado a vela. En primer lugar, se detallarán las hipótesis de trabajo adoptadas, así como las simplificaciones necesarias para reducir la complejidad del modelo. Seguidamente, se identificarán las principales limitaciones asociadas tanto al software utilizado como a las librerías aplicadas, lo cual resulta clave a la hora de interpretar correctamente los resultados obtenidos. Asimismo, se describirán las condiciones de contorno implementadas, los ajustes sobre parámetros físicos relevantes y la estrategia de control de los actuadores principales, como el timón y las velas rígidas.

3.1. Limitaciones del software e hipótesis generales

Durante el desarrollo del modelo de simulación del buque propulsado por velas rígidas, ha sido necesario adoptar una serie de simplificaciones que, si bien facilitan el tratamiento computacional del problema, también conllevan ciertas limitaciones en cuanto al nivel de realismo alcanzable. Este apartado recoge las principales restricciones impuestas por el entorno de simulación, así como las hipótesis generales adoptadas para reducir la complejidad del modelo.

En primer lugar, cabe destacar que OpenModelica es un entorno de simulación que no está específicamente diseñado para simulaciones de buques. En concreto, la librería *ShipSim*, aunque ofrece una buena base para el modelado de buques y elementos de propulsión como timones y velas, presenta una serie de limitaciones importantes:

- No se contempla la interacción aerodinámica entre velas. Cada vela se considera de forma independiente, lo que implica que fenómenos como la pérdida de eficiencia debida al apantallamiento de una vela sobre otra no están representados. Este efecto, especialmente relevante en configuraciones de múltiples velas alineadas, podría alterar significativamente el rendimiento en condiciones reales.

- La respuesta hidrodinámica del casco está modelada mediante una función polinómica simplificada de tercer orden, lo que implica que fenómenos como el oleaje no están representados. En otras palabras, el modelo considera una resistencia al avance en aguas tranquilas, sin tomar en cuenta el estado real del mar.

- Se elimina la influencia de las olas tanto en la resistencia al avance como en el comportamiento dinámico del buque.

- Los perfiles aerodinámicos de las velas (NACA 0015) están idealizados y no contemplan la pérdida de sustentación a altos ángulos de ataque, ni otros fenómenos no lineales como la separación del flujo. Se asume que cada vela responde de forma perfectamente estable al viento incidente.

- No se modela el sistema de fondeo, propulsión secundaria ni equipos auxiliares, por lo que su influencia en el peso total y distribución de cargas no ha sido incluida.

Junto con las limitaciones propias del entorno de simulación, se han adoptado una serie de hipótesis de partida que permiten enfocar el modelo:

- El viento se considera constante durante cada simulación. Es decir, no se modelan rachas, turbulencias ni cambios de dirección o intensidad dentro de una misma ejecución. Esto permite aislar el efecto del ángulo y velocidad del viento en cada simulación individual.

- Se considera que las velas pueden ajustarse de forma precisa y automática a cualquier ángulo requerido, sin tener en cuenta, errores de control o condiciones de seguridad operacional.

- No se incluyen pérdidas estructurales, mantenimiento ni eficiencia del sistema de transmisión.

- El buque opera en condiciones normales de carga y con un calado constante, ignorando la variación de masas a lo largo del tiempo.

- Ausencia de propulsión convencional. No se ha definido ninguna fuente de empuje convencional (hélice), por lo que el único avance del buque se produce por la acción del viento sobre las velas rígidas. Esta condición es deliberada, dado que el objetivo del estudio es analizar el potencial de las velas como sistema de propulsión principal en una determinada ruta.

3.2. Condiciones de contorno aplicadas.

Además de las limitaciones del software y las hipótesis previas, es necesario establecer las condiciones de contorno con las que se han configurado todas las simulaciones realizadas en OpenModelica. Estas condiciones son fundamentales, ya que definen el entorno físico en el que se evaluará el comportamiento del buque propulsado mediante velas rígidas.

En OpenModelica, y más concretamente con la librería ShipSim, estas condiciones de contorno se introducen mediante diferentes componentes. Por un lado, el componente *Modelica.Mechanics.MultiBody.World* fija el sistema de referencia global del modelo y establece la gravedad. Por otro, el módulo *ShipSim.Components.Environment* se encarga de definir las propiedades físicas del entorno marino y atmosférico (corrientes y vientos) en el que se desarrollan las simulaciones. A continuación, se detallan ambos bloques y los parámetros que se han configurado:

3.2.1. Gravedad

El componente *Modelica.Mechanics.MultiBody.World* ha sido configurado con una aceleración gravitacional estándar de:

$$g = 9.81\ m/s^2$$

Además, en este componente se define también la dirección y el sentido del vector gravedad respecto al sistema de coordenadas del modelo. En este caso, se ha utilizado la configuración por defecto, donde la gravedad apunta en el eje negativo Z global.

3.2.2. Propiedades del entorno: densidad, viscosidad, viento y corrientes.

Como se ha mencionado anteriormente, la librería *ShipSim* incluye un componente específico denominado *ShipSim.Components.Environment*, encargado de definir los parámetros físicos del entorno en el que se desarrolla la simulación. Este bloque permite establecer no solo las propiedades de los fluidos implicados (aire y agua de mar), sino también las condiciones dinámicas del medio, como la velocidad y dirección del viento o las corrientes marinas.

En primer lugar, dentro de este componente se introducen los valores de densidad y viscosidad cinemática tanto para el aire como para el agua. Estos parámetros son fundamentales, ya que afectan directamente al cálculo de las fuerzas aerodinámicas e hidrodinámicas que actúan sobre el buque, y por lo tanto al equilibrio de fuerzas que determina su velocidad final.

Los valores utilizados en el presente estudio son los siguientes:

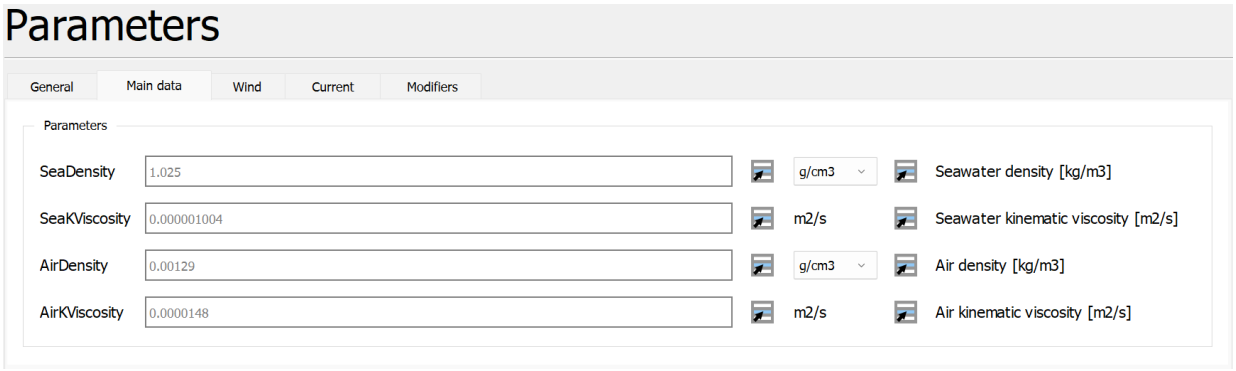

Figura 7: Parámetros de densidad y viscosidad para aire y agua

Además de estas propiedades físicas, el módulo *Environment* permite definir las condiciones de viento y corriente como vectores externos que afectan al comportamiento del buque. Ambos se introducen con dos parámetros principales:

- Velocidad (en m/s)
- Dirección (en grados respecto al eje de referencia global)

Esta forma de introducir el viento permite representar su incidencia como un flujo uniforme, lo cual es útil para el barrido de situaciones de entrada que posteriormente darán lugar al diagrama polar. Para cada simulación, se definen una magnitud concreta del viento y un ángulo de incidencia, variando ángulos de timón y vela. lo que permite observar cómo responde el sistema bajo diferentes configuraciones.

Del mismo modo, en el modelo se puede definir la existencia de corrientes marinas utilizando exactamente el mismo esquema: velocidad constante y dirección de flujo. En este trabajo, no se ha considerado ninguna corriente.

3.2.3. Condiciones de contorno internas al modelo

Se han incorporado también elementos auxiliares necesarios para el correcto funcionamiento del modelo en OpenModelica:

- FixedTranslation: para ubicar el timón y las velas en sus posiciones reales con respecto al centro del buque.

- Entradas (input) definidas: para los ángulos de ataque de las velas y del timón. Estas entradas se controlan mediante señales externas (escalonadas) para simular múltiples configuraciones de navegación.

- Estela y flujo en el timón: dado que no existe propulsión real, se ha ajustado el coeficiente de estela y el diámetro de flujo del timón para que el modelo sea estable, teniendo en cuenta que el flujo detrás del casco no es impulsado por ninguna hélice.

3.3. Estrategia de control del ángulo de las velas y del timón

La lógica de control implementada para automatizar la variación de los ángulos de vela y timón se desarrolló íntegramente utilizando componentes básicos de OpenModelica. El sistema fue diseñado para generar una señal de tipo escalera que incrementa los valores angulares paso a paso con el tiempo, de manera que el modelo pueda recorrer de forma autónoma un rango de combinaciones sin intervención manual.

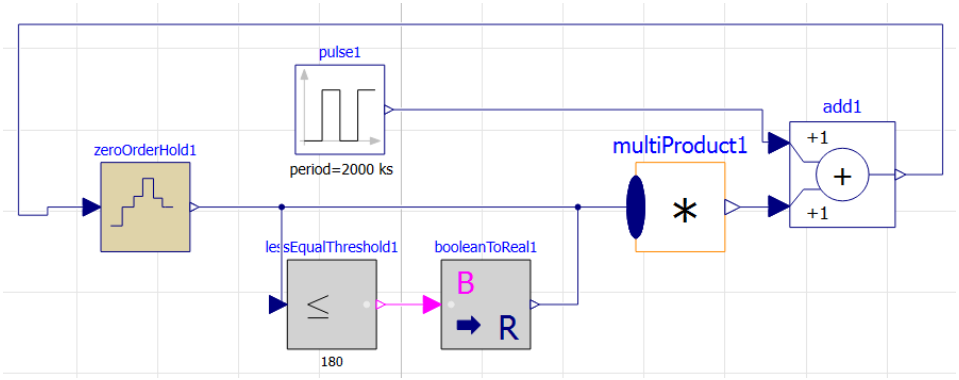

Figura 8: Señal escalera en función del tiempo en *OpenModelica*

La arquitectura utilizada puede observarse en la Figura 8. En ella se identifican los siguientes bloques:

- *Pulse1*: Este generador de pulsos define la amplitud del valor del escalón. En el ejemplo mostrado, la amplitud de la señal está definida en 20, lo cual implica que cada paso angular será de 20º.Este componente también permite definir el valor inicial de la señal escalón, el cual se define en el offset. El tiempo que dure el escalón vendrá definido por el componente *ZeroOrderHold.*

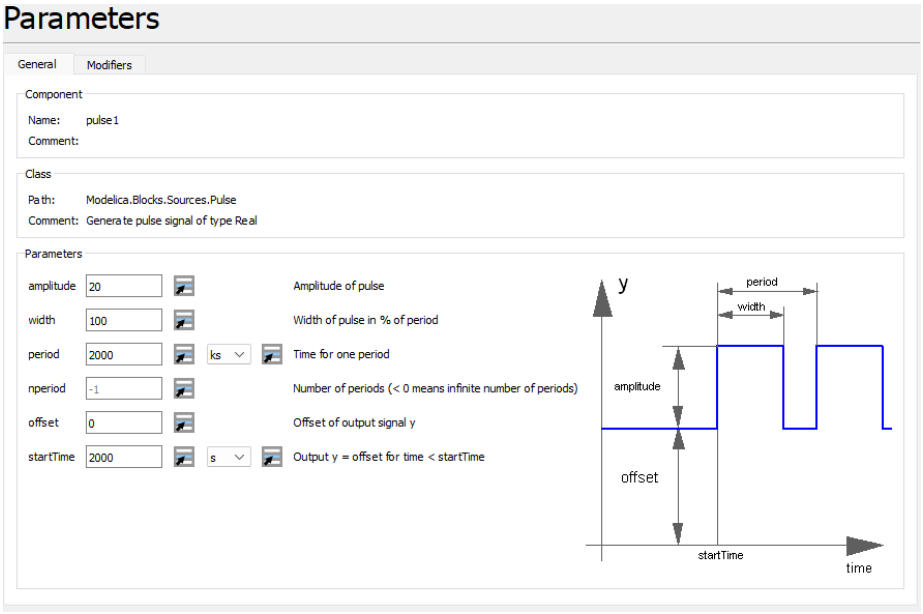

Figura 9: Parámetros del componente *Pulse*

- *LessEqualThreshold1*: Este bloque compara el valor actual del ángulo con un umbral predefinido (en este caso 180º). Cuando se alcanza este límite se activa una lógica para reiniciar el contador, pasar a cero el ángulo de vela y aumentar el ángulo de timón. En este caso cuando la señal de salida del *zeroOrderHold* es menor de 180 la señal de este componente es *True* pero al aumentarlo cambia a *False*, lo cual genera un cambio en el siguiente bloque.

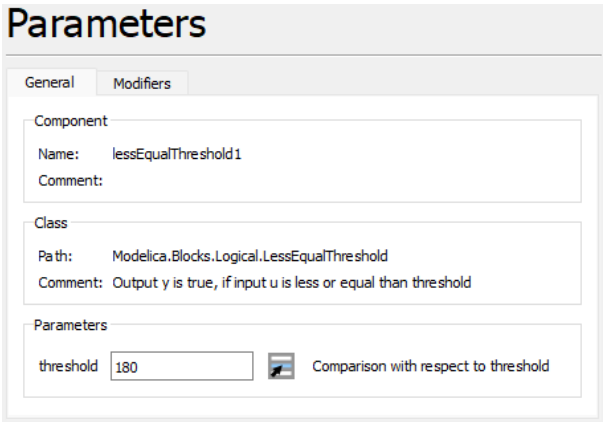

Figura 10: Parámetros del componente *LessEqualThreshold*

- *BooleanToReal1*: El componente convierte la salida booleana del comparador (lessEqualThreshold1) en una señal numérica, facilitando así su integración con los bloques aritméticos que manejan las funciones. En este caso una vez el comparador detecta que se ha superado el límite de 180 genera una señal booleana (False) que transforma a una señal real (0), reiniciando la escalera.

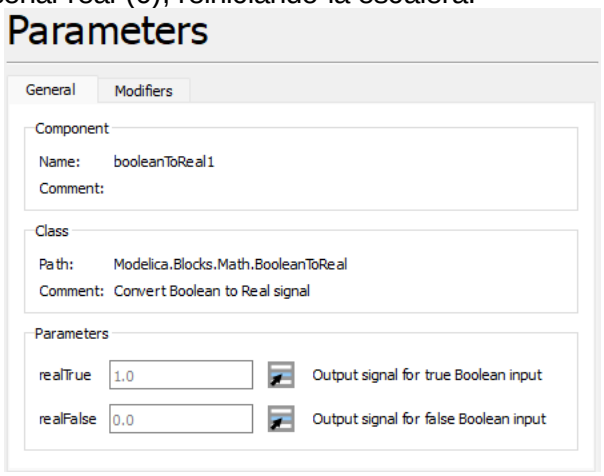

Figura 11: Parámetros del componente BooleanToReal

- *Multiproduct1* y *add1*: Estos bloques realizan la multiplicación y suma necesaria para incrementar el valor de los ángulos paso a paso. En combinación con la lógica anterior, permiten implementar una secuencia de control en la que, por ejemplo, se recorren todos los ángulos de vela entre 0º y 180º antes de incrementar el ángulo de timón en el siguiente ciclo.

- *zeroOrderHold*: Este bloque se utiliza para mantener constante el valor de una señal durante un intervalo de tiempo definido,en este caso 2000 segundos, actualizándose únicamente en los instantes de muestreo especificados. En este sistema, el valor de entrada al bloque es el resultado de la suma del componente pulse y la propia salida del zeroOrderHold, lo que permite generar una señal escalonada donde cada peldaño tiene una amplitud igual a la del pulse. Esta configuración ha sido empleada tanto para el control del ángulo de las velas como del timón, garantizando que cada valor se mantenga fijo durante un tiempo suficiente como para que el sistema alcance una velocidad estable.

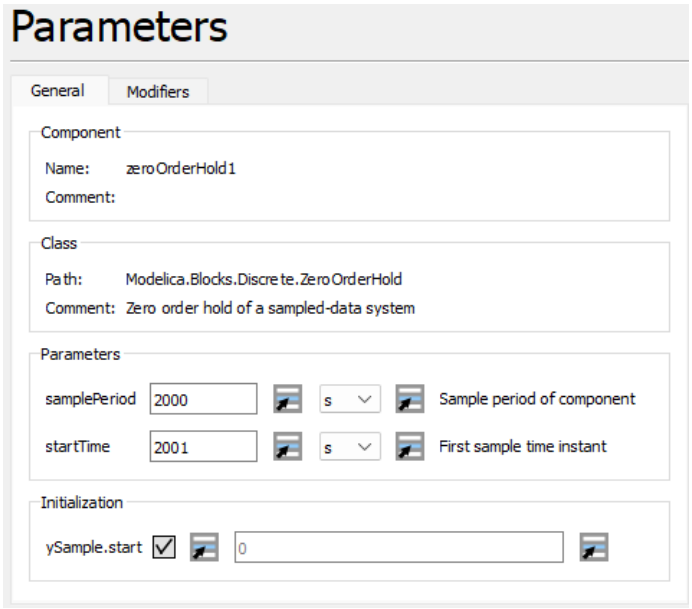

Figura 12: Parámetros del componente *zeroOrderHold*

Mediante la combinación de estos bloques y los parámetros definidos, se ha conseguido generar una señal tipo escalera a la salida del bloque *multiProduct1*. Esta señal aumenta en incrementos determinados por la amplitud del bloque *pulse1*, y se reinicia automáticamente al alcanzar el umbral establecido en el componente *lessEqualThreshold1*. Este comportamiento resulta especialmente útil para definir variaciones progresivas en los ángulos de vela y timón dentro de la simulación, permitiendo controlar el punto de inicio y el límite de cada barrido angular en función de los parámetros configurados en los bloques *pulse1* y *zeroOrderHold1*. Incluyendo dos bloques que generen una señal de escalera se pueden obtener resultados como los siguientes:

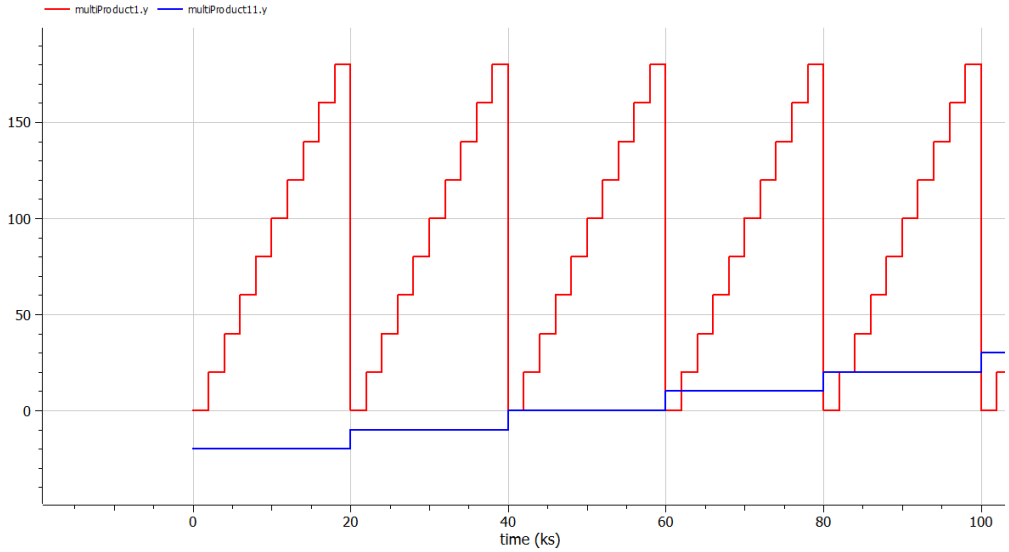

Figura 13: Señal escalera para variación de ángulos de timón y vela

4. Simulaciones y procesamiento de datos

Una vez definido el modelo del buque en OpenModelica, se procede a la fase de simulación, tratamiento de los resultados y elaboración de los diagramas polares. Esta etapa proporciona los datos experimentales que permitirán caracterizar el comportamiento del buque propulsado por velas rígidas bajo distintas condiciones de viento. El objetivo final es generar representaciones gráficas que relacionen la dirección y velocidad del viento con la velocidad

alcanzada por el buque en cada configuración. Para ello, se ha seguido una metodología desde la planificación de los barridos de parámetros, pasando por la extracción y limpieza de los datos simulados, hasta su postprocesado y representación gráfica. En los apartados siguientes se detallan todos estos pasos de forma estructurada.

4.1. Simulaciones realizadas y variabilidad de los parámetros

Antes de abordar el procesamiento de los datos y la construcción final de los diagramas polares, resulta necesario presentar la estructura de simulación desarrollada en OpenModelica. En la Figura 14 se muestra el sistema completo de bloques que ha sido implementado. En él se observa la configuración del buque con las velas dispuestas, así como los módulos escalera encargados de generar las señales que permiten recorrer los distintos ángulos de timón y de velas.

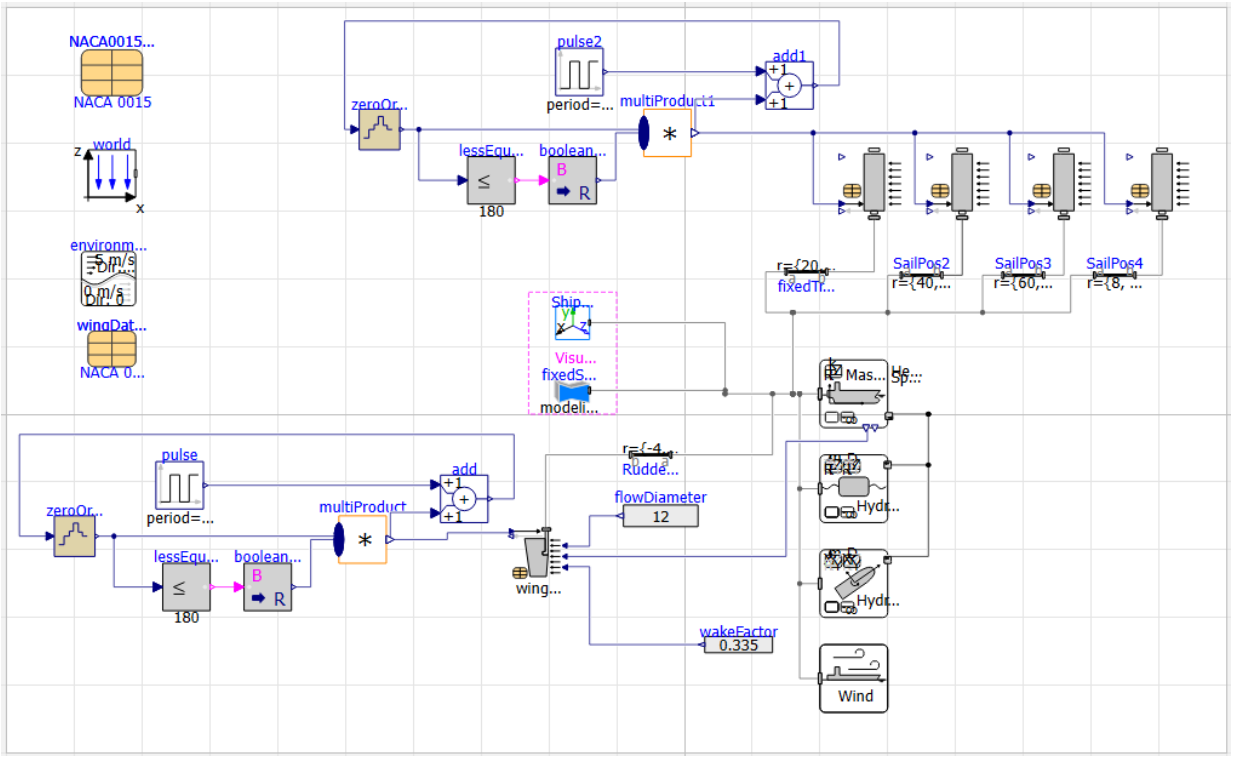

Figura 14: Sistema completo de bloques en OpenModelica de un buque propulsado con velas rígidas

Para cada simulación, se han programado barridos de los ángulos de vela desde 0° hasta 180°, con un incremento de 1° por escalón. Se ha limitado el rango hasta 180° ya que, debido a la simetría del perfil NACA0015 utilizado, no resulta necesario considerar el ciclo completo de 360°. Por otro lado, el ángulo del timón se ha variado desde -35° hasta +35°, incrementando 0,5° por iteración, que representa el rango máximo de operación del sistema de gobierno. Sin embargo, para reducir el número de simulaciones y limitar el tiempo de cálculo, este barrido se ha realizado de 10 en 10 grados de timón, completando así un conjunto más manejable de condiciones de prueba.

Antes de proceder con el análisis completo de combinaciones de ángulos, se ha ejecutado una simulación de ejemplo con una configuración arbitraria de timón y velas. El objetivo de esta prueba era verificar el comportamiento dinámico del sistema y asegurarse de que el buque alcanzaba un régimen estacionario de velocidad. Tal y como puede apreciarse en la Figura 15 correspondiente a esa simulación, el buque logra estabilizar su velocidad antes de los 1800 segundos, lo que ha justificado este valor como tiempo total de simulación para cada ángulo de

vela. Este enfoque permite asegurar que cada conjunto de condiciones alcanza una velocidad suficientemente representativa del comportamiento del buque bajo la influencia del viento.

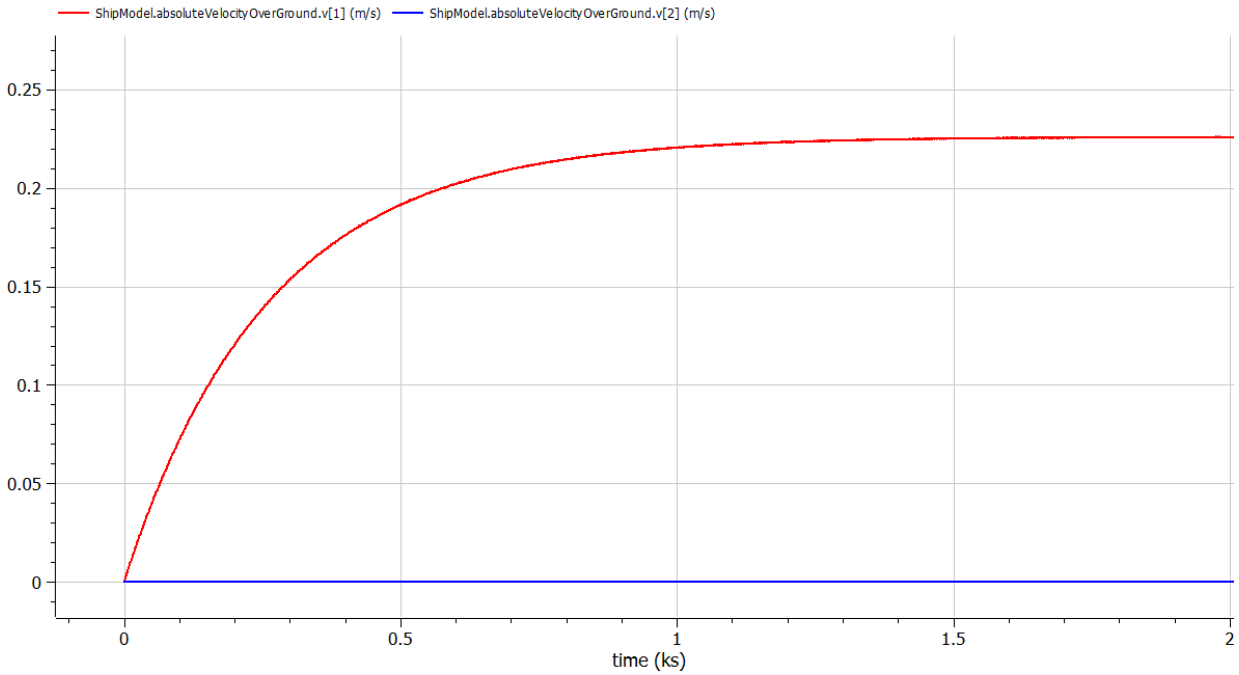

Figura 15: Componentes de la velocidad del buque para ángulo de timón y velas fijo

El procedimiento seguido implica realizar un barrido completo de ángulos de vela (181 pasos) para cada una de las posiciones fijas del timón (de -35° a +35° en pasos de 10°, es decir, 7 configuraciones). Por tanto, para cada simulación se resuelven 181 x 1800 segundos de cálculo, es decir, se genera una señal escalonada con cada uno de los ángulos de timón y vela. Este proceso se repite para siete velocidades de viento distintas, comprendidas entre 5 y 35 nudos, en intervalos de 5 nudos (convertidas a m/s para su uso en el modelo).

El volumen de datos a procesar ha sido considerable. Debido a limitaciones computacionales del equipo utilizado, no fue posible ejecutar una única simulación que integrara todos los escenarios posibles. Que el archivo resultante fuera tan pesado hacía que OpenModelica no pudiese generar las gráficas ni exportar los mismos datos. Por esa razón se optó por dividir cada velocidad de viento en simulaciones independientes, una para cada grupo de ángulos de timón, y cada una con su correspondiente barrido de ángulos de vela.

En resumen, el estudio ha requerido la ejecución de:

- 7 configuraciones de timón por velocidad de viento
- 7 velocidades de viento distintas
- Lo que resulta en un total de 49 simulaciones independientes

Un ejemplo de los resultados obtenidos en una simulación es el que se puede apreciar en la Figura 16, la cual representa datos reales obtenidos de OpenModelica.

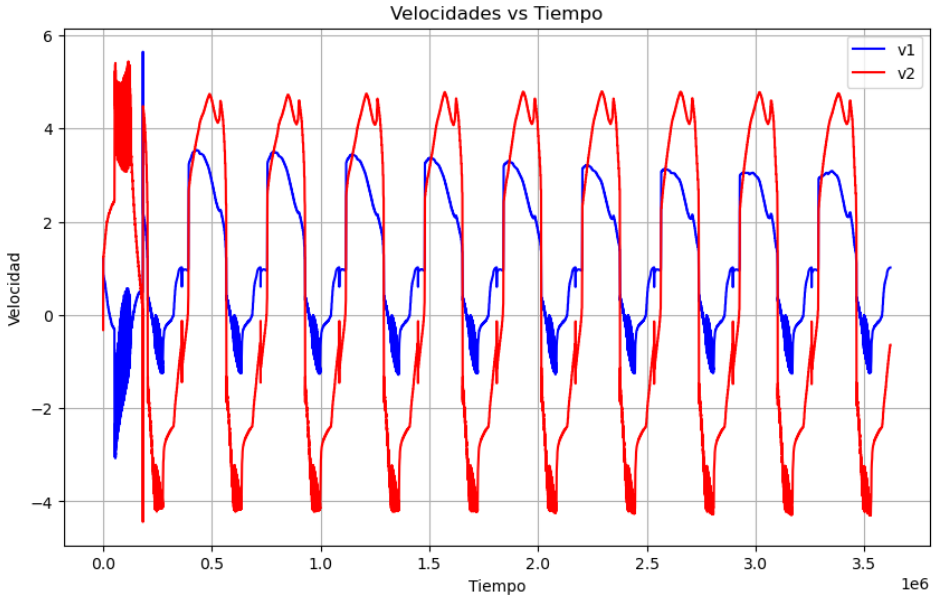

Figura 16: Resultados de una simulación con cambios de ángulo de vela y timón

Cada simulación tiene una duración de entre 10 y 20 minutos en función del paso de integración temporal establecido, lo que supone un mínimo de 8,5 h de simulaciones, sin tener en cuenta el tiempo de exportación de los datos y representación gráfica.

Este planteamiento, aunque demandante a nivel computacional, ha permitido obtener una base de datos sobre el comportamiento del buque propulsado por velas bajo diferentes combinaciones de ángulos y condiciones de viento.

En cuanto a la obtención de resultados, los datos se exportan directamente desde OpenModelica en formato CSV. Por cada simulación se genera un archivo independiente que recoge la información solicitada en la interfaz de representación del software. En el presente estudio, para caracterizar el comportamiento del buque, se han extraído principalmente las componentes longitudinales (v[1]) y transversal (v[2]) de la velocidad absoluta sobre el fondo (absoluteVelocityOverGround). Estas dos columnas, junto con la columna de tiempo, conforman la estructura típica de los archivos generados.

La elección de la velocidad *OverGround* como referencia se debe a que es la que refleja de forma más precisa el desplazamiento real del buque, permitiendo posteriormente calcular con exactitud los tiempos de tránsito entre puertos. El tamaño final de cada archivo CSV depende directamente del intervalo de muestreo establecido en la simulación, ya que un paso temporal más reducido genera un mayor volumen de datos y, por tanto, archivos más pesados.

4.2. Postprocesamiento de datos y generación de diagramas polares

El tratamiento de los datos obtenidos de las simulaciones constituye un paso clave para la obtención de diagramas polares. Para ello, se han desarrollado y utilizado varios scripts en Python

En primer lugar, a partir de los archivos CSV exportados desde *OpenModelica*, se generan dos nuevas columnas para cada conjunto de datos. La primera de ellas corresponde al rumbo o ángulo de avance del buque, calculado a partir de sus componentes de velocidad longitudinal y transversal mediante la expresión:

$$\theta = abs(arctan2(v_y, v_x)$$

La segunda columna recoge la magnitud de la velocidad, obtenida mediante la raíz cuadrada de la suma de los cuadrados de dichas componentes:

$$V = \sqrt{(v_x^2 + v_y^2)}$$

Estas dos variables (ángulo y magnitud de la velocidad) permiten caracterizar el comportamiento buque.

Una vez generados estos datos, se procede a un filtrado con el objetivo de retener únicamente los valores más representativos para la construcción del diagrama polar. Para ello, el rango angular total se divide en intervalos de 0,1 radianes y, para cada intervalo, se selecciona el valor máximo de velocidad, descartando el resto. Este proceso, implementado mediante un bucle *for*, garantiza que se conservan únicamente los puntos que definen la "envolvente" superior de las velocidades alcanzadas, eliminando así valores intermedios o redundantes.

El procedimiento anterior se aplica de forma automática a todos los archivos de una misma carpeta (cada uno correspondiente a una combinación específica de ángulo de timón) y genera un único archivo CSV con los puntos seleccionados. A partir de este conjunto, se interpola para obtener curvas suaves y continuas que representen el comportamiento del buque.

En este punto, se evaluaron diferentes técnicas de interpolación para seleccionar la que ofreciese un compromiso óptimo entre precisión y realismo. Se probaron tres métodos:

- Interpolador lineal (*LinearSpline*): conecta los puntos con segmentos rectos, generando transiciones poco naturales para un diagrama polar.

- Interpolador de Lagrange: aunque teóricamente adecuado para pasar por todos los puntos, en la práctica mostró desviaciones significativas en los extremos del rango angular, además de problemas de estabilidad en los valores intermedios.

- Interpolador cúbico (*CubicSpline*): proporciona una curva continua con derivadas suaves, adaptándose de forma más realista a las formas redondeadas características de los diagramas polares.

Tras la comparación, se optó por el *CubicSpline* debido a que generó las curvas más naturales y precisas, minimizando desviaciones respecto a los datos originales y evitando los problemas de los otros métodos.

El resultado de este proceso es un conjunto de curvas suavizadas que definen, para cada velocidad de viento simulada, la relación entre el ángulo de ataque del viento y la velocidad efectiva del buque. Estas curvas interpoladas constituyen la base para su posterior uso en los cálculos de tiempo de viaje y en la evaluación del rendimiento en distintas condiciones operativas.

4.3. Consideración de la interacción entre velas

Una vez obtenidos los datos brutos de velocidad del buque para cada combinación de ángulos de vela y timón, se ha incorporado una corrección destinada a considerar el efecto de la interacción entre las velas. Este fenómeno, conocido como apantallamiento, se produce cuando las velas situadas a sotavento reciben un flujo de viento reducido debido a la presencia de velas situadas a barlovento.

El buque modelado cuenta con cuatro velas dispuestas longitudinalmente a lo largo de la cubierta, separadas 20 metros entre sí. Cada vela presenta la misma geometría (cuerda y altura) y se orienta de forma paralela a las demás, de manera que comparten el mismo ángulo respecto a la crujía del buque.

Para estimar el efecto de apantallamiento, el cálculo parte del viento aparente que incide sobre la embarcación, el cual se obtiene restando vectorialmente la velocidad de avance del buque a la velocidad del viento verdadero. Este viento aparente es el que realmente incide en la vela y, por tanto, el que define el empuje aerodinámico. Una vez determinado este vector, se calcula la proyección perpendicular sobre el plano de la vela, lo que permite obtener el ángulo de incidencia efectivo.

Bajo un enfoque conservador, se asume que:

- El viento aparente incide perpendicularmente a todas las velas.
- Todas las velas tienen el mismo ángulo de ataque y, por tanto, son paralelas entre sí.
- La primera vela recibe el 100 % del flujo incidente, mientras que las velas posteriores solo reciben una fracción proporcional a la sección eficaz proyectada que no ha sido apantallada por la vela precedente.

La sección eficaz (*Sef*) para cada vela situada a sotavento se determina mediante:

$$Sef = d \cdot sin(\alpha)$$

donde:
- d es la distancia entre velas,
- α es el ángulo entre la dirección del viento aparente y la crujía del buque.

En el caso de contar con cuatro velas, la superficie total efectiva sobre la que actúa el viento se expresa como:

$$Stotal, ef = S_{vela} + 3 \cdot S_{ef}$$

donde S_{vela} es la superficie real de una vela.

La relación entre la velocidad corregida (V_{red})y la velocidad original (V_{orig}) se obtiene considerando que la fuerza ejercida por las velas es proporcional a la densidad del aire (ρ), a un coeficiente aerodinámico C, a la superficie efectiva S y al cuadrado de la velocidad aparente:

$$F \propto \frac{1}{2} \rho \, C \, S V^2$$

Si se mantienen constantes ρ y C, el cociente entre la fuerza con interacción y la fuerza sin interacción se reduce a:

$$\frac{F_{int}}{F_{sin}} = \frac{S_{total,ef}}{4 \cdot S_{vela}}$$

y, en consecuencia, la relación entre las velocidades es:

$$\left(\frac{V_{red}}{V_{orig}}\right)^2 = \frac{3 \cdot S_{ef} + S_{vela}}{4 \cdot S_{vela}}$$

lo que nos permite obtener:

$$V_{red} = V_{orig} \cdot \sqrt{\frac{3 \cdot S_{ef} + S_{vela}}{4 \cdot S_{vela}}}$$

En este estudio se ha optado por una aproximación simplificada para estimar la reducción de velocidad debido a la interacción entre velas. El planteamiento parte de la premisa de que, manteniendo constantes la densidad del aire, el coeficiente aerodinámico y la velocidad aparente del viento, la fuerza generada por el conjunto de las velas es directamente proporcional a la superficie efectiva expuesta al viento. Dado que la resistencia al avance del buque se ha considerado, en primera aproximación, proporcional al cuadrado de la velocidad, la relación entre la velocidad corregida y la velocidad original puede expresarse como la raíz cuadrada del cociente entre la superficie efectiva con apantallamiento y la superficie total sin interacción.

Este enfoque es conservador por varias razones. En primer lugar, asume que las pérdidas de rendimiento por apantallamiento afectan de igual manera a todas las velas situadas a sotavento de la primera, lo que reduce la superficie efectiva total de forma más severa de lo que ocurriría en condiciones reales. En segundo lugar, el cálculo geométrico de la zona eficaz de cada vela se ha realizado considerando únicamente la proyección perpendicular del viento aparente sobre el plano de la vela, lo que no contempla posibles ganancias aerodinámicas por canalización del flujo o efectos tridimensionales que podrían incrementar el empuje real.

5. Resultados y análisis de diagramas polares

En este apartado se presentan y analizan los diagramas polares obtenidos tras el proceso de simulación y postprocesado. Las gráficas cubren un rango de velocidades de viento aparente comprendido entre 5 y 35 nudos, con incrementos de 5 nudos. Cada diagrama se ha calculado para ángulos de incidencia del viento comprendidos entre 180° (viento completamente por popa) y 35° respecto a la proa. Este límite superior se debe a que, con vientos más cercanos a la proa (entre 0° y 35°), la capacidad de propulsión a vela se reduce drásticamente debido a la pérdida de sustentación, imposibilitando el avance.

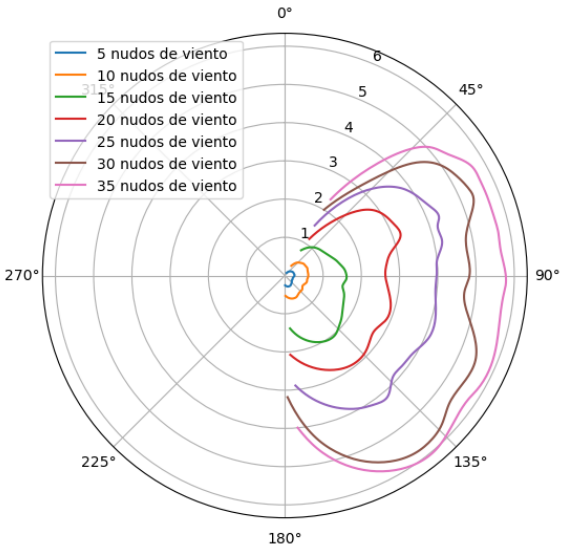

Figura 17: Diagrama polar del buque base

En los resultados, se observa que los diagramas polares muestran un mejor comportamiento cuanto mayor es la velocidad de viento, presentando curvas más suaves. No obstante, la ganancia de velocidad del buque se reduce significativamente entre 30 y 35 nudos, evidenciando una tendencia a la saturación debido a las limitaciones hidrodinámicas y estructurales del buque. Por razones de seguridad y realismo operativo, no se han considerado vientos superiores a 35 nudos, ya que estos podrían asociarse a condiciones meteorológicas extremas (temporales o tormentas) que comprometerían la integridad estructural y la seguridad de la navegación.

Los resultados iniciales muestran ciertas irregularidades y discontinuidades en las curvas polares, como se puede apreciar en la Figura 18. Este comportamiento se atribuye, en gran medida, a las limitaciones de OpenModelica y a la complejidad de la librería ShipSim, que proporciona una base sólida para la modelización del buque y sus velas, pero no alcanza la precisión de herramientas más avanzadas o específicas de dinámica de fluidos computacional (CFD). Tras aplicar el filtrado y la interpolación mediante *splinecúbico*, la curva resultante adquiere una forma más continua y coherente con el comportamiento esperado de la propulsión a vela. Este mismo procedimiento se ha aplicado a todas las velocidades de viento consideradas, obteniendo mejoras similares en la calidad y legibilidad de las curvas, aunque solo se presenta aquí un caso para simplificar la exposición.

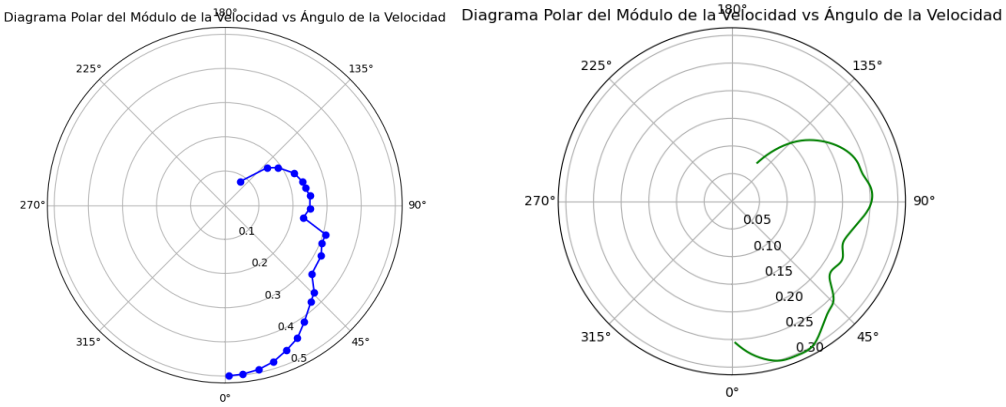

Figura 18: Diagrama polar para 5 nudos antes y después del postprocesado

El diagrama polar en sí constituye una representación bidimensional en la que el radio de cada punto indica la velocidad de avance del buque para un ángulo específico del viento incidente. Su utilidad radica en que permite, de un solo vistazo, identificar los rangos de ángulos y velocidades de viento más favorables para la navegación, así como estimar el comportamiento del buque en condiciones menos propicias. En nuestro caso, se ha aplicado además una corrección por interacción entre velas para reflejar la pérdida de eficiencia causada por el apantallamiento de las velas situadas más a popa.

A lo largo de las distintas condiciones de viento, se observa que:

- Para vientos de popa y aleta (ángulos cercanos a 180°), las velocidades alcanzadas son relativamente altas, aunque con una dependencia notable de la intensidad del viento.

- En vientos de través (alrededor de 90°), se logran buenas velocidades, haciendo de este rango uno bastante eficiente para la propulsión a vela.

- Conforme el ángulo se reduce hacia vientos de ceñida (>30° y <90°), la velocidad disminuye progresivamente hasta alcanzar el límite.

Finalmente, cabe señalar que, aunque el presente trabajo se ha desarrollado con recursos limitados, el resultado obtenido es una primera aproximación al comportamiento de un buque

granelero equipado con velas rígidas. Esta fase de obtención de diagramas polares es especialmente relevante, ya que representa una parte fundamental sobre el que se sustenta la etapa siguiente de desarrollo del algoritmo de *weather routing*. Sin una caracterización como esta, no sería posible estimar con los tiempos de viaje ni realizar adecuadamente el análisis económico posterior.

Capítulo 4
Algoritmo de weather routing

Tras la obtención y procesado de los diagramas polares en el capítulo anterior, el siguiente paso consiste en implementar un algoritmo de *weather routing* que permita estimar los tiempos de viaje en condiciones reales. Esta herramienta combina los datos polares con información meteorológica histórica para simular trayectorias y velocidades del buque a lo largo de una ruta definida. La finalidad es, por un lado, cuantificar el impacto de las condiciones de viento sobre la operación y por otro, generar un conjunto de resultados que sirvan de base para el análisis económico posterior. El desarrollo de este algoritmo ha implicado la selección de fuentes de datos fiables, una revisión de enfoques existentes y la adaptación de soluciones a las necesidades específicas.

1. MERRA-2, datos de viento reales

Para que el algoritmo de optimización de rutas meteorológicas (*weather routing*) sea útil, su pilar fundamental debe ser una fuente de datos de viento precisa, fiable y con una buena cobertura tanto temporal como espacial. Por ello, una de las primeras tareas del proyecto fue la de investigar y seleccionar una base de datos de primer nivel que sirviera como motor para todas las simulaciones.

La investigación de distintas fuentes de datos, guiada en gran parte por la documentación de la plataforma *Renewables.ninja* (una herramienta de referencia en el análisis de recursos para energías renovables), nos llevó a seleccionar MERRA-2 como la opción idónea.

MERRA-2 son las siglas de *Modern-Era Retrospective analysis for Research and Applications, Version 2*. Se trata de un reanálisis atmosférico global desarrollado y mantenido por la Oficina de Modelado y Asimilación Global (GMAO) de la NASA. (Global Modeling and Assimilation Office (GMAO), 2015)

Lo que hace la NASA para obtener dichos datos del MERRA es lo siguiente:

- Ejecutar un modelo numérico: Utilizan un modelo computacional avanzado que simula el comportamiento de la atmósfera de todo el planeta.

- Asimilar datos reales: A medida que el modelo avanza en el tiempo, lo corrigen y lo ajustan constantemente introduciendo millones de observaciones reales procedentes de una enorme variedad de fuentes: satélites, globos meteorológicos, boyas oceánicas, aviones y estaciones en tierra.

El resultado es una recreación del clima histórico que cubre todo el globo con una resolución bastante alta. Es una base de datos realista y completa del tiempo que ha hecho en cualquier parte del mundo desde 1980 hasta el día de hoy.

Aunque para este trabajo nos centraremos exclusivamente en el viento, el valor de MERRA-2 reside en la cantidad y variedad de variables atmosféricas que ofrece, todas ellas con una resolución temporal horaria. Algunos de los datos que se pueden obtener, además de las componentes del viento, son:

- Temperatura a diferentes altitudes.
- Presión a nivel del mar y en altura
- Humedad específica y relativa.

- Precipitación: Lluvia, nieve y otros tipos.
- Nubosidad: Porcentaje de cielo cubierto y tipo de nubes.
- Radiación solar: Radiación de onda corta y larga en la superficie, clave para estudios de energía solar.
- Aerosoles: Concentración de partículas en suspensión como polvo, sal marina o cenizas.
- Variables de la superficie terrestre como la humedad del suelo o la evaporación.

Esto convierte a MERRA-2 en una herramienta potente no solo para la meteorología, sino para campos muy diversos. Para este proyecto, sus datos de viento a lo largo de toda una ruta oceánica son indispensable.

Dentro de la gran cantidad de información que ofrece MERRA-2, es necesario seleccionar el conjunto de datos o paquete que mejor se adapte a las necesidades del proyecto. Para este estudio, el paquete elegido es el M2T1NXFLX. Su nombre completo es el siguiente:

MERRA-2 tavg1_2d_flx_Nx: 2d,1-Hourly,Time-Averaged,Single-Level,Assimilation,Surface Flux Diagnostics V5.12.4

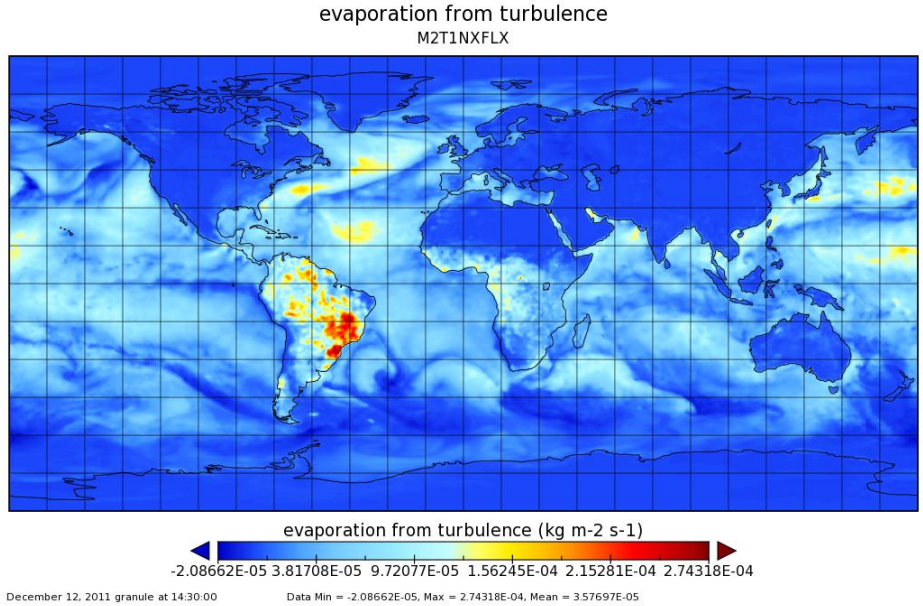

Figura 19: Paquete de datos M2T1NXFLX(Global Modeling and Assimilation Office (GMAO), 2015)

El nombre nos da bastante información de la fuente de datos :

- *tavg1_2d_flx_Nx*: Esta parte nos indica que son datos promediados en el tiempo (tavg), en 2 dimensiones (2d), relacionados con los flujos de superficie (flx).

- *1-Hourly*: Se refiere a la resolución temporal. Esto significa que tenemos datos de las condiciones atmosféricas para cada hora de cada día

- *Single-Level*: Indica que los datos corresponden a un único nivel de altitud, en este caso, cerca de la superficie, que es donde opera nuestro buque.

- *Assimilation*: Vuelve a recalcar que los datos son producto del proceso de asimilación, combinando modelos y observaciones reales.

Para el propósito de este estudio, esta resolución es suficiente. Estamos analizando el comportamiento de un buque granelero, un buque de grandes dimensiones e inercias, cuyos cambios de rumbo y velocidad son muy lentos. Disponer de datos cada minuto, por ejemplo, no

aportaría una mejora en la precisión de la ruta y multiplicaría en gran manera la cantidad de datos a procesar. Por otro lado, una resolución menor (por ejemplo, cada 6 horas) podría hacer que el algoritmo no detectara cambios de viento importantes que sí afectan a la navegación.

La resolución espacial nativa de MERRA-2 es de 0.5 grados de latitud por 0.625 grados de longitud, esto significa que la NASA ha dividido la superficie de todo el planeta en una malla de celdas rectangulares. En el ecuador, cada una de estas celdas mide aproximadamente 55 km por 69 km. Para cada una de estas celdas y para cada hora desde 1980, MERRA-2 nos ofrece un valor para todas sus variables.

Cuando nuestro algoritmo de *weather routing* necesita saber el viento en un punto concreto del océano por el que navega el barco, lo que hace es identificar en qué celda de esta malla se encuentra y extrae los datos de viento correspondientes. Esto nos asegura tener información de viento realista en cualquier lugar del planeta por el que pueda transcurrir la ruta.

Como se mencionó anteriormente, de todas las variables disponibles en este paquete, para el cálculo de la ruta solo se utilizarán las componentes de viento hacia el este (U) y hacia el norte (V), que nos permiten determinar con precisión la magnitud y dirección del viento en cada punto.

1.1. Obtención de datos

Una vez identificado el conjunto de datos M2T1NXFLX de MERRA-2 como la fuente se identificó acceder a esta información de forma que sea compatible con el algoritmo de *weather routing*

El punto de partida fue el Jupyter Notebook proporcionado por la propia NASA, que ilustra las diferentes vías de acceso mediante código Python. Es crucial señalar que, para poder acceder a los datos, es requisito indispensable registrarse como usuario en la plataforma *Earthdata* de la NASA.

Las metodologías de acceso propuestas presentan un dilema entre velocidad, almacenamiento y conveniencia. Las opciones eran las siguientes:

- Acceso en *streaming*: Permite leer datos específicos directamente desde el servidor de la NASA. Aunque evita la necesidad de almacenamiento local, su principal inconveniente es los tiempos de espera, ya que cada consulta puede tardar varios segundos, un tiempo demasiado elevado para un algoritmo que necesita realizar millones de consultas.

- Descarga y borrado puntual: Un enfoque intermedio que descarga el archivo diario completo, extrae la información y luego lo elimina. Sigue presentando un cuello de botella significativo en el tiempo de descarga de cada fichero.

- Descarga y almacenamiento local: Esta es la estrategia que garantiza el máximo rendimiento, con tiempos de acceso casi instantáneos una vez los datos están en el disco duro. Para un algoritmo que depende de la velocidad de consulta, esta es la alternativa funcional.

Las limitaciones de las demás opciones obligaban al almacenamiento local, sin embargo, esto llevaba a un segundo problema, el volumen de datos a almacenar.

Cada archivo diario del paquete M2T1NXFLX ocupa aproximadamente 380 MB. Dado que el estudio utiliza los datos de todo el año 2024 (366 días), el espacio de almacenamiento total requerido se calcula en:

$$380\,\frac{MB}{día}\cdot 366\ días = 139080\ MB\ \approx 139{,}1\ MB$$

Un total de casi 140 GB de datos es una cantidad muy elevada para un portátil de trabajo estándar.

Para resolver el problema entre rendimiento y almacenamiento, se diseñó e implementó una solución de reducción de datos. El objetivo era conservar la velocidad de acceso del almacenamiento local, pero con una cantidad de datos drásticamente menor.

Para ello, se desarrolló un script en Python que automatiza una serie de tareas para cada uno de los 366 días del año a analizar. El flujo de trabajo del script es el siguiente:

1. Descarga del archivo original: Se conecta al servidor de la NASA y descarga el archivo NetCDF (.nc4) completo del día correspondiente, con su tamaño original de 380 MB, el cual tiene el siguiente formato: https://data.gesdisc.earthdata.nasa.gov/data/MERRA2/M2T1NXFLX.5.12.4/2024/07/MERRA2_400.tavg1_2d_flx_Nx.20240705.nc4

2. Extracción de variables: Utilizando librerías especializadas como xarray, se abre el conjunto de datos y se seleccionan exclusivamente las variables necesarias para el proyecto: las componentes de viento U10M (viento hacia el este a 10 metros) y V10M (viento hacia el norte a 10 metros).

3. Creación del archivo reducido: Se genera un nuevo archivo NetCDF que contiene únicamente las dos variables de viento seleccionadas, junto con sus metadatos asociados (coordenadas de latitud, longitud y tiempo).

4. Eliminación del archivo original: Una vez creado y verificado el nuevo archivo reducido, se procede a eliminar el fichero original de 380 MB para liberar el espacio en disco.

Este proceso logró reducir el tamaño de cada archivo diario de 380 MB a tan solo 23 MB, lo que supone una reducción de aproximadamente el 94% del tamaño original.

Gracias a esta optimización, el espacio total de almacenamiento necesario para la base de datos de viento de todo un año pasó de los casi 140 GB a una cifra mucho más manejable:

$$23\,\frac{MB}{día}\cdot 366\ días = 8418\ MB\ \approx 8{,}4\ GB$$

El resultado final es una base de datos local, completa y de acceso casi instantáneo, que sienta una base sólida y eficiente para la ejecución del algoritmo de *weather routing*.

1.2. Archivo NetCDF4

La implementación de la solución de procesamiento de datos no solo resolvió el problema del volumen de almacenamiento, sino que también permitió conservar los datos en su formato original, NetCDF (Network Common Data Form), con extensión .nc4. La decisión de mantener este formato, en lugar de migrar a otros más genéricos como CSV fue debido a que NetCDF presenta una serie de ventajas técnicas.

NetCDF es un estándar en las ciencias atmosféricas, oceanográficas y climáticas para el almacenamiento y la compartición de datos científicos en formato de matrices o *arrays*. Sus

características principales explican por qué se considera el formato idóneo para este tipo de aplicación:

- Es autodescriptivo. Una de sus mayores fortalezas es que en un mismo archivo se almacenan no solo los datos numéricos brutos, sino también los metadatos que los contextualizan. Esto significa que el propio archivo contiene información sobre las variables que almacena (ej. U10M, V10M), sus unidades (ej. m/s), el sistema de coordenadas al que pertenecen (latitud, longitud, tiempo) y cualquier otra información descriptiva relevante. Esto convierte a cada archivo en una especie de contenedor de datos auto explicativo, lo que garantiza la coherencia y reduce enormemente la posibilidad de errores en la interpretación de los datos.

- Eficiencia con datos matriciales. El formato fue diseñado específicamente para un acceso eficiente a subconjuntos de datos dentro de grandes matrices multidimensionales, que es precisamente la estructura de la información de viento (una malla 2D que varía en el tiempo). Esto permite que el algoritmo de *weather routing* consulte porciones específicas del archivo sin necesidad de cargar el fichero completo en la memoria RAM del sistema.

- Portabilidad e interoperabilidad. Los archivos generados son completamente portables, lo que significa que son independientes del sistema operativo o la arquitectura del ordenador en el que se crearon. Un archivo .nc4 generado en un sistema Linux puede ser leído sin ningún problema en Windows o macOS, siempre que se disponga de las librerías adecuadas. Esta característica facilita la colaboración, el uso de diferentes equipos de trabajo y asegura la reproducibilidad de los resultados a largo plazo.

De esta forma, se logró transformar un conjunto de datos de origen muy pesado y de difícil gestión a una base de datos local. El resultado es una colección de archivos de viento para todo un año que constituye el pilar fundamental sobre el que se construye y opera el algoritmo de optimización de rutas desarrollado en este trabajo.

2. Revisión bibliográfica

Para contextualizar el trabajo desarrollado en esta tesis, se ha realizado una revisión de la literatura existente en el campo de la optimización de rutas para buques con sistemas de asistencia eólica. Un estudio considerado es el presentado a continuación, titulado: *"AN OPTIMIZATION METHOD FOR ECONOMICAL SHIP-ROUTING AND SHIP OPERATION CONSIDERING THE EFFECT OF WIND-ASSISTED ROTORS"* (Sun et al., 2020). Este trabajo aborda un problema muy similar al del presente trabajo, por lo que su análisis es fundamental para establecer un punto de partida.

El objetivo principal de este estudio. es proponer un método de optimización para la ruta y la operación de un buque con el fin de minimizar el Consumo Total de Combustible (TFOC). Para ello, se desarrolla una estrategia que combina un algoritmo de búsqueda con un optimizador en tiempo real.

El núcleo de su método de enrutamiento es una versión mejorada del algoritmo A*. El algoritmo A* es un conocido método de búsqueda en grafos que se utiliza para encontrar la ruta más corta entre dos puntos. La metodología del estudio se basa en un enfoque de celdas (*cell-based*), donde la carta náutica se discretiza en una rejilla.

Un aspecto fundamental de este trabajo es el modelo desarrollado para estimar el consumo de combustible en condiciones reales de mar. Este modelo tiene en cuenta tanto los efectos

negativos de la resistencia al avance como los positivos del sistema de asistencia eólica. Concretamente, se consideran los siguientes factores:

- La resistencia al avance del buque.
- La resistencia al viento sobre la superestructura del buque.
- El empuje adicional generado por los rotores de asistencia eólica (rotores Flettner)

Para que el proceso de optimización sea computacionalmente viable, se destaca que en lugar de calcular estas complejas interacciones físicas en tiempo real para cada una de las miles de posibilidades que evalúa el algoritmo, se recurre a un modelo basado en datos Se utiliza una Red Neuronal Artificial (ANN) entrenada previamente con más de 32,000 simulaciones para predecir de forma casi instantánea la velocidad resultante del buque y su consumo de combustible bajo unas condiciones de mar y operación dadas.

El método propuesto se aplicó a un buque petrolero de gran tamaño (VLCC) equipado con seis rotores Flettner. Se analizaron dos rutas comerciales: una en el Océano Pacífico y otra en el Océano Índico.

Los resultados demostraron que la optimización de la ruta y la operación del buque permite obtener ahorros de combustible significativos en comparación con la ruta de distancia mínima.

- Para la ruta del Océano Pacífico, se logró una reducción del TFOC de hasta un 15% al combinar la optimización de ruta, la gestión de la potencia del motor y el uso de los rotores.

- Para la ruta del Océano Índico, caracterizada por vientos laterales fuertes y constantes, el efecto de los rotores fue aún más pronunciado, contribuyendo a un ahorro de más del 8% por sí solos y alcanzando un ahorro total de casi el 17% en el escenario más optimizado.

La conclusión principal del estudio es que la integración de sistemas de asistencia eólica, junto con algoritmos de optimización de rutas y operaciones, es una solución eficaz para la reducción del consumo.

3. Flujograma del algoritmo

En este apartado se detalla el proceso de diseño y desarrollo del algoritmo de *weather routing*, que constituye el núcleo de este Trabajo de Fin de Máster. Para ofrecer una visión completa del trabajo realizado, no solo se presentará la versión final del algoritmo, sino que se expondrá su evolución cronológica. Se comenzará por el planteamiento conceptual inicial y las primeras implementaciones, analizando sus limitaciones y los desafíos computacionales que surgieron. A continuación, se describirán las sucesivas optimizaciones para resolver dichos problemas y para integrar de forma realista tanto la dinámica del buque como las condiciones meteorológicas.

3.1. Planteamiento y estrategia inicial

El primer paso en el diseño del algoritmo consistió en definir y modelar el problema de forma conceptual. A diferencia de los sistemas de enrutamiento convencionales, donde el objetivo suele ser minimizar la distancia, para un buque propulsado exclusivamente por velas, el factor crítico en este caso es el tiempo de viaje. Dado que no existe un consumo de combustible asociado a la propulsión, la distancia recorrida se convierte en una variable secundaria frente a la duración total de la travesía.

Con este objetivo en mente, se propuso modelar el viaje como la expansión de un árbol de búsqueda. La idea fundamental es la siguiente.

El viaje se inicia en un nodo raíz, que representa la posición y el instante de partida, a partir de este nodo, se simula el paso del tiempo en intervalos discretos de una hora. Esta granularidad se eligió para que coincidiera directamente con la resolución temporal de la base de datos de viento MERRA-2. En cada hora, se expande un nodo, lo que implica calcular un conjunto de nuevas posiciones posibles (nodos hijos) a las que el buque podría llegar en la siguiente hora. Este cálculo se realizaría utilizando la siguiente información: el diagrama polar de velocidad del buque en función del viento y de su ángulo de incidencia y los datos de viento reales para esa ubicación y momento. El proceso se repetiría de forma iterativa, generando un árbol de posibles rutas que se expandiría hora a hora, hasta que alguna de sus ramas alcanzara la zona de destino.

Desde el inicio del planteamiento, se identificó que el principal desafío sería la gestión del tamaño de este árbol de búsqueda. Una expansión sin control de todas las posibilidades daría lugar a un número de nodos inmanejable. Por ello, se plantearon teóricamente varias estrategias de poda para reducir el espacio de búsqueda y descartar las rutas menos prometedoras de forma temprana:

- Poda por Restricción Geográfica: Se planteó la eliminación de cualquier nodo hijo cuya posición se encontrara sobre tierra.

- Poda por Dirección: Se consideró la posibilidad de descartar aquellos nodos que representaran un movimiento en una dirección opuesta al destino. La lógica era evitar que el algoritmo explorase rutas que, a priori, se alejaban del objetivo.

- Poda de Nodos Rezagados: Se propuso una tercera poda consistente en eliminar, tras un cierto número de horas de viaje, los nodos que se encontraran más alejados del progreso general de la búsqueda, es decir, los caminos que se estuvieran quedando atrás.

Aunque estas ideas de poda parecían razonables a nivel conceptual para guiar al algoritmo y optimizar la búsqueda, verificaría más adelante su utilidad. Como se verá más adelante, el coste computacional, tanto en tiempo de ejecución como en consumo de memoria, se convirtió rápidamente en el factor limitante y el principal condicionante en el diseño final del algoritmo.

3.2. Conceptos iniciales del prototipo del algoritmo

En este apartado se detalla el proceso evolutivo seguido para el desarrollo del algoritmo de *weather routing*. Se expondrá de forma cronológica el razonamiento detrás de cada decisión de diseño, partiendo de los conceptos fundamentales y las primeras implementaciones. Asimismo, se analizarán tanto las estrategias que resultaron exitosas como aquellas que, a pesar de ser prometedoras a nivel teórico, fueron descartadas por generar cuellos de botella computacionales. Finalmente, se explicará cómo se integraron los datos de viento y los diagramas polares del buque para dotar al modelo de realismo.

3.2.1. Generación de nodos: DFS y BFS

El objetivo de esta fase inicial era validar la lógica fundamental de la expansión de nodos en un árbol de búsqueda antes de introducir complejidades como la ortodrómica o el diagrama polar del buque o los datos reales de viento.

El diseño inicial se basó en un algoritmo de búsqueda por profundidad (DFS, Depth-First Search). Partiendo de un nodo inicial, el sistema generaba nuevos nodos hijos con ángulos y

distancias aleatorias, simulando de forma abstracta el avance del buque. La estrategia del DFS consiste en explorar una rama del árbol de búsqueda tan profundamente como sea posible antes de retroceder para explorar nuevas alternativas, como se puede apreciar en la Figura 20

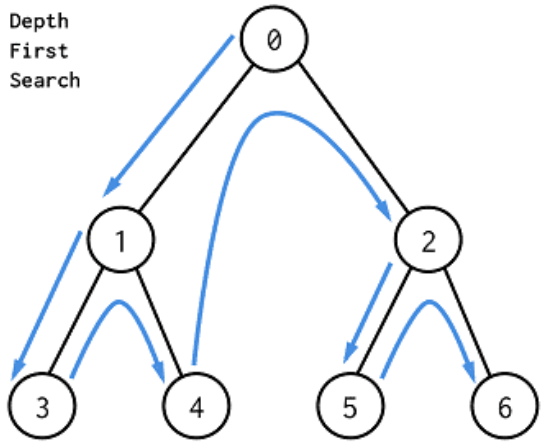

Figura 20: Esquema de funcionamiento de un algoritmo DFS

Se constató rápidamente que esta estrategia era inadecuada para el objetivo de encontrar la ruta de menor tiempo. Al sumergirse por completo en una rama antes de explorar las demás, el primer nodo en alcanzar la zona de destino no representaba la ruta más rápida, sino simplemente la primera solución que el algoritmo encontraba en su recorrido. El resultado era, en esencia, arbitrario y no garantizaba la optimalidad del tiempo de viaje.

Para resolver la deficiencia del DFS, se adoptó un algoritmo de Búsqueda en Anchura (BFS, Breadth-First Search). Esta metodología explora el árbol de manera muy distinta, por niveles. Primero, evalúa todos los nodos que se encuentran a un paso (una hora) del origen. Una vez completado ese nivel, procede a evaluar todos los nodos del siguiente nivel (dos horas de viaje), y así sucesivamente.

A diferencia del DFS, la Figura 21 representa el recorrido BFS. Se aprecia una exploración por capas: primero se visita el nodo raíz (0), luego todos los nodos del nivel 1 (nodos 1 y 2), y después se comienza a explorar el nivel 2 (nodos 3, 4, 5 y 6).

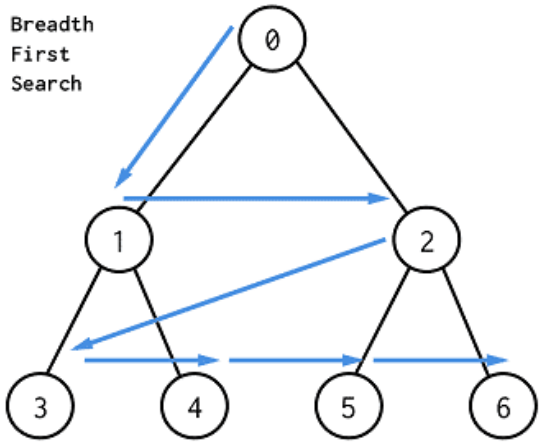

Figura 21: Esquema de funcionamiento de un algoritmo BFS

Esta propiedad del BFS valida su elección frente al DFS como una estrategia de búsqueda más correcta para el problema. Si bien, como se detallará más adelante, la versión final del algoritmo incorporará optimizaciones adicionales y no será un BFS puro, este prototipo sirvió para establecer que la búsqueda por niveles garantiza alcanzar la solución óptima.

En la Figura 22 se puede observar una representación gráfica del resultado de ejecutar el prototipo del algoritmo BFS hasta una profundidad de 5 niveles. La imagen representa el árbol de búsqueda completamente expandido, donde cada nivel representa una hora de viaje, mostrando la densa red de nodos hijos generados a partir del punto de partida central. Como se evidencia en el gráfico, a pesar de haber explorado exhaustivamente todas las rutas posibles durante 5 horas, ninguno de los nodos finales ha logrado alcanzar la zona de llegada, de color rojo. Esta figura representa claramente la inmensa cantidad de nodos generados solamente en 5 niveles.

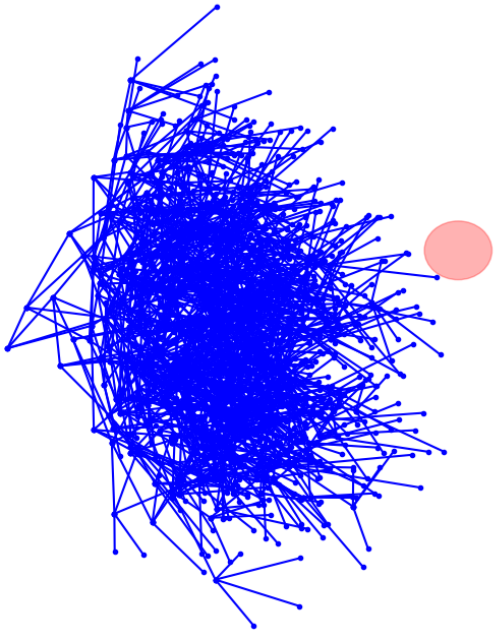

Figura 22: Árbol de búsqueda BFS de 5 niveles

La generación de esta representación gráfica, con una profundidad de 5 niveles, requirió alrededor de 10 segundos de tiempo de cómputo, un valor razonable. Sin embargo, se comprobó que para árboles con una profundidad superior a 10 niveles, el tiempo necesario para la obtención de todos los nodos aumenta de forma exponencial. Este comportamiento es una consecuencia directa propia de los algoritmos como el BFS, lo que confirma que su aplicación directa es computacionalmente imposible para travesías que requieren cientos de niveles de profundidad.

3.2.1.1. Diferencias fundamentales de programación

A nivel de programación, la diferencia fundamental entre la Búsqueda en Profundidad (DFS) y la Búsqueda en Anchura (BFS) reside en la estructura de datos que se utiliza para gestionar los nodos pendientes de explorar. Esta elección es la que define por completo el comportamiento de cada algoritmo:

- La búsqueda BFS se implementa utilizando una cola (*Queue*), una estructura de datos de tipo FIFO (*First-In, First-Out*, primero en entrar, primero en salir). Cuando

se generan nuevos nodos hijos, estos se añaden al final de la cola. El algoritmo siempre extrae el siguiente nodo a explorar del principio de la cola. Este mecanismo es el que obliga al algoritmo a explorar todos los nodos de un mismo nivel antes de pasar al siguiente.

- La búsqueda DFS utiliza una pila (*Stack*), una estructura de tipo LIFO (*Last-In, First-Out*, último en entrar, primero en salir). Los nuevos nodos hijos se apilan y el algoritmo siempre toma el último que se añadió.

Por lo tanto, es esta distinción en la estructura de datos (cola frente a pila) la que da lugar a los patrones de exploración diferentes que se han visualizado, y la que convierte al BFS en la herramienta adecuada para garantizar el resultado más óptimo para este problema.

3.2.2. Estructura de datos y reconstrucción de la trayectoria

Un aspecto fundamental en el diseño de este algoritmo de búsqueda basado en árboles es la estructura de datos del nodo. Cada nodo debe almacenar la información mínima necesaria para que el algoritmo funcione, pero también debe permitir la reconstrucción de la trayectoria final una vez se alcanza el destino. La eficiencia con la que se gestione esta información es muy importante, ya que un enfoque ineficiente puede provocar que el consumo de memoria crezca de forma descontrolada, limitando la profundidad y la viabilidad de la búsqueda.

Almacenamiento de la trayectoria completa

Una vez establecido el modelo de búsqueda en árbol, el siguiente desafío de diseño fue determinar cómo almacenar la información en cada nodo. Un nodo no solo debe contener su estado actual (posición y tiempo), sino que también debe facilitar la reconstrucción de la ruta completa una vez se alcance el destino. La solución más directa e intuitiva para este problema, y la que se adoptó en la fase inicial de desarrollo, fue que cada nodo almacenara la trayectoria completa que lo precedía.

En este enfoque, cada nodo se concibe como un objeto de datos que contiene, además de su propia posición y el tiempo de viaje acumulado, una lista ordenada con las coordenadas de todos los nodos predecesores, desde el punto de partida hasta su padre directo. El mecanismo de expansión, por tanto, funciona de la siguiente manera: al generar un nodo "hijo" a partir de un "padre", el nuevo nodo primero recibe una copia de la lista de trayectoria completa del padre y, a continuación, añade sus propias coordenadas al final de esa lista.

La principal ventaja de esta metodología es su simplicidad en la reconstrucción del camino. Cuando el algoritmo determina que un nodo ha alcanzado la zona de destino, la ruta óptima ya está contenida íntegramente dentro de ese nodo final. No se necesita ningún algoritmo de retroceso (backtracking) ni procesamiento adicional; la solución es inmediata.

En la Figura 23 se puede apreciar cómo era la estructura inicial de almacenamiento de datos para cada nodo, siendo los nodos de mayor nivel los que almacenan mayor número de coordenadas y por lo tanto los más costosos en términos de memoria.

```
--- Nodo Explorado ---
  Coordenadas (x, y): (5.8, -0.32)
  Tiempo acumulado hasta este nodo: 5.8
  Trayectoria completa: [(0, 0), (5.8, -0.32)]
----------------------
--- Nodo Explorado ---
  Coordenadas (x, y): (9.43, -3.08)
  Tiempo acumulado hasta este nodo: 10.37
  Trayectoria completa: [(0, 0), (5.8, -0.32), (9.43, -3.08)]
----------------------
--- Nodo Explorado ---
  Coordenadas (x, y): (9.86, -8.99)
  Tiempo acumulado hasta este nodo: 16.29
  Trayectoria completa: [(0, 0), (5.8, -0.32), (9.43, -3.08), (9.86, -8.99)]
----------------------
--- Nodo Explorado ---
  Coordenadas (x, y): (10.6, -0.04)
  Tiempo acumulado hasta este nodo: 25.27
  Trayectoria completa: [(0, 0), (5.8, -0.32), (9.43, -3.08), (9.86, -8.99), (10.6, -0.04)]
----------------------
--- Nodo Explorado ---
  Coordenadas (x, y): (19.15, -4.29)
  Tiempo acumulado hasta este nodo: 34.82
  Trayectoria completa: [(0, 0), (5.8, -0.32), (9.43, -3.08), (9.86, -8.99), (10.6, -0.04), (19.15, -4.29)]
----------------------
```

Figura 23: Almacenamiento de trayectorias inicial

Ineficiencia de la memoria

Aunque el enfoque de almacenar la trayectoria completa en cada nodo es sencillo, su aplicación práctica se encuentra con un obstáculo a medida que aumenta la escala del problema, ya que hay un consumo de memoria ineficiente que crece de forma insostenible. Para cuantificar este efecto, se desarrolló un modelo simplificado que compara el uso de memoria de las dos estrategias de almacenamiento de datos discutidas.

En este modelo, se simula el crecimiento de un árbol de búsqueda con un factor de ramificación (b) de 4, es decir, cada nodo genera cuatro nuevos hijos. Se analiza el uso de memoria (en unidades relativas) a medida que aumenta la profundidad (d) del árbol. Se comparan dos casos:

- Estrategia 1 (Trayectoria Completa): La memoria de un nodo a profundidad *d* es proporcional a *d+1* (para almacenar su propia información y la de todos sus predecesores).

- Estrategia 2 (Enlace al Padre): La memoria de cada nodo es constante, ya que solo almacena su propia información y una referencia a su padre (modelado como 2 unidades).

En la Figura 24 se muestra los resultados de este modelo en una escala lineal, lo que permite apreciar la magnitud real de la diferencia entre ambas estrategias.

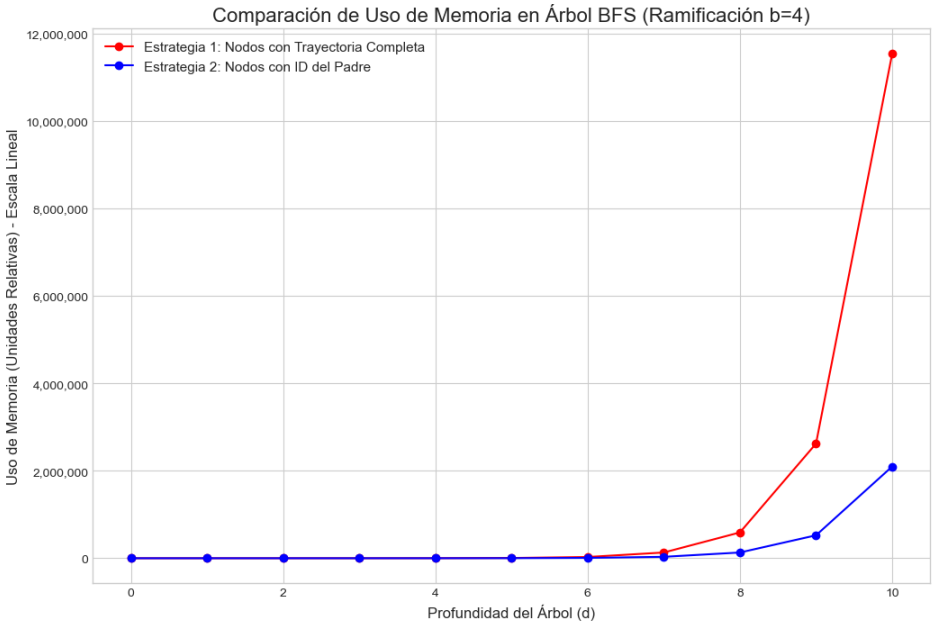

Figura 24: Comparación de uso de memoria en Árbol de nodos

Como se observa en la gráfica, la diferencia es drástica y se manifiesta cada vez más a profundidades mayores. Mientras que la memoria para la estrategia del enlace al padre (línea azul) crece de forma exponencial (un comportamiento esperado, ya que el número de nodos es b^d), la memoria para la estrategia de la trayectoria completa (línea roja) crece a un ritmo mayor. Esto se debe a que no solo aumenta el número de nodos de forma exponencial, sino que el tamaño de cada nodo individual también aumenta linealmente con la profundidad.

Esta simulación demuestra de forma cuantitativa que la estrategia de almacenar la trayectoria completa se vuelve computacionalmente inmanejable a profundidades reducidas, haciendo inviable su uso para simular una travesía de cientos de horas. Por lo tanto, se confirma que la única estrategia factible es aquella que utiliza una estructura de nodo de tamaño constante, como la del enlace al padre.

Enlace al nodo padre

Para resolver el problema de escalabilidad de memoria del apartado anterior, se implementó una estructura de datos más eficiente. Este enfoque abandona la idea de almacenar la trayectoria completa en cada nodo y, en su lugar, utiliza un sistema de referencias que reduce drásticamente los requisitos de memoria, permitiendo que el algoritmo explore árboles de búsqueda mucho más grandes y profundos.

La clave de esta solución se basa en cambiar la información que contiene cada nodo. En esta estructura optimizada, cada nodo del árbol almacena únicamente dos elementos esenciales:

- Sus propios datos: Un conjunto de información que define el estado actual del buque, como su posición geográfica (latitud y longitud) y el tiempo de viaje acumulado hasta ese punto.

- Una única referencia a su nodo padre: En lugar de una lista con toda la historia, se guarda una sola referencia o "puntero" que apunta al nodo del cual este fue generado.

Con este diseño, el espacio de memoria requerido por cada nodo es constante y reducido, sin importar si se encuentra en el nivel 2 o en el nivel 700 del árbol de búsqueda. La información

que define al nodo no crece con la profundidad, lo que resuelve el problema del crecimiento no lineal de la memoria.

La adopción de esta estructura de datos tiene un impacto positivo en la viabilidad del algoritmo, aunque se introduce un pequeño coste computacional adicional para la reconstrucción final de la ruta, este es insignificante en comparación con los ahorros de memoria obtenidos.

Reconstrucción Inversa (Backtracking)

La ruta final no está disponible de forma inmediata. Una vez que el algoritmo de búsqueda (por ejemplo, el BFS) encuentra un nodo que ha llegado a la zona de destino, se debe realizar un proceso de reconstrucción inversa o *backtracking* para obtener la trayectoria completa.

Este proceso es muy rápido y se realiza de la siguiente manera:

1. Se comienza con el nodo final que ha alcanzado el destino.
2. Se añade la posición de este nodo a una lista que contendrá la ruta final.
3. A continuación, se accede a su referencia "padre" para identificar el nodo predecesor en la ruta.
4. Se repite el proceso: se añade la posición del padre a la lista de la ruta y se salta a su respectivo padre.
5. Esta cadena de saltos hacia atrás continúa, nodo por nodo, hasta que se alcanza el nodo raíz (cuyo padre es nulo), que es el punto de partida de la travesía.

Al finalizar este recorrido inverso, se obtiene la lista completa de coordenadas que conforman la ruta óptima, ordenada desde el origen hasta el destino, así como la trayectoria reconstruida en el gráfico, tal como se aprecia en la Figura 25

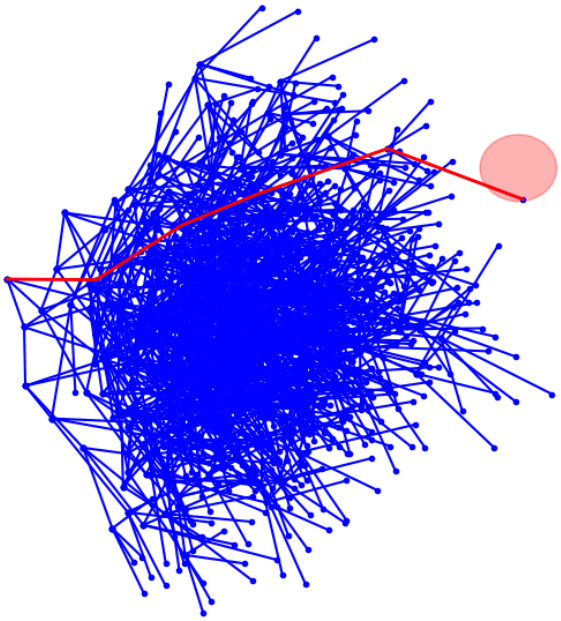

Figura 25: Trayectoria reconstruida por algorítmo BFS

3.3. Implementación del movimiento geográfico

El primer paso para dotar de realismo al prototipo inicial fue abandonar la generación de nodos sin sentido y situar el problema sobre la superficie terrestre. Esta transición de un espacio

abstracto a uno geográfico implicó la definición de un sistema de coordenadas real y la implementación de un modelo de movimiento sobre una esfera.

Sistema de coordenadas en la esfera terrestre

Para representar la posición del buque, se adoptó el sistema de coordenadas geográficas estándar de latitud y longitud, las cuales se pueden ver representadas en la Figura 26. Todos los nodos del árbol de búsqueda pasarían a representar un punto específico sobre el globo. Para los cálculos de navegación, se parte de una suposición fundamental y ampliamente aceptada en este tipo de problemas: se modela la Tierra como una esfera perfecta. Aunque en realidad es una esfera achatada, para la escala de una travesía transoceánica, esta aproximación simplifica los cálculos trigonométricos sin introducir un error significativo, permitiendo determinar los arcos de esfera que recorre el buque.

En el contexto de este estudio, se definieron los siguientes puntos para la travesía principal:

- Punto de Partida: Hampton Roads, EE. UU. (aproximadamente 36.9° N, 76.3° O).
- Zona de Llegada: Róterdam, Países Bajos (definida por un área alrededor de 51.9° N, 4.5° E).

Navegación Ortodrómica

Moverse sobre una superficie esférica no es lo mismo que sobre un plano. La distancia más corta entre dos puntos es un arco de círculo máximo, conocido como ruta ortodrómica (great-circle path). El algoritmo debe simular este tipo de movimiento.

En cada paso de una hora, se calcula la nueva posición del buque a partir de su punto actual, un rumbo y la distancia recorrida. Para ello, se utilizan las siguientes fórmulas de navegación geodésica, derivadas de la trigonometría esférica:

La nueva latitud (φ_2) se calcula como:

$$\varphi_2 = arcsin(sin(\varphi_1) \cdot cos\left(\frac{d}{R}\right) + cos(\varphi_1) \cdot sin\left(\frac{d}{R}\right) \cdot cos(\theta))$$

Y la nueva longitud (λ_2) como:

$$\lambda_2 = \lambda_1 + atan2(sin(\theta) \cdot sin\left(\frac{d}{R}\right) \cdot cos(\varphi_1), cos\left(\frac{d}{R}\right) - sin(\varphi_1) \cdot sin(\varphi_2))$$

Donde:

- φ_1 y λ_1 son la latitud y longitud iniciales en radianes.
- d es la distancia recorrida sobre el arco de círculo máximo.
- R es el radio de la Tierra (asumido como 6371 km).
- θ es el rumbo (acimut) en radianes.
- $atan2$ es la función arcotangente de dos argumentos.

De esta forma, cada pequeño segmento de una hora de la ruta generada por el algoritmo es un arco ortodrómico que representa de forma precisa el desplazamiento del buque sobre la superficie terrestre.

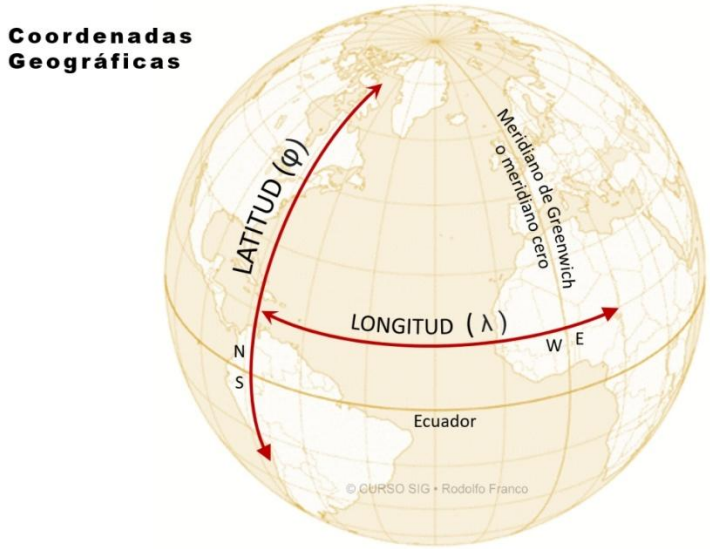

Figura 26: Latitud y longitud en el globo terráqueo (Franco, 2020)

3.4. Integración de datos de viento reales y diagramas polares

Una vez establecido el marco geográfico, el siguiente paso fue reemplazar el modelo de movimiento aleatorio por un motor físico realista. Para lograrlo, fue necesario integrar dos fuentes de información clave, los datos meteorológicos de la base de datos MERRA-2 y los diagramas polares que definen el rendimiento del buque. El objetivo de esta fase era que cada paso de la simulación representara un desplazamiento lo más físicamente coherente.

Aplicación de los Datos de Viento

La integración de los datos de viento se realiza de forma directa. Para cada nodo que se expande en el árbol de búsqueda, el algoritmo utiliza la información de estado del nodo (su latitud, longitud y el tiempo de viaje acumulado) para realizar una consulta a la base de datos local de MERRA-2. El proceso es el siguiente:

1. La fecha y hora del nodo se utilizan para identificar y abrir el archivo .nc4 diario correspondiente.
2. Las coordenadas del nodo se usan para localizar el punto más cercano en la malla de datos del archivo.
3. Se extraen las magnitudes de las componentes de viento U (este-oeste) y V (norte-sur) para la hora específica.

Este proceso, que se repite para cada nodo explorado, asegura que toda la simulación se base en condiciones de viento históricas y realistas.

Generación de Rutas a partir de los Diagramas Polares

La información de viento por sí sola no es suficiente; es necesario pasarla a un desplazamiento del buque. De esto se encarga la función principal de generación de rutas, que actúa como el motor físico del algoritmo. Para cada nodo, esta función establece cinco rumbos potenciales iniciales (a 30°, 60°, 90°, 120° y 150° respecto a la proa). A continuación, para cada uno de estos rumbos, se ejecuta un cálculo para determinar la distancia que el buque recorrería en una hora:

1. Se calcula el ángulo de incidencia real del viento con respecto al rumbo propuesto.

2. Con la velocidad real del viento (asegurando la consistencia de unidades, pasando de m/s a nudos cuando es necesario) y el ángulo de incidencia, se consultan los diagramas polares del buque. Dado que la velocidad del viento real rara vez coincidirá exactamente con las curvas de 5, 10, 15 nudos, etc., se realiza una interpolación lineal entre las dos curvas más cercanas para obtener la velocidad precisa del buque.
3. La velocidad resultante (en m/s) se convierte en la distancia en kilómetros que el buque puede recorrer en el intervalo de una hora de la simulación.

Esta distancia es la que finalmente se utiliza en las fórmulas de navegación ortodrómica para calcular la posición del nuevo nodo hijo.

Corrección de Rumbo en Ceñida

Finalmente, se dotó a esta función de una inteligencia de navegación para gestionar las situaciones en las que el viento viene de proa, una zona donde las velas no pueden generar sustentación (la zona de no avance o "*no-go zone*", típicamente entre 0° y 30° a cada lado del viento como se puede apreciar en Figura 27).

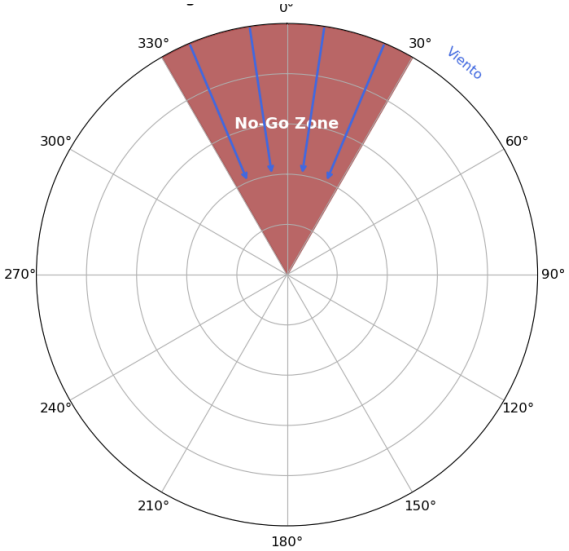

Figura 27: Esquema de la No-Go Zone

Para evitar que la búsqueda se detenga en estas condiciones, se implementó una lógica de corrección. Si uno de los rumbos propuestos cae dentro de esta zona de no avance, el algoritmo no lo descarta. En su lugar, ajusta automáticamente dicho rumbo al ángulo mínimo posible en el que el buque todavía puede navegar en ceñida (es decir, al borde de esa zona de 30°). Esto permite al algoritmo simular la maniobra de virada por avante (*tacking*), posibilitando que el buque avance de forma efectiva contra el viento, un comportamiento crucial para cualquier simulación de navegación a vela realista.

3.5. Optimización y análisis de cuellos de botella

Una vez que el algoritmo fue capaz de generar rutas realistas en un entorno geográfico el problema de la generación exponencial de nodos se convirtió en el principal obstáculo. El número de nodos generados por el BFS crecía de forma tan descontrolada que hacía imposible simular más de unas pocas horas de viaje en un tiempo razonable. La solución pasaba por implementar estrategias de poda para descartar nodos inviables de forma temprana.

Estrategias de Poda Iniciales

Los primeros intentos de optimización se basaron en el uso de librerías geoespaciales para crear y validar las restricciones geográficas. Específicamente, se emplearon *Shapely*, una biblioteca de *Python* para la manipulación y el análisis de objetos geométricos, y *GeoPandas*, para trabajar con datos espaciales. *Shapely* permite la creación de geometrías como puntos, líneas y polígonos, así como la ejecución de operaciones complejas como la creación de buffers o la comprobación de si un punto está contenido dentro de un polígono (*contains*). Por otra parte, *GeoPandas* facilita la importación de cartografía (como los contornos de los continentes) y la gestión de estas geometrías en un formato de tabla. La combinación de ambas herramientas permitió implementar dos comprobaciones para cada nodo generado:

- Validación del Corredor de Navegación: Se definió una ruta central y se generó un buffer a su alrededor, creando un objeto de tipo *Polygon* que representaba el pasillo marítimo válido. La función *en_pasillo* realizaba una operación geométrica para comprobar si cada nuevo nodo estaba contenido dentro de este polígono. Se puede apreciar en la Figura 28 el corredor de navegación como una franja azul que conecta el continente americano con el continente europeo.

- Validación de la zona de llegada: a su vez se definió una zona de llegada mediante un buffer circular que funcionaba de forma exactamente igual que la validación del corredor de navegación. La Figura 28 presenta un círculo rojo central que concuerda con la zona de llegada establecida para el prototipo.

- Comprobación de Tierra (*es_tierra*): De forma similar, se utilizó una función que, para cada nodo, comprobaba si sus coordenadas estaban contenidas dentro de alguno de los polígonos que representan los continentes del planeta.

Aunque estas soluciones eran precisas, su impacto en el rendimiento del algoritmo fue muy negativo.

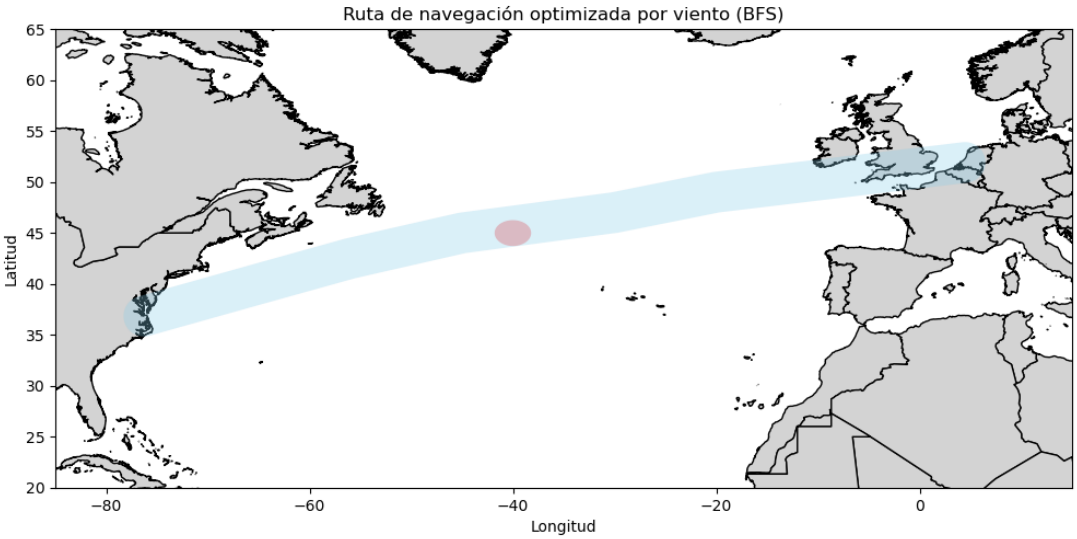

Figura 28: Corredor de navegación y zona de llegada mediante buffers geométricos

Análisis de Rendimiento e Identificación de Cuellos de Botella

Al ejecutar el prototipo con estas podas, el tiempo de computación era excesivamente alto. Para diagnosticar la causa de forma precisa y objetiva, se utilizaron herramientas de análisis de rendimiento (*profilers*). Un *profiler* es un programa que mide el tiempo de ejecución que se consume en cada una de las funciones de un código, permitiendo identificar los "cuellos de botella" que ralentizan el sistema.

La Figura 29 muestra el resultado de analizar una ejecución del algoritmo con la herramienta *%%prun -l 30* lo cual muestra .30 lineas de código con el mayor número de llamadas o tiempo de ejecución.

```
         2945123 function calls (2911105 primitive calls) in 13.576 seconds

   Ordered by: internal time
   List reduced from 5304 to 30 due to restriction <30>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     1194    8.326    0.007    8.326    0.007 predicates.py:472(contains)
       12    0.861    0.072    0.861    0.072 collections.py:352(draw)
      2/0    0.453    0.226    0.000          backend_agg.py:185(draw_text)
      614    0.294    0.000    0.294    0.000 {built-in method _io.open_code}
       88    0.169    0.002    0.177    0.002 {built-in method _imp.create_dynamic}
      614    0.118    0.000    0.118    0.000 {built-in method marshal.loads}
    28675    0.082    0.000    8.653    0.000 decorators.py:62(wrapped)
     4276    0.080    0.000    0.151    0.000 path.py:319(make_compound_path)
     7554    0.075    0.000    0.146    0.000 inspect.py:882(cleandoc)
    12484    0.071    0.000    0.071    0.000 transforms.py:110(__init__)
      614    0.070    0.000    0.070    0.000 {method 'read' of '_io.BufferedReader' objects}
        1    0.069    0.069    0.069    0.069 raw.py:39(read)
     2691    0.063    0.000    0.063    0.000 {built-in method nt.stat}
     1127    0.054    0.000    0.058    0.000 functools.py:35(update_wrapper)
     4276    0.049    0.000    0.049    0.000 _geometry.py:403(get_exterior_ring)
208804/208745    0.045    0.000    0.058    0.000 {built-in method builtins.isinstance}
   688/687    0.041    0.000    0.123    0.000 series.py:389(__init__)
     4167    0.039    0.000    0.039    0.000 _geometry.py:506(get_geometry)
    75236    0.039    0.000    0.084    0.000 __init__.py:778(__getitem__)
```

Figura 29: Resultado de ejecutar %%prun -l 30 en el prototipo

El análisis de la Figura 29 ofrece información muy relevante. De un tiempo total de ejecución de 13.576 segundos, una única función interna de la librería *shapely*, predicates.py:472(*contains*), consumió 8.326 segundos. Esto significa que esta simple operación de comprobación fue responsable de más del 60% del tiempo total de computación.

Esta función *contains()* es la que utilizan internamente tanto la comprobación del pasillo (*en_pasillo*) como la de tierra (*es_tierra*) y como la de zona de llegada. La razón de su alto coste es la complejidad matemática de una operación "punto en polígono", especialmente cuando los polígonos tienen miles de vértices, como una línea de costa. Cuando esta operación tan costosa se ejecuta millones de veces (una por cada nodo generado en el árbol de búsqueda), se convierte en un cuello de botella que ralentiza de gran manera el algoritmo.

Esta fase de análisis fue un punto de inflexión en el desarrollo. Proporcionó evidencia de que *GeoPandas* y *Shapely*, era inviable. Esto obligó a rediseñar la estrategia de poda, buscando métodos matemáticamente más simples y a encontrar una forma de reducir drásticamente el número total de nodos a evaluar.

Simplificación de las restricciones geométricas

El análisis de rendimiento aclaró que la viabilidad del algoritmo dependía de sustituir las operaciones geoespaciales por alternativas computacionalmente más ligeras. La solución, por tanto, fue rediseñar la forma en que se definían y validaban las restricciones geográficas, sustituyendo los objetos *Polygon* por estructuras matemáticas mucho más simples.

- Redefinición del corredor de navegación: Se abandonó el uso de buffer para el pasillo. En su lugar, el corredor de navegación pasó a definirse mediante dos líneas poligonales simples: una que representa el límite norte y otra el límite sur. La función de validación ahora no comprueba si un punto está contenido en un área, sino que realiza una serie de comparaciones matemáticas mucho más rápidas para determinar si el punto se encuentra por debajo de la línea poligonal superior y por encima de la línea poligonal inferior. Este cálculo es órdenes de magnitud más eficiente que una operación *contains*.

- Simplificación de la zona de llegada: De forma análoga, se descartó el buffer circular para la zona de destino. La nueva implementación define la zona de llegada como un simple rectángulo geográfico, delimitado por unas coordenadas de latitud y longitud mínimas y máximas. La comprobación para saber si un nodo ha alcanzado el destino se reduce a cuatro comparaciones numéricas directas, una operación que es casi instantánea.

- Exclusión de zonas críticas (Islas): Para refinar el espacio de búsqueda y evitar obstáculos que pudieran encontrarse dentro del corredor principal (como por ejemplo, el archipiélago de las Azores), se implementó una capa adicional de poda llamada en_zona_critica. Esta estrategia consiste en definir una lista de pequeños rectángulos geográficos que encuadran estas islas o zonas a evitar. Antes de validar un nuevo nodo, el algoritmo comprueba de forma muy rápida si sus coordenadas caen dentro de alguno de estas zonas prohibidas. Si es así, el nodo es descartado. Este método permite excluir con gran precisión zonas de no navegación sin recurrir al coste computacional de analizar un mapa de costas detallado.

Este rediseño, el cual se puede apreciar en la Figura 30 motivado directamente por los resultados del análisis de cuellos de botella, fue un paso crucial. Al sustituir las complejas y lentas funciones de las librerías geoespaciales por operaciones matemáticas sencillas, se eliminó el principal factor que limitaba el rendimiento del algoritmo. Esto permitió que la ejecución fuera mucho más rápida y, sobre todo, que el sistema pudiera escalar para explorar árboles de búsqueda más profundos sin que el tiempo de computación se volviera prohibitivo.

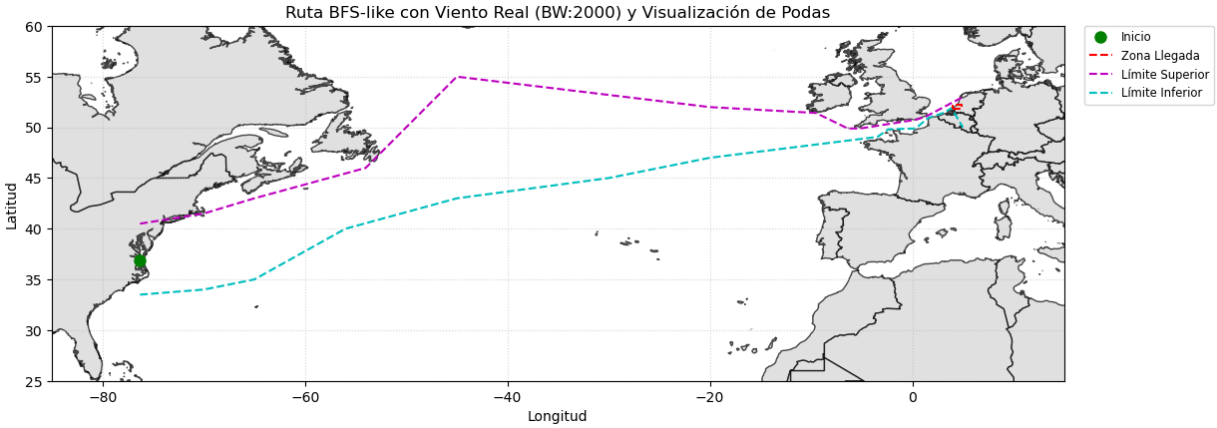

Figura 30:Corredor de navegación y zona de llegada simplificados

Estrategia de podas descartadas

Durante el desarrollo, se probaron varias estrategias de poda que, aunque lógicas, resultaron ser demasiado rígidas o ineficientes y no se incluyeron en la versión final.

Una de ellas fue la poda de nodos en retroceso, que eliminaba cualquier nodo que se moviera en una longitud opuesta a la dirección general del viaje. Otra fue la poda periódica de nodos rezagados, que cada ciertas horas descartaba las rutas que se hubieran quedado más atrás del nodo más adelantado.

Finalmente, se concluyó que su funcionalidad era absorbida de forma más flexible e inteligente por el sistema de Búsqueda en Haz, por lo que estas reglas explícitas fueron eliminadas.

Poda de celdas visitadas

Otra estrategia de poda que se implementó y probó fue la de celdas visitadas. Para evitar que el algoritmo entrara en bucles o reexplorara innecesariamente las mismas zonas geográficas, se discretizó el océano en una cuadrícula. La resolución de esta cuadrícula se definía truncando las coordenadas de latitud y longitud a un número determinado de decimales. Cuando un nodo era explorado, su celda correspondiente se marcaba como "visitada", y cualquier nodo futuro que cayera en esa misma celda era descartado.

Aunque esta técnica es muy eficaz para reducir el espacio de búsqueda, se descubrió que era contraproducente en situaciones meteorológicas complejas. En zonas de calma o durante una tormenta, es realista que un buque apenas avance o incluso se vea forzado a retroceder temporalmente, permaneciendo en la misma área geográfica durante varias horas. El problema era que el algoritmo, al generar nuevos nodos (que representaban un instante de tiempo posterior) en una celda ya marcada como "visitada", los eliminaba incorrectamente. Si el buque permanecía atrapado en una de estas zonas durante un tiempo prolongado, esta poda llegaba a eliminar todos los nodos activos de la frontera de búsqueda, provocando que el algoritmo terminara de forma prematura sin haber encontrado una solución.

3.6. Inteligencia del algoritmo

Las fases anteriores del desarrollo demostraron que, si bien el algoritmo ya era capaz de simular rutas realistas, se enfrentaba a un problema en términos computacionales. El método de Búsqueda en Anchura (BFS), al explorar todas las posibilidades por igual, generaba una nube de nodos que crecía de forma exponencial. El rendimiento observado era insuficiente: una simulación de 5 niveles tardaba apenas unos segundos, pero al aumentar la profundidad a 10 niveles, el tiempo se disparaba a más de 10 minutos. Para 20 niveles, la ejecución podía superar la media hora o incluso no terminar nunca.

Se necesitaba una solución para guiar la búsqueda, una forma de dotar al algoritmo de un sentido de la orientación que le permitiera centrar sus esfuerzos en las rutas más prometedoras y descartar las que inicialmente no llevaban a ninguna parte. Esta inteligencia se implementó a través de una función heurística.

El Concepto de Heurística

Una heurística no es más que una regla que ayuda a resolver un problema de forma más rápida cuando el método exhaustivo es demasiado lento. No garantiza encontrar la mejor solución posible, pero sí una solución suficientemente buena en un tiempo razonable.

Un algoritmo de búsqueda muy conocido que utiliza este principio es *A* ("*A estrella*")*. A* combina el BFS (que siempre busca el camino más corto recorrido hasta ahora) con una función heurística que estima la distancia que queda hasta el destino. De esta forma, prioriza los nodos que no solo han recorrido un camino corto, sino que también *parecen* estar más cerca de la meta.

Aunque nuestro algoritmo final se inspira en esta idea de usar una guía, no es una implementación de A*. A* sigue siendo un algoritmo que puede llegar a explorar una gran cantidad de nodos para garantizar que encuentra la ruta óptima. Dada la escala de nuestro problema, se necesitaba una solución aún más agresiva, una que no solo guiara la búsqueda, sino que la podara notablemente.

Búsqueda en Haz (*Beam Search*)

La Búsqueda en Haz, o *Beam Search*, es un algoritmo de búsqueda heurística que se sitúa en un punto intermedio entre una búsqueda exhaustiva como el BFS y una búsqueda guiada. Su objetivo es reducir drásticamente el espacio de búsqueda para hacerlo computacionalmente manejable, sin perder la capacidad de encontrar una solución de alta calidad.

El mecanismo es sencillo pero muy interesante. A diferencia del BFS, que conserva en memoria todos los nodos generados en cada nivel, el Beam Search, tras expandir un nivel, ordena a todos los nodos hijos según su puntuación heurística y descarta a todos excepto a un número fijo de los mejores. Este número se conoce como el ancho del haz (*Beam Width*), y es el parámetro fundamental que controla el equilibrio entre el rendimiento del algoritmo y la exhaustividad de la búsqueda. Un haz más ancho explora más posibilidades, pero consume más recursos; un haz más estrecho es más rápido, pero corre un mayor riesgo de descartar prematuramente la ruta óptima.

Este proceso de expandir, evaluar, podar se repite en cada nivel, lo que transforma el crecimiento exponencial del árbol en un crecimiento prácticamente lineal, haciendo que la búsqueda sea factible para travesías de larga duración.

Un algoritmo que solo conservara los mejores nodos en cada paso no sería la mejor opción aunque si lo pareciese ya que se centraría exclusivamente en las rutas que parecen mejores en el momento actual, lo cual es muy eficiente, sin embargo, esto conlleva un riesgo significativo, el algoritmo podría quedarse atascado. Podría encontrar una ruta que es buena a corto plazo, pero que le impide alcanzar una solución global mucho mejor (por ejemplo, siguiendo vientos moderados pero constantes, podría descartar una ruta que implica pasar por una pequeña zona de calma para luego alcanzar vientos muy favorables).

Para mitigar este riesgo, la implementación final no utiliza un Beam Search simple, sino uno diversificado. Esto significa que el haz de nodos que sobrevive en cada poda no se compone únicamente de los mejores, sino de dos grupos distintos:

- Nodos de explotación: La gran mayoría del haz está compuesta por los nodos con la mejor puntuación heurística. Su función es la de "explotar" las rutas que, según la información disponible, son las más prometedoras.

- Nodos de exploración: Una pequeña porción del haz se reserva para nodos que se seleccionan de forma aleatoria del grupo de los que, en principio, iban a ser descartados. Estos nodos cumplen una función de exploración. Aunque es muy probable que no conduzcan a una solución mejor, introducen diversidad en la búsqueda. Permiten que rutas menos obvias o que atraviesan una fase temporalmente mala sobrevivan, lo que podría ser crucial para navegar con éxito situaciones meteorológicas complejas y encontrar soluciones que un algoritmo puramente codicioso habría ignorado.

Esta combinación de explotación y exploración crea un algoritmo más robusto, capaz de encontrar soluciones de mejor calidad, aunque no completamente óptimas. Se obtienen resultados cercanos al óptimo, pero es prácticamente imposible que el obtenido sea el mejor puesto que las podas reducen bastante el número de caminos estudiados.

Diseño de la Heurística

La Búsqueda en Haz necesita una forma de saber qué nodos son los mejores. De esto se encarga la función heurística, cuyo diseño fue un proceso evolutivo:

- Distancia: La primera heurística que se probó puntuaba los nodos en función de su distancia ortodrómica al destino. Sin embargo, para un velero, la ruta más corta casi

nunca es la más rápida. Un nodo puede estar geográficamente más cerca del destino, pero encontrarse en una zona de calma total, siendo mucho menos prometedor que un nodo más lejano que navega a gran velocidad con vientos favorables.

- Velocidad Efectiva: La solución definitiva y más potente fue basar la heurística en la velocidad efectiva hacia el destino. La velocidad efectiva hacia el destino no es la velocidad real del buque, sino la componente de esa velocidad que va en la dirección del objetivo. Un nodo se considera más prometedor cuanto mayor es su velocidad hacia el destino, es decir, cuanto más rápido se está acercando a la meta. Esta métrica es una combinación perfecta de distancia, tiempo y dirección, penaliza de forma natural los nodos que van lentos o en la dirección equivocada y recompensa a aquellos que, gracias a las condiciones del viento y al rendimiento del buque, logran un avance hacia el Este en cada hora. Es esta función la que dota al algoritmo de su verdadera inteligencia de navegación.

Checkpoint dinámicos

Una vez establecida la Búsqueda en Haz y la heurística basada en la Velocidad Efectiva hacia el destino el algoritmo era ya funcional, pero todavía presentaba una debilidad en geografías complejas. Una heurística que apunta al destino final puede no detectar zonas de tierra y guiar al algoritmo hacia rutas sin salida. Por ejemplo, en la ruta de Hampton Roads a Róterdam, una heurística que solo considerase la velocidad hacia el destino final podría intentar trazar una ruta en línea recta, estrellando al buque contra la costa de Francia en lugar de guiarlo por el paso obligatorio del Canal de la Mancha.

Para resolver esta limitación, se implementó un sistema de objetivos mediante el uso de un checkpoint intermedio. Este checkpoint intermedio se coloca a la entrada del canal de la Mancha, como se puede apreciar en la Figura 31 para asegurar que lo atraviese y el haz de nodos no se neutralice contra la costa europea.

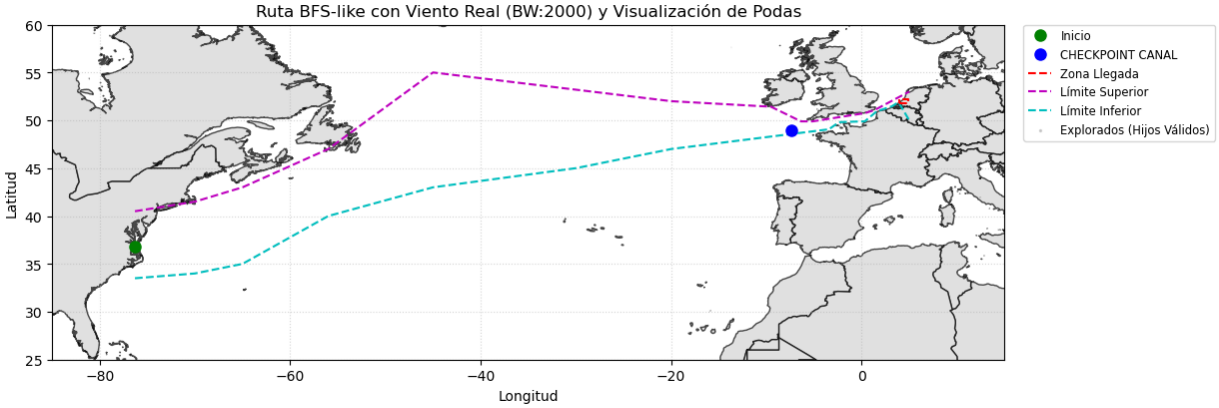

Figura 31: Gráfica con checkpoints

El mecanismo es el siguiente: en lugar de tener un único objetivo final, la función heurística opera en dos fases, cambiando su punto de referencia en función de la posición del buque:

1. Fase de aproximación al checkpoint: Para la ruta estudiada, se definió un checkpoint estratégico a la entrada occidental del Canal de la Mancha. Mientras la longitud del nodo que se está evaluando no haya superado la longitud de este checkpoint, la función heurística calcula la velocidad efectiva a destino exclusivamente hacia dicho checkpoint. Esto fuerza a todo el haz de búsqueda a priorizar las rutas que se dirigen eficazmente hacia la entrada del canal.

2. Fase de aproximación al destino final: En el momento en que un nodo supera la longitud del checkpoint, la lógica de la heurística cambia automáticamente. A partir de ese punto, la función pasa a calcular la puntuación de la heurísticahacia el destino final en Róterdam.

Este sistema de guiado en dos etapas dota al algoritmo de una capacidad estratégica. Le permite resolver un problema de navegación a gran escala dividiéndolo en sub-objetivos más sencillos y lógicos. Primero, se enfoca en la tarea de llegar a la entrada del Canal de la Mancha de la forma más rápida posible y, una vez allí, se concentra en la tarea de navegar el canal hasta el puerto.

3.7. Validación final del algoritmo

Una vez finalizado el desarrollo, el último paso fue la validación del sistema en su conjunto. El objetivo de esta fase era confirmar que el comportamiento del algoritmo era coherente y que los resultados producidos eran lógicos y realistas desde la perspectiva de la navegación, especialmente en lo que respecta a las soluciones que se implementaron.

Al igual que en fases anteriores, el principal método de validación fue el análisis visual de las rutas completas generadas. Se ejecutó el algoritmo en una serie de casos de prueba con diferentes fechas de partida y se representaron gráficamente las trayectorias resultantes sobre un mapa. Esta visualización permitió evaluar de forma cualitativa varios aspectos clave:

- Eficacia de los checkpoints: la validación confirmó de forma concluyente la importancia del sistema de checkpoints dinámicos. En ejecuciones donde se desactivaba esta funcionalidad, se observó que la heurística tendía a trazar rutas que, aunque eficientes en su avance hacia el Este, se dirigían directamente contra la costa de Inglaterra o Irlanda. La activación del checkpoint en el Canal de la Mancha corrigió este comportamiento, guiando con éxito a la totalidad del haz de búsqueda hacia el paso correcto antes de enfocarlo hacia el destino final.

- Robustez en condiciones meteorológicas adversas: una de las validaciones más importantes fue comprobar cómo se comportaba el algoritmo en zonas de calma (vientos muy flojos) o de tormenta (vientos muy fuertes), situaciones que provocaban el fallo de las estrategias de poda preliminares como la de "celdas visitadas". Se observó que el algoritmo final es robusto ante estas condiciones: es capaz de mantener el buque en una misma área durante varias horas sin que el sistema de poda elimine todos los nodos, permitiendo que la simulación espere a que mejoren las condiciones, un comportamiento más realista.

- Coherencia de la navegación: en general, se observó que las trayectorias eran fluidas y lógicas. Las maniobras se producían de forma natural en respuesta a los cambios en las condiciones meteorológicas, y las rutas respetaban siempre las restricciones geográficas impuestas por el corredor de navegación y las zonas críticas.

Esta validación visual, la cual se puede apreciar en la Figura 32 proporcionó la confianza necesaria en el correcto funcionamiento del algoritmo. Si bien no siempre genera rutas perfectas que circunnavegan las grandes tormentas, sí demuestra ser una herramienta robusta, capaz de encontrar caminos viables y de reaccionar de forma inteligente a las restricciones.

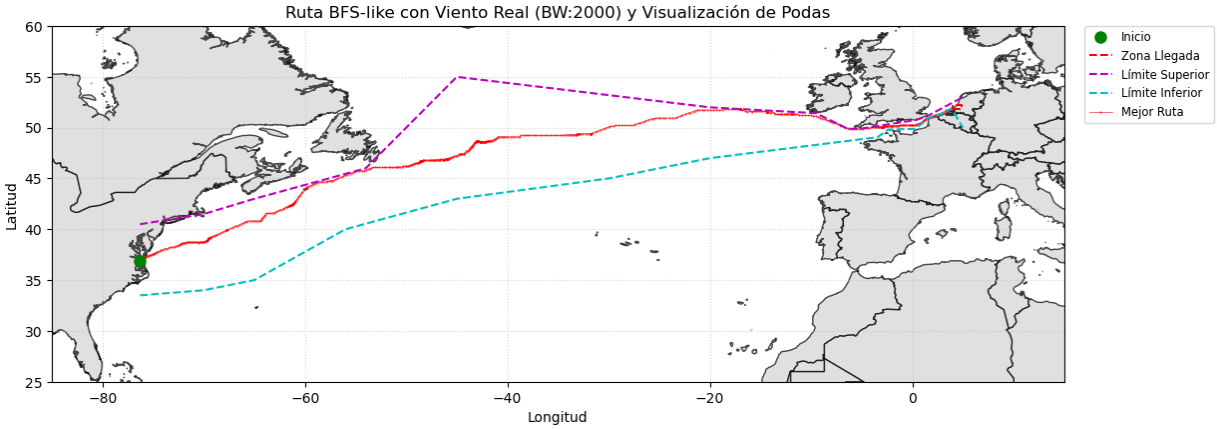

Figura 32: Resultado de una simulación de tiempo de viaje (salida 01/01/2024 00:00h)

Es importante subrayar que el desarrollo de este algoritmo se realiza para la creación de una herramienta de simulación indispensable para alcanzar el objetivo principal de este trabajo de fin de master. La finalidad última de todo este trabajo es poder ejecutar este algoritmo para cientos de fechas de salida a lo largo de un año completo, generando así un conjunto de datos robusto sobre los tiempos de viaje y, fundamentalmente, sobre su variabilidad.

Esta información es lo esencial para el posterior estudio económico. Solo estimando con precisión cuánto tarda un buque propulsado a vela en completar su ruta y cuán predecibles son esos tiempos, se podrá realizar una comparación justa y rigurosa con un buque convencional. Por tanto, este algoritmo conecta la navegación a vela con el análisis de viabilidad económica.

En el siguiente apartado, se discutirán las limitaciones prácticas de esta herramienta, principalmente en lo que respecta a su elevado coste computacional, y las soluciones de que se tuvieron que adoptar para poder llevar a cabo esta campaña de simulaciones.

3.8. Desafíos computacionales y ejecución a gran escala

Una vez validado el algoritmo, el paso final era ejecutarlo a gran escala para generar los datos necesarios para el estudio económico: una simulación para cada hora de salida de todo un año. Fue en esta fase donde el coste computacional, que hasta ahora había sido una guía para el diseño, se convirtió en un obstáculo que requirió una solución específica.

En las pruebas realizadas sobre un ordenador portátil se determinó que el tiempo medio de ejecución para una única simulación (una única fecha y hora de salida) era de aproximadamente 30 minutos. Para llevar a cabo el estudio completo, se necesitaba ejecutar una simulación para cada una de las 8,760 horas que componen un año.

El cálculo del tiempo total necesario en un único ordenador es el siguiente:

$$8760 \text{ simulaciones} \cdot 0{,}5 \frac{\text{horas}}{\text{simulación}} = 4380 \text{ horas}$$

$$4380 \text{ horas} \div 24 \frac{\text{horas}}{\text{día}} = 182{,}5 \text{ días}$$

Esto significa que ejecutar la campaña de simulación completa de forma secuencial habría requerido aproximadamente seis meses de computación ininterrumpida, una escala de tiempo completamente inviable para la realización de este trabajo.

Para superar esta barrera, se consiguió acceso a un entorno de Computación de Alto Rendimiento (HPC): el superordenador Magerit, perteneciente al Centro de Supercomputación y Visualización de Madrid (CeSViMa), gestionado por la UPM.

La ventaja fundamental de un sistema de supercomputación es su capacidad para el procesamiento en paralelo. En lugar de ejecutar las 8,760 simulaciones una detrás de otra, el problema se descompuso en tareas más pequeñas e independientes. Se crearon 366 scripts de Python, donde cada script era responsable de ejecutar las 24 simulaciones correspondientes a las salidas de un único día del año.

Estos 366 trabajos se enviaron al gestor de colas del superordenador (SLURM). El sistema Magerit permite la ejecución simultánea de un máximo de 100 trabajos por usuario. Por lo tanto, los 366 scripts se ejecutaron en lotes concurrentes. Este enfoque redujo drásticamente el tiempo total de computación: la campaña de simulación completa, que habría tardado seis meses, se pudo completar en aproximadamente cuatro días.

```
#!/bin/bash
##----------------------- Start job description ---------------$
#SBATCH --partition=standard
#SBATCH --job-name=0212
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=6G
#SBATCH --time=20:00:00
#SBATCH --mail-type=ALL
#SBATCH --mail-user=javier.sgil@alumnos.upm.es
##------------------------ End job description ----------------$
module --force purge
module load apps/2021
module load Python/3.11.5-GCCcore-13.2.0

python -u margerit0212.py &

wait
```

Figura 33: SLURM ejemplo para el día 02/12

La Figura 33 muestra un ejemplo de uno de los 366 scripts de envío utilizados para lanzar las simulaciones en el superordenador Magerit. Este tipo de archivo se comunica con el gestor de colas SLURM para solicitar recursos y ejecutar el código. El script se compone de tres partes principales:

- Descripción del Trabajo: Las líneas que comienzan con *#SBATCH* definen los parámetros de la simulación. En este caso, se especifica el tiempo máximo de ejecución (*--time*), la memoria RAM solicitada por CPU *(--mem-per-cpu*), el número de tareas (*--ntasks*) y la dirección de correo electrónico para recibir notificaciones sobre el estado del trabajo.

- Carga del Entorno: Las líneas module load se encargan de preparar el entorno de computación, cargando la versión de Python necesaria para la ejecución.

- Lanzamiento del Script: La línea final*, python -u margerit0212.py*, es el comando que ejecuta el algoritmo. El argumento *-u* se utiliza para que las salidas del programa (como los *prints* de estado) se muestren en tiempo real en los ficheros de registro, y *margerit0212.py* es el nombre del script de Python que contiene la simulación para un día específico del año.

La utilización de la supercomputación no fue una simple mejora, sino un requisito indispensable que hizo posible la obtención de los datos necesarios para el posterior análisis económico.

4. Explicación del algoritmo final

En este capítulo se procederá a explicar la estructura y el funcionamiento del código final del algoritmo de *weather routing*. El objetivo es describir los componentes principales del script, desde las librerías y parámetros iniciales hasta la lógica de las funciones más importantes, para ofrecer una comprensión clara de su implementación. El código completo se adjunta en el Anexo (página 129).

4.1. Librerías Utilizadas

El desarrollo del algoritmo se ha apoyado en un conjunto de librerías de Python de código abierto, cada una especializada en una tarea concreta. Las más relevantes son:

- Pandas: Se utiliza para la carga y manipulación de los datos de los diagramas polares, que se encuentran en formato CSV.
- NumPy: Es la librería fundamental para el cálculo numérico en Python. Se emplea para operaciones matemáticas, como la interpolación de velocidades.
- netCDF4: Biblioteca especializada que permite la lectura de los archivos en formato NetCDF (.nc4), utilizada para acceder a la base de datos de viento de MERRA-2.
- Collections.deque: Proporciona una implementación de una cola de alta eficiencia, que es la estructura de datos sobre la que se construye la frontera de búsqueda del algoritmo.
- Datetime: Módulo estándar para la manipulación de fechas y horas, indispensable para gestionar el tiempo de la simulación.
- CSV y OS: Módulos estándar para la escritura de los ficheros de resultados y para la gestión de rutas de archivos.

4.2. Parámetros Globales de la Simulación

Al inicio del script se define un conjunto de parámetros globales que configuran tanto la campaña de simulación completa como las características de cada viaje individual. Estos permiten modificar fácilmente las condiciones del estudio sin necesidad de alterar el código principal. Los más importantes son:

- Parámetros de la Campaña de Simulación: Se definen el nombre del archivo CSV donde se guardarán los resultados, el rango de fechas a simular (FECHA_INICIO_SIMS_STR, FECHA_FIN_SIMS_STR) y el intervalo en horas entre cada nueva simulación (INTERVALO_HORAS_SIMS).
- Parámetros Geográficos del Viaje: Se establecen las coordenadas del punto de salida (inicio), la zona de llegada (zona_llegada_rect), las coordenadas del CHECKPOINT_CANAL y las listas de puntos que definen el corredor de navegación (puntos_limite_superior y puntos_limite_inferior).

- Parámetros del Algoritmo: Se configuran las variables que controlan la Búsqueda en Haz, como el ancho del haz (BEAM_WIDTH) y la frecuencia con la que se aplica la poda (PODA_BEAM_CADA_N_EXPANDIDOS).

4.3. Análisis de las Funciones

A continuación, se describen las funciones que componen el algoritmo, en el orden lógico de funcionamiento.

Funciones para la Gestión de los Diagramas Polares

Este conjunto de funciones se encarga de cargar y procesar los datos de rendimiento del buque. La función *cargar_datos_polares* lee los archivos CSV y los almacena en una estructura de datos en memoria. Posteriormente, la función *calcular_velocidad_barco* es la que, recibiendo una velocidad y un ángulo de viento, interpola los datos de las curvas polares para devolver la velocidad que alcanzaría el buque en esas condiciones.

Funciones para el Acceso a los Datos de Viento

Estas funciones actúan como la interfaz entre el algoritmo y la base de datos de MERRA-2. La función *get_wind_at_node* recibe las coordenadas y la fecha de un nodo y se encarga de localizar y abrir el archivo .nc4 correspondiente. Para optimizar el rendimiento, se implementó un sistema de caché que mantiene en memoria el último archivo de viento abierto, evitando así tener que leerlo del disco repetidamente si varios nodos consecutivos pertenecen al mismo día.

Funciones Geográficas y Auxiliares

Este bloque contiene un conjunto de herramientas para realizar cálculos geográficos. Destacan la función *mover*, que calcula la nueva posición de un nodo a partir de un rumbo y una distancia usando navegación ortodrómica, y las funciones de comprobación como *en_corredor_lineal* y *en_zona_llegada*, que validan si un nodo se encuentra dentro de las áreas permitidas.

4.4. Función *obtener_rutas*

Esta función es el verdadero motor físico de la simulación. Su responsabilidad es determinar los movimientos realistas que puede realizar el buque durante una hora, dadas unas condiciones meteorológicas y de posición específicas. Se llama cada vez que el algoritmo principal necesita expandir un nodo para generar a sus hijos. A continuación, se detalla su lógica de funcionamiento.

Datos:

- Posición actual: coordenadas (latitud, longitud) del nodo padre.
- Tiempo actual: la fecha y hora correspondientes al nodo padre.
- Datos polares: la estructura de datos con el rendimiento del buque.

Resultado:

- Lista de tramos válidos: una lista de tuplas, donde cada una contiene un par (Rumbo, Distancia) que el buque puede recorrer en la siguiente hora.

Inicialización;

Obtener el vector de viento (U, V) para la posición y tiempo actual desde la base de datos MERRA-2;

Calcular la velocidad y la dirección del viento real a partir de sus componentes;
Inicializar una lista vacía para almacenar las nuevas rutas;
for cada uno de los 5 rumbos predefinidos ***do***
    Calcular el ángulo de incidencia del viento sobre el buque para ese rumbo;
    if el ángulo de incidencia está en la "zona de no avance" ***then***
        Corregir el rumbo del buque al ángulo mínimo de navegación en ceñida;
        Actualizar el ángulo de incidencia al valor límite de la zona de no avance;
    end
    Consultar los diagramas polares con la velocidad del viento e interpolar para obtener la velocidad del buque;
    Calcular la distancia (en km) que se recorrería en una hora a esa velocidad;
    Añadir el par (Rumbo Corregido, Distancia) a la lista de nuevas rutas;
end
Devolver la lista de nuevas rutas;

El funcionamiento de esta función comienza obteniendo las condiciones de viento exactas para la posición y el momento en el que se encuentra el buque. A partir de ahí, itera sobre un conjunto de cinco rumbos predefinidos (que representan las opciones de navegación que se van a evaluar) y calcula para cada uno la velocidad que alcanzaría la embarcación.

El paso más importante es la consulta a los diagramas polares (línea 10). Dado que la velocidad del viento real raramente coincidirá con las curvas exactas de los diagramas (5, 10, 15 nudos, etc.), se realiza una interpolación lineal entre las dos curvas más cercanas para obtener un valor de velocidad del buque.

Finalmente, la función incorpora una lógica de navegación avanzada (líneas 6-9). Si al evaluar un rumbo se determina que este caería dentro de la zona de no avance del buque (es decir, navegando directamente contra el viento), el algoritmo no lo descarta. En su lugar, lo corrige al ángulo mínimo posible en el que las velas todavía pueden generar empuje. Esto simula la capacidad de un velero real de ceñir o virar para avanzar contra el viento, dotando a la simulación de un comportamiento mucho más realista.

4.5. Función principal: Weather Routing

Esta función es el motor de todo el sistema, dirige el proceso de búsqueda de la ruta óptima, combinando la exploración de nodos con las restricciones geográficas, la física del buque y las estrategias de poda inteligente. Aunque su nombre contiene "BFS", su comportamiento final es el de una Búsqueda en Haz (Beam Search), una versión mucho más avanzada y eficiente. Su objetivo es encontrar el camino de menor duración desde el punto de partida hasta la zona de llegada.

Datos:

- Punto de partida: coordenadas (latitud, longitud) y fecha/hora de salida.
- Datos del buque: los diagramas polares cargados en memoria.
- Entorno geográfico: la zona de llegada, el corredor de navegación y los checkpoints.
- Parámetros de búsqueda: el "ancho del haz" y la frecuencia de la poda.

Resultado:

- Tiempo de Viaje Óptimo: El menor número de horas encontrado para llegar al destino.
- Trayectoria Óptima: La secuencia de coordenadas horarias de la ruta más rápida.

Inicialización;

```
Calcular distancias iniciales al checkpoint y al destino final;
Crear una cola vacía para la búsqueda (frontera);
Crear un registro para almacenar todos los nodos generados;
Crear el nodo raíz (posición=Punto de Partida, tiempo=0, padre=Nulo);
Añadir el nodo raíz a la cola y al registro;
Inicializar Mejor_Tiempo_Encontrado a un valor infinito;
```

Antes de comenzar la búsqueda, el algoritmo prepara todas las estructuras de datos y variables necesarias. Se calculan las distancias iniciales que servirán de referencia para la heurística. Se crea la cola (una estructura deque), que almacenará todos los "caminos" o nodos que están pendientes de ser explorados. Se inicializa un registro que guardará la información de cada nodo que se genere, con su identificador único y una referencia a su padre, lo que será crucial para reconstruir la ruta al final. Finalmente, se crea el nodo raíz que representa el barco en el puerto de salida en el momento cero, y se añade a la cola para dar comienzo al bucle principal.

```
while la cola de búsqueda no esté vacía do
    //  1. PROCESAMIENTO DEL NODO ACTUAL
    Extraer el primer nodo de la cola (el más antiguo o el que tenga más puntuación de heurística)
    Comprobar condiciones de parada (ej: si este camino ya es más largo que la mejor ruta encontrada);
    //  2. EXPANSIÓN Y GENERACIÓN DE NODOS HIJOS
    Llamar a la función obtener_rutas para el nodo actual;
    for cada Tramo Válido (Rumbo, Distancia) devuelto por obtener_rutas do
            Calcular la nueva posición geográfica usando navegación ortodrómica;
            if la nueva posición es válida (dentro del corredor, no en zona crítica) then
                    Crear un nuevo nodo hijo con la nueva posición, tiempo+1, y referencia al nodo
                    actual como padre;
                    Añadir el nuevo nodo hijo al registro y a la cola;
                    if el nuevo nodo ha llegado a la Zona de Llegada then
                            Actualizar Mejor_Tiempo_Encontrado si este camino es más rápido;
                    end
            end
    end
    //  3. PODA PERIÓDICA (BÚSQUEDA EN HAZ)
    if es momento de podar (ej: cada 1000 nodos expandidos) then
            if el tamaño de la cola supera el "ancho del haz" then
                    Calcular la "promesa" (heurística) de cada nodo en la cola;
                    Ordenar la cola según la heurística de cada nodo;
                    Conservar los N mejores nodos (Explotación);
                    Conservar M nodos aleatorios adicionales (Exploración);
                    Descartar todos los demás nodos de la cola;
            end
    end
end // fin del while
//  4. FINALIZACIÓN Y RECONSTRUCCIÓN DE LA RUTA
if se encontró una ruta (Mejor_Tiempo_Encontrado no es infinito) then
    Reconstruir la trayectoria siguiendo los punteros al padre desde el nodo final;
    Devolver Tiempo de Viaje Óptimo y Trayectoria Óptima;
else
    Devolver un fallo;
end
```

El corazón del algoritmo es un gran bucle ***while*** que se ejecuta mientras haya rutas posibles por explorar. En cada iteración, se extrae el nodo más antiguo de la cola para ser analizado, manteniendo así una exploración ordenada por niveles, similar a la del BFS.

El proceso se divide en dos fases principales: expansión y poda. En la fase de expansión (parte 2, “Expansión y generación de nodos hijos”), se llama al *obtener_rutas* para generar todos los posibles movimientos realistas para la siguiente hora. Cada uno de estos movimientos genera un nuevo nodo hijo que, tras pasar los filtros de las restricciones geográficas, se añade a la cola.

La fase de poda (parte 3, “Poda periódica”) es la que implementa la Búsqueda en Haz y es la que dota al algoritmo de eficiencia. Este proceso no se ejecuta en cada iteración, sino periódicamente (por ejemplo, cada 1000 nodos expandidos). Cuando se activa, el algoritmo para la exploración para hacer una limpieza inteligente de la cola:

- Evalúa a todos los nodos pendientes usando la heurística, que les asigna una puntuación de lo prometedores que son.

- Selecciona a un número limitado de los mejores y a un pequeño número de nodos al azar.

- Descarta a todos los demás, rutas poco prometedoras son eliminadas de la memoria en este paso, permitiendo al algoritmo centrar sus recursos en las que de son más importantes.

Una vez que el bucle termina, si se ha encontrado una solución, se procede a la reconstrucción (parte 4, “Finalización y reconstrucción de la ruta”), que consiste en seguir el rastro de los nodos padre desde el punto de llegada hasta el de partida para obtener la trayectoria de coordenadas de la ruta solución.

Ajuste de hiperparámetros de búsqueda

Es fundamental destacar que el comportamiento de la Búsqueda en Haz se controla mediante una serie de hiperparámetros clave. Los dos más importantes, que definen el equilibrio entre la exhaustividad de la búsqueda y el coste computacional, son el ancho del haz (BEAM_WIDTH) y la frecuencia de la poda (PODA_BEAM_CADA_N_EXPANDIDOS).

- BEAM_WIDTH (Ancho del Haz): Este parámetro define el número máximo de rutas candidatas que el algoritmo mantiene activas en memoria. Un valor más alto permite al algoritmo explorar un mayor número de posibles trayectorias de forma simultánea, aumentando la probabilidad de encontrar la ruta óptima. Sin embargo, esto se hace a costa de un mayor consumo de memoria y tiempo de ejecución. Un valor más bajo hace que el algoritmo sea más rápido y ligero, pero aumenta el riesgo de descartar prematuramente la que podría haber sido la mejor ruta.

- PODA_BEAM_CADA_N_EXPANDIDOS (Frecuencia de Poda): Este valor determina cada cuántos nodos explorados se activa el proceso de poda. Un número pequeño significa que la "limpieza" de la cola de búsqueda se realiza con mucha frecuencia, manteniendo el consumo de memoria bajo control. Un número más grande permite que la cola crezca más entre podas, lo que puede aumentar el pico de memoria utilizado.

La selección de estos valores finales fue el resultado de un proceso de experimentación en el que se realizaron múltiples pruebas. En estas pruebas se constató la relación directa entre el número de nodos conservados y el rendimiento, a mayor ancho del haz, el algoritmo explora más a fondo el espacio de soluciones, pero el tiempo de ejecución de cada simulación se incrementa de forma muy notable.

Por tanto, los parámetros fijados en el código final representan un equilibrio entre la exhaustividad de la búsqueda y el coste computacional. Se determinó que estos valores ofrecían un equilibrio adecuado para las simulaciones. Es importante subrayar que, con un hardware más potente, especialmente con una arquitectura de ordenador diseñada para este tipo de algoritmos que requieren un uso intensivo de memoria y procesador, seguramente se podrían utilizar anchos de haz mucho mayores, lo que potencialmente podría conducir a la obtención de resultados de ruta aún más optimizados.

4.6. Función *ejecutar_simulaciones*

Esta es la función de más alto nivel del script. Su propósito no es encontrar una ruta, sino actuar como un sistema que gestiona la ejecución de miles de simulaciones. Es la responsable de iterar a través de todas las fechas de salida, llamar al algoritmo de búsqueda para cada una, y registrar los resultados para su posterior análisis.

Datos:

- Parámetros Globales: El rango de fechas de la campaña, el intervalo de horas entre salidas y el nombre del archivo de resultados.

Resultado:

- Un archivo CSV que contiene una línea por cada simulación ejecutada, con el tiempo de viaje resultante y otros datos de interés.

Inicialización;

Leer los parámetros globales de la campaña de simulación;
Abrir (o crear si no existe) el archivo CSV de resultados en modo de añadir ('append');
Escribir la fila de cabecera en el archivo si es nuevo;
Inicializar la Fecha_Actual con la fecha y hora de la primera simulación;
while la Fecha_Actual sea menor o igual a la Fecha Final de la campaña ***do***
    // Ejecutar una simulación individual
    Registrar el tiempo de inicio;
    Llamar a la función principal de búsqueda (routing_bfs_real_wind) con la Fecha_Actual;
    Registrar el tiempo de fin y calcular la duración total del cómputo;
    // Guardar el resultado en el archivo
    if el algoritmo encontró una ruta ***then***
        Calcular la fecha de llegada a partir del tiempo de viaje;
        Escribir una nueva línea en el CSV con: [Fecha_Actual, Duración_Cómputo, Tiempo_Viaje, Fecha_Llegada];
    else
        Escribir una nueva línea en el CSV indicando que no se encontró ruta ("N/A");
    end
    // Preparar la siguiente iteración
    Incrementar la Fecha_Actual en el intervalo de horas definido (ej: 1 hora);
end // fin del while
Cerrar el archivo CSV;

El funcionamiento de esta función se centra en un gran bucle while que recorre todo el periodo de estudio. En cada iteración de este bucle, se simula un único viaje completo:

- Se establece una fecha y hora de salida.
- Se invoca a la función principal de búsqueda (*routing_bfs_real_wind*) para que calcule la ruta óptima para esa salida específica.

- Se mide el tiempo de computación que ha tardado la simulación en completarse.

- Una vez que la función de búsqueda devuelve un resultado (ya sea el tiempo de viaje en horas o un fallo), este se escribe inmediatamente en el archivo CSV.

Este método de escritura inmediata después de cada simulación es una medida de seguridad crucial para procesos de larga duración, ya que asegura que si la ejecución se interrumpe por cualquier motivo, los resultados calculados hasta ese momento no se pierden.

Al finalizar el bucle, se obtiene el resultado principal de todo el trabajo de simulación: un archivo de datos estructurado que contiene los tiempos de viaje para cada hora de salida a lo largo de un año. Este archivo es la materia prima indispensable sobre la que se construirá todo el análisis de variabilidad y el estudio económico que se presentan en los siguientes capítulos.

5. Obtención de tiempos de viaje y estudio estadístico

Este apartado se centra en el resultado de la aplicación del algoritmo a gran escala: la obtención de un conjunto de datos de los tiempos de viaje para la ruta entre Hampton Roads y Róterdam. A partir de estos datos, se realizará un análisis estadístico para medir la duración de las travesías y, fundamentalmente, su variabilidad.

5.1. Ejecución de las simulaciones anuales

Como se justificó previamente, la ejecución de la campaña de simulación completa era computacionalmente inviable en un único ordenador. La solución fue utilizar el sistema de supercomputación Magerit del CeSViMa.

El objetivo era simular tantas salidas como fuera posible durante el año 2024. Sin embargo, surgió una limitación práctica: la base de datos de viento de MERRA-2 utilizada para este estudio finaliza el 31 de diciembre de 2024. Dado que la duración media de un viaje es de varias semanas, cualquier simulación que comenzara a finales de diciembre necesitaría datos de viento de enero de 2025 para poder completarse, datos de los que no se disponía.

Por este motivo, la campaña de simulación se ejecutó para todas las horas de salida comprendidas entre el 1 de enero de 2024 y, aproximadamente, el 3 de diciembre de 2024, asegurando así que todas las rutas tuvieran datos de viento suficientes para llegar a su destino dentro del año. Tras aproximadamente cuatro días de cómputo en paralelo, se obtuvo el resultado final: un único archivo en formato CSV que contiene más de 8,100 entradas exitosas. Cada una de estas entradas representa un viaje completado, con su fecha de salida y el tiempo de viaje resultante en horas, constituyendo la base para el siguiente análisis estadístico.

5.2. Adaptación del algoritmo para la ruta de vuelta

Además de la ruta principal de ida (Hampton Roads - Róterdam), se realizó una segunda tanda de simulaciones para la ruta de retorno (Róterdam - Hampton Roads).

La hipótesis inicial, basada en la navegación a vela histórica, era que la trayectoria óptima seguiría una ruta muy al sur para aprovechar los vientos alisios (*Trade Winds*). Para comprobarlo, se configuró el algoritmo con libertad de movimiento: se invirtieron los puntos de salida y llegada, y se definió un corredor de navegación con un límite inferior extremadamente bajo, permitiendo que la búsqueda explorase de forma natural tanto las rutas del norte como las del sur, cerca de las Canarias.

Los resultados de las simulaciones fueron concluyentes y contrarios a la hipótesis inicial. El algoritmo determinó que la ruta más rápida no era la de los alisios, el gran desvío necesario para alcanzar esas latitudes añadía una penalización de distancia y tiempo que no llegaba a ser compensada por los vientos favorables. Incluso se realizaron pruebas adicionales forzando al algoritmo a pasar por checkpoints situados más al sur, pero en todos los casos, estas rutas guiadas resultaron en tiempos de viaje superiores a los que el algoritmo encontraba de forma autónoma.

Por tanto, no se optó por una ruta, sino que se aceptó el resultado que el propio algoritmo descubrió como el más eficiente: una trayectoria de retorno más parecida a la de ida. Aun así, el código para la simulación de vuelta requirió las siguientes adaptaciones:

- Puntos de salida y llegada: como es lógico, se invirtieron las coordenadas de los puntos de inicio y zona_llegada_rect para representar la nueva travesía.

- Corredor de navegación: se definió un nuevo corredor de navegación. Aunque su trazado es similar al de la ruta de ida, es importante destacar que el límite inferior se mantuvo deliberadamente muy al sur. Esta decisión se tomó para no restringir por completo al algoritmo; si en alguna simulación se presentaran condiciones meteorológicas excepcionalmente anómalas que hicieran viable una ruta más al sur, el algoritmo tendría la libertad de explorarla.

- Direcciones de búsqueda: el conjunto de rumbos relativos que el algoritmo explora en cada paso (angulos_rumbo_usuario) fue modificado para priorizar el avance hacia el suroeste, en lugar del noreste.

- Lógica del checkpoint: la condición lógica dentro de la función heurística que determina cuándo se ha superado el checkpoint del Canal de la Mancha fue ajustada para funcionar correctamente en un viaje con rumbo oeste.

Es importante destacar que el núcleo del algoritmo (la Búsqueda en Haz, la estructura de datos, el motor físico de consulta de viento y diagramas polares) permaneció intacto. Esto demuestra la flexibilidad de la herramienta desarrollada, que puede ser adaptada para resolver diferentes problemas de enrutamiento simplemente modificando sus parámetros estratégicos de entrada. Este código modificado se puede encontrar en los Anexos (página 143)

5.3. Post-procesamiento de los datos

El algoritmo de *weather routing* se diseñó para encontrar la ruta más rápida posible dada una fecha y hora de salida fijas. Sin embargo, no posee la inteligencia para decidir si estratégicamente es mejor esperar una o varias horas en puerto para zarpar con condiciones de viento más favorables. Para corregir esta limitación y obtener un conjunto de datos que refleje la mejor estrategia posible, se desarrolló un script de post-procesamiento que optimiza los resultados brutos de la simulación.

El funcionamiento de este script es el siguiente:

1. Carga y ordenación: primero, se carga el conjunto de datos completo con todos los viajes y se ordena cronológicamente por fecha de salida. Este orden es esencial para el correcto funcionamiento del proceso.

2. Iteración y búsqueda de oportunidades: A continuación, el script itera sobre cada uno de los viajes registrados. Para cada viaje (llamémoslo viaje i), que tiene una fecha de salida S_i y una fecha de llegada original L_i, el script busca en todos los viajes que salieron *después* que él.

3. Identificación de la mejor alternativa: Dentro de ese subconjunto de viajes posteriores, busca si existe alguno que, a pesar de haber salido más tarde, llegara a su destino *antes* que el viaje *i*. Si encuentra uno o más de estos casos, identifica la fecha de llegada más temprana de todas ellas

4. Recálculo del Tiempo de Viaje: Si se ha encontrado una llegada mejor, se recalcula el tiempo de viaje para la salida original S_i. La nueva duración corregida es el tiempo total transcurrido desde la salida original S_i hasta la nueva y mejorada llegada. Este nuevo valor representa la estrategia óptima de "esperar en puerto" y zarpar en el momento más favorable.

El resultado de este proceso, como se puede apreciar en la Figura 34 es un conjunto de datos final y optimizado. Cada tiempo de viaje en este nuevo archivo representa el mínimo tiempo real posible para una oportunidad de salida, ya que incorpora la posibilidad de que el buque espere en puerto a que las condiciones meteorológicas mejoren. Es este conjunto de datos corregido el que se utilizará para todo el análisis estadístico que se presenta a continuación. El script de este procesamiento de datos se puede encontrar en los anexos (página 156).

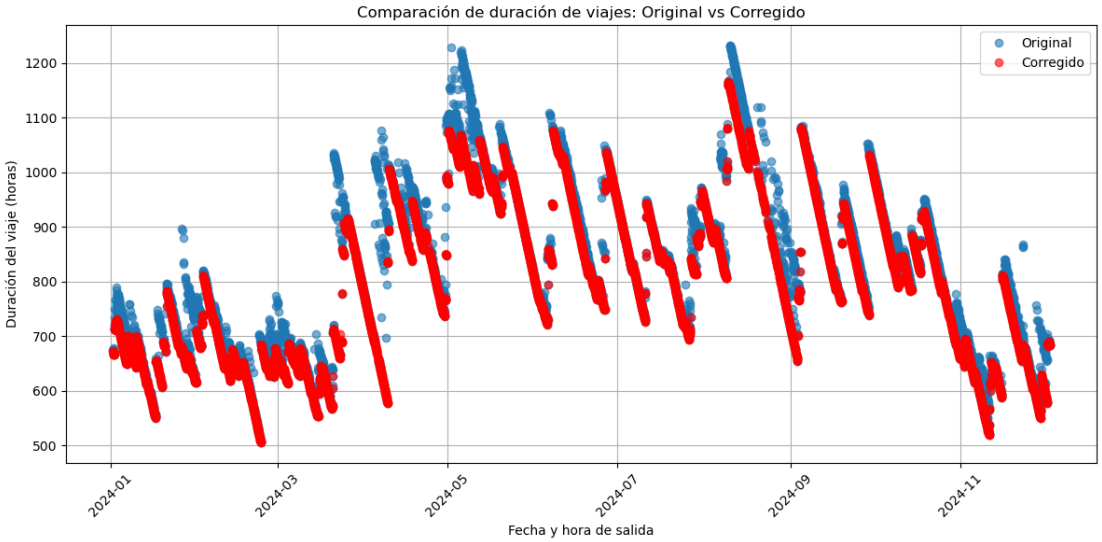

Figura 34: Datos de tiempos de viaje de ida corregidos

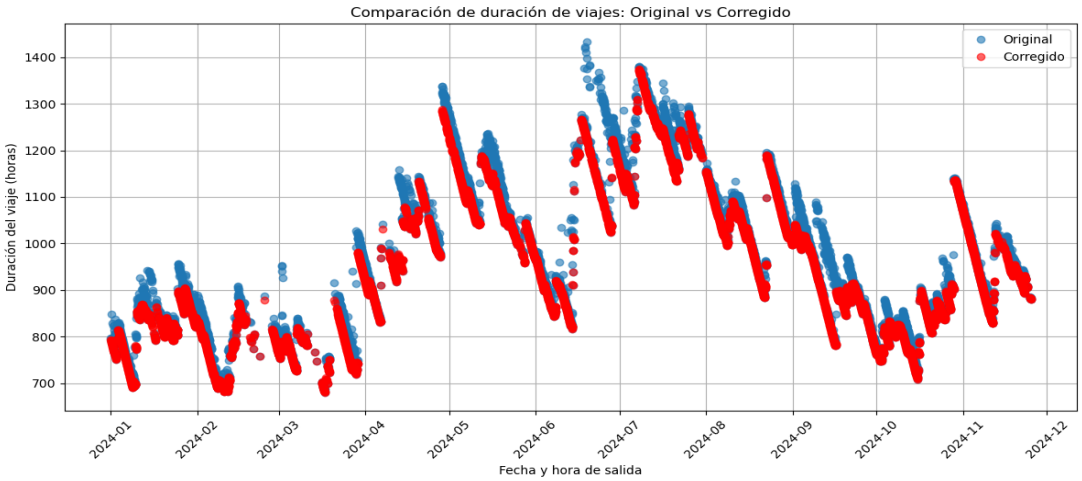

Figura 35: Datos de tiempos de viaje de vuelta corregidos

5.4. Análisis estadístico de los tiempos de viaje

Con el conjunto de datos final, limpio y corregido, se procede a realizar el análisis estadístico de los resultados. El objetivo de este análisis es, por un lado, cuantificar la duración esperada de las travesías y, por otro, y más importante para el estudio económico, medir su variabilidad.

La principal herramienta para este análisis es el diagrama de caja y bigotes (*boxplot*). En los gráficos presentados a continuación, se han ajustado los "bigotes" para que representen los percentiles 10 y 90. Esto significa que los bigotes muestran el rango de tiempo en el que se encuentra el 80% de todos los viajes, ofreciendo una visión muy robusta de la variabilidad esperada al excluir el 10% de los viajes más rápidos y el 10% de los más lentos.

Análisis del Viaje de Ida (Hampton Roads -Róterdam)

A continuación, en la Figura 36 se presenta el diagrama de caja y bigotes que resume la distribución de los tiempos de viaje para la ruta de ida.

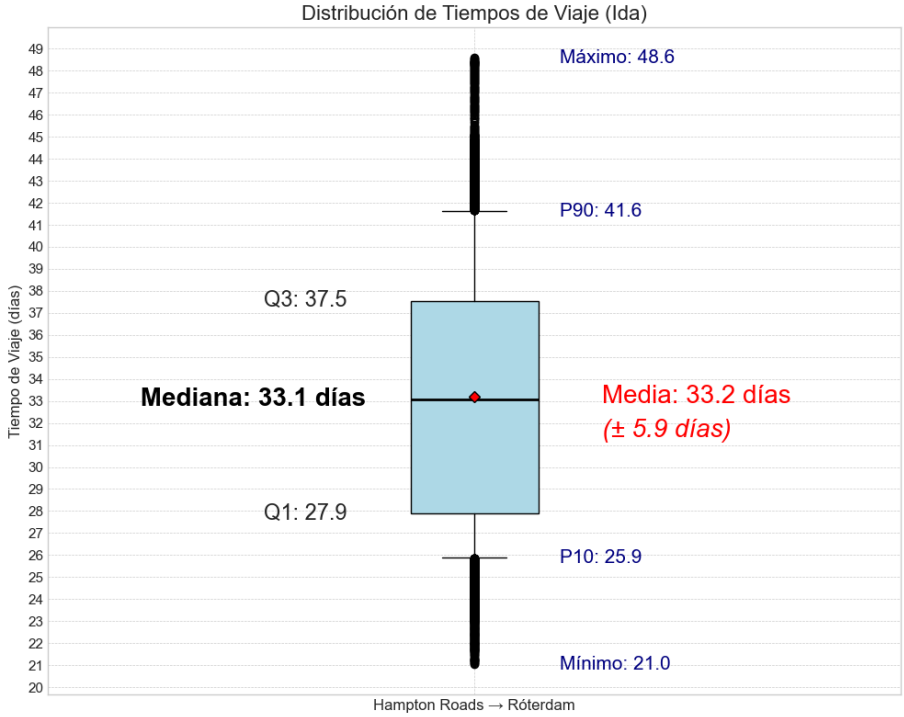

Figura 36: Boxplot de los tiempos de viaje de ida

Los resultados numéricos extraídos de las más de 8,100 simulaciones exitosas se resumen en la siguiente tabla:

Tabla 4: Datos estadísticos de los tiempos de viajes de ida

Métrica estadística	Tiempo de viaje(días)
Media	33,2
Mediana	33,1
Desviación estándar	5,9
Mínimo (viaje más rápido)	21
Máximo (viaje más lento	48,6
Percentil 10 (P10)	25,9
Percentil 25 (P25)	27,9
Percentil 75 (P75)	37,5
Percentil 90 (P90)	41,6

Como se puede observar, el tiempo de viaje mediano para la ruta de ida es de 33,1 días. La media, muy similar con 33,2 días, sugiere una distribución de los datos relativamente simétrica.

La variabilidad es el factor clave de este análisis. La "caja" del diagrama, que representa el 50% central de los viajes, muestra que la mitad de las travesías se completan en un tiempo de entre 27,9 y 37,5 días. Ampliando el rango, se observa que el 80% de todos los viajes (delimitado por los "bigotes" en los percentiles 10 y 90) se finalizaron en un tiempo comprendido entre 25,9 y 41,6 días. Esta dispersión, cuantificada por una desviación estándar de 5,9 días, es considerable y será un factor crucial en el posterior análisis económico. Los Boxplot se han generado a través de la herramienta Python con todos los datos de los tiempos de viaje calculados.

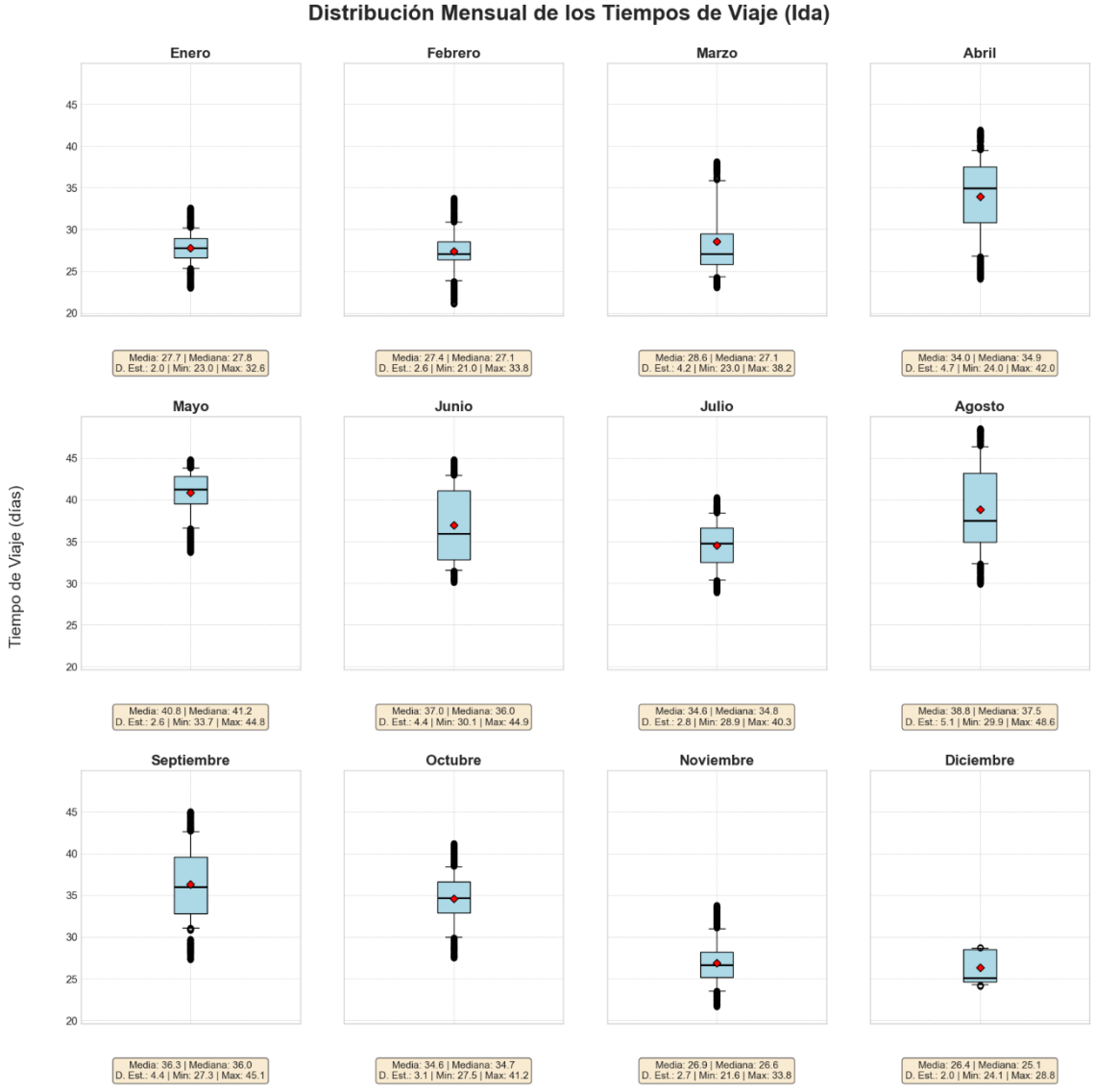

Figura 37: Boxplot de los tiempos de viaje (ida) de cada mes del año 2024

El análisis de los datos mensuales de la Figura 37 revela un patrón estacional muy marcado, con diferencias significativas tanto en la duración como en la predictibilidad de los viajes a lo largo del año.

Se identifican claramente dos temporadas de navegación. Los meses de invierno y finales de otoño son notablemente más rápidos, sin tener en cuenta el mes de diciembre, el cual únicamente cuenta con simulaciones hasta el día 4, ya que los días posteriores terminan en

enero de 2025, cuyos datos de viento no se han considerado y por lo tanto no se pueden simular dichos viajes. Noviembre se establece como el mes más eficiente, con un tiempo de viaje mediano de 26,6 días. Le siguen de cerca febrero (27,1 días) y enero (27,8 días).

Por otro lado, los meses de primavera tardía y verano son considerablemente más lentos. Mayo es el mes con el peor rendimiento, con una mediana de 41,3 días, seguido de agosto (37,5 días). Esto significa que un viaje típico en mayo es aproximadamente 14,6 días más largo (un 55% más) que un viaje típico en noviembre, una diferencia de más de dos semanas que evidencia el enorme impacto de los patrones de viento estacionales.

La desviación estándar revela qué meses ofrecen viajes más consistentes.

- Los meses más predecibles son enero (desviación de solo 1,96 días) y mayo (2,63 días). Aunque mayo es el mes más lento, sus resultados son relativamente consistentes. Les siguen muy de cerca febrero y noviembre con desviaciones de alrededor de 2,7 días, haciendo de los meses de noviembre, enero y febrero los más interesantes para realizar los viajes.

- El mes más impredecible y con mayor variabilidad es agosto, con una desviación estándar de 5,11 días. Esto indica que, además de ser un mes lento, los tiempos de viaje pueden variar en más de 5 días con respecto a la media, lo que introduce una gran incertidumbre operativa. Le siguen abril (4,67 días) y septiembre (4,44 días) como meses de alta variabilidad, probablemente por ser periodos de transición estacional.

Los valores mínimos y máximos absolutos a lo largo del año (excluyendo diciembre) subrayan la importancia de la influencia meteorológica:

- El viaje más rápido de todas las simulaciones se registró en febrero, completándose en 21,0 días.
- El viaje más lento tuvo lugar en agosto, requiriendo 48,6 días.

La diferencia de más de 27 días entre el mejor y el peor escenario posible demuestra que, si bien la ruta es viable, su rendimiento económico y operativo está fundamentalmente condicionado por la época del año en que se realiza la travesía.

Análisis del Viaje de vuelta (Róterdam -Hampton Roads)

Se realizó un análisis idéntico para las simulaciones de los viajes de vuelta.

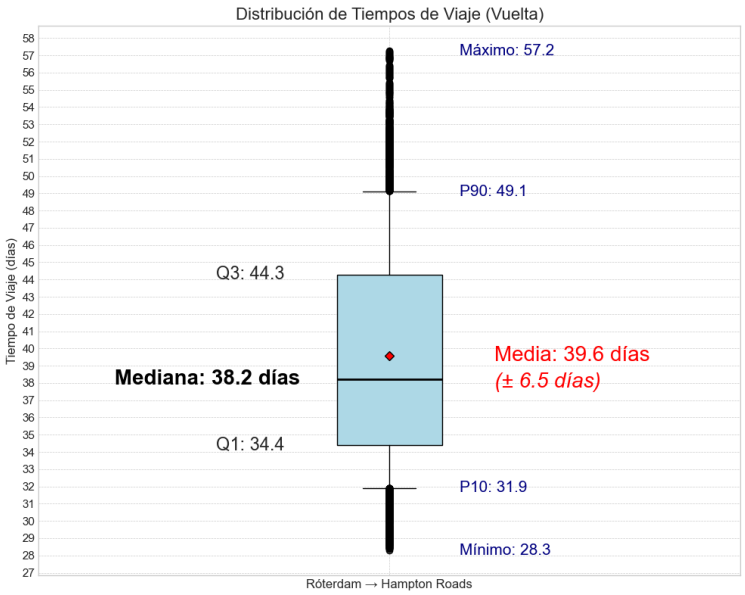

Figura 38: Boxplot de los tiempos de viaje de vuelta

Tabla 5: Datos estadísticos de los tiempos de viajes de vuelta

Métrica estadística	Tiempo de viaje(días)
Media	39,6
Mediana	38,2
Desviación estándar	6,5
Mínimo (viaje más rápido)	28,3
Máximo (viaje más lento	57,2
Percentil 10 (P10)	31,9
Percentil 25 (P25)	34,4
Percentil 75 (P75)	44,3
Percentil 90 (P90)	49,1

El tiempo de viaje mediano para la ruta de retorno es de 38,2 días. La variabilidad, medida por la desviación estándar de 6,5 días, 1,5 días más que en el viaje de ida, lo que sugiere que la ruta es menos predecible. El 80% de los viajes se completan en un rango de entre 31,9 y 49,1 días.

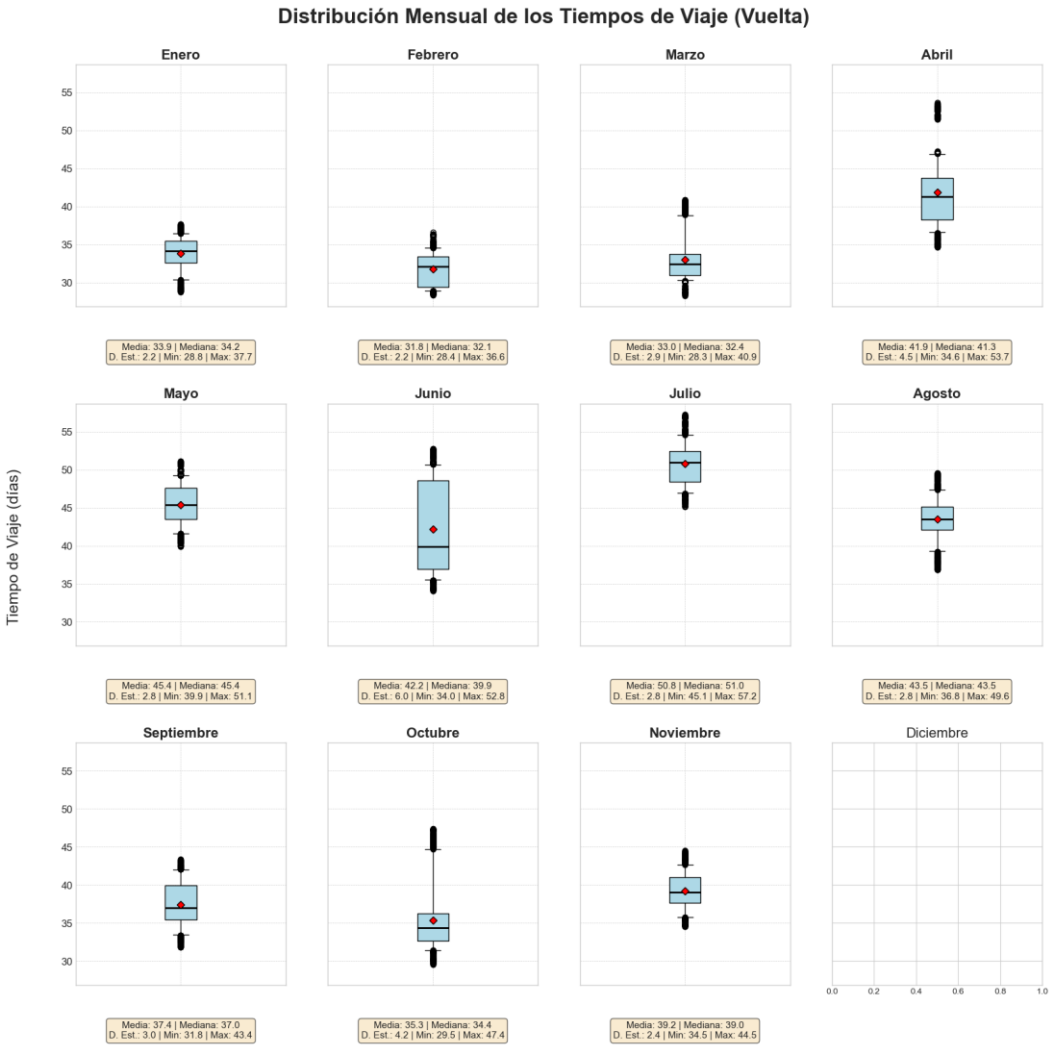

Figura 39: Boxplot de los tiempos de viaje (vuelta) de cada mes del año 2024

El análisis de los datos de los viajes de vuelta revela un comportamiento estacional claro, con variaciones importantes en la duración y la regularidad de las travesías a lo largo del año.

Se pueden distinguir dos temporadas principales para la navegación de vuelta. Los meses de invierno se perfilan como los más eficientes. Febrero destaca como el mes más rápido, con un tiempo de viaje mediano de 32,1 días, seguido de cerca por marzo, con 32,4 días, y enero, con 34,2 días.

En contraste, los meses de finales de primavera y verano presentan travesías más largas. Julio se posiciona como el mes menos favorable, con una mediana de 51 días, seguido por mayo, con 45,4 días. Esto implica que un viaje en julio es, en promedio, casi 19 días más largo (un 59% más) que uno en febrero. Esta diferencia de casi tres semanas refleja el gran impacto que tienen los patrones de viento estacionales en la ruta de vuelta.

La desviación estándar de los datos nos permite conocer qué meses ofrecen una mayor consistencia en los tiempos de viaje. Los meses con mayor regularidad son enero y febrero, ambos con una desviación estándar de solo 2,2 días. Les sigue de cerca noviembre, con 2,4 días. Esto convierte a los meses de invierno no solo en los más rápidos, sino también en los más fiables para planificar las operaciones. El mes con la mayor variabilidad y, por tanto, el más impredecible, es junio, con una desviación estándar de casi 6 días. Esto significa que, además de ser un mes relativamente lento, la duración de los viajes puede variar significativamente,

introduciendo una notable incertidumbre. Le siguen abril (4,5 días) y octubre (4,2 días) como meses de alta variabilidad, probablemente por corresponder a periodos de transición entre estaciones.

Los valores extremos registrados a lo largo del año confirman la fuerte influencia de las condiciones meteorológicas en los viajes de vuelta, el viaje más rápido de todas las simulaciones se completó en marzo, con una duración de 28,3 días. El viaje más lento tuvo lugar en julio, requiriendo 57,3 días.

La diferencia de 29 días entre el escenario más favorable y el más desfavorable demuestra que, aunque la ruta de vuelta es viable, su rendimiento operativo y económico depende fundamentalmente de la época del año en que se realice

Finalmente, es importante señalar que no se dispone de datos para el mes de diciembre, ya que las simulaciones que comienzan en dicho mes finalizan en enero del año siguiente, para el cual no se han considerado datos de viento. Además, como era de esperar, los tiempos de viaje de vuelta son bastante más largos que los de ida. Esto se debe a que los patrones de viento predominantes en la zona de navegación son, en general, más favorables para el trayecto de ida, haciendo que la vuelta sea una travesía más lenta.

Capítulo 5
Balance eléctrico y consumos de combustible

En este capítulo se va a calcular y comparar el consumo diario de combustible de las dos alternativas de buque: el de referencia, con propulsión convencional y el propulsado a vela. Este análisis es un paso fundamental antes de abordar el estudio económico, ya que la diferencia en el consumo de combustible es la principal variable que determinará la viabilidad del buque a vela.

Para ello, se analizará el perfil de consumo de cada buque por separado, desglosándolo en dos grandes bloques: el consumo de la propulsión principal y el consumo para la generación de energía eléctrica de los servicios a bordo.

El objetivo final es obtener una cifra clara, en toneladas de combustible por día, que cuantifique el ahorro que ofrece el buque a vela. Este dato será un pilar central en el análisis de costes operativos.

1. Buque con propulsión convencional

Para poder realizar una comparación, es necesario establecer un perfil de consumo de combustible para un buque equipado con un sistema de propulsión convencional. Este análisis se basa en un buque con las mismas dimensiones y características principales que el buque a vela, pero equipado con un motor principal para su propulsión.

1.1. Selección del motor propulsor

El primer paso para determinar el consumo de un buque es calcular la potencia que su motor principal debe entregar para que pueda navegar a una velocidad de operación estándar. Para ello se hará uso de la curva de resistencia al avance la cual se obtiene de la librería ShipSim (Basilio Puente & M. Dolores Fernandez, 2021)

La curva de resistencia al avance viene determinada a partir de una curva polinómica de tercer orden de la siguiente forma: {{9.1437e2, 3}, {-5.2863e3, 2}, {1.9657e4, 1}}. La cual se traduce en la siguiente gráfica:

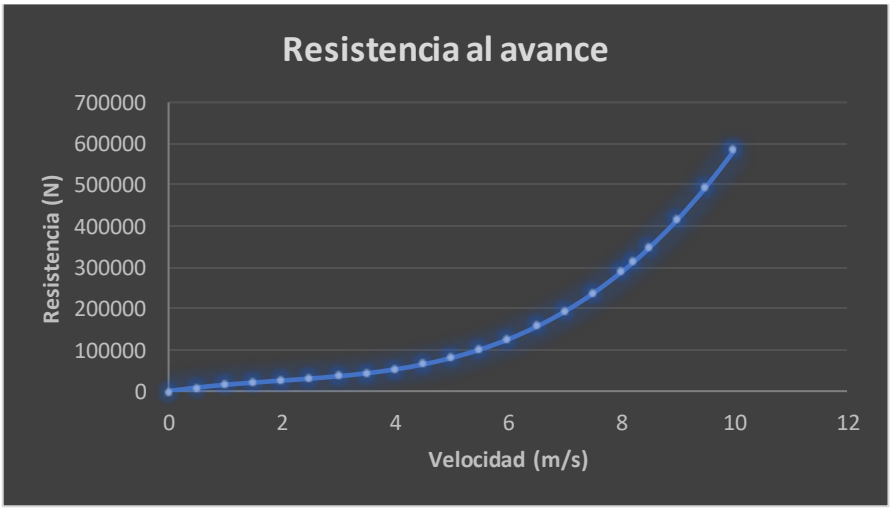

Figura 40: Resistencia al avance (N) VS velociad (m/s)

Suponiendo una velocidad para el buque convencional de 16 nudos, 8.23 m/s obtenemos una resistencia de 313476 N incluyendo un 15% del margen de mar. Para obtener la potencia necesaria del motor se calculará la potencia efectiva (EKW) y la potencia al freno (BKW) considerando un rendimiento del propulsor del 60% y un rendimiento de la línea de ejes del 98%.

$$EKW = R * V = 313476 * 8.23/1000 = 2580 \text{ kW}$$

R: resistencia al avance (N)
V: velocidad del buque (m/s)

$$BKW = \frac{EKW}{\eta_{prop}\eta_{lineas\,de\,ejes}} = 4387 \text{ kW}$$

η_{prop}: rendimiento propulsivo
$\eta_{lineas\,de\,ejes}$: rendimiento de las líneas de ejes

Suponiendo que el motor no trabajará nunca por encima del 90% de su MCR:

$$BKW^* = \frac{BKW}{\%MCR} = 4875 \text{ kW}$$

Una vez determinada la potencia propulsiva requerida, el siguiente paso es la selección de un motor principal comercial que sea capaz de suministrar dicha potencia de la forma más eficiente posible. Para este estudio, se evaluaron dos opciones del fabricante Wärtsilä, ambas capaces de satisfacer la demanda de 4387 kW operando a un régimen de carga estándar del 84% de su Potencia Máxima Continua (MCR).

Los dos modelos seleccionados fueron:

- Wärtsilä 8V31, con una potencia de 5200 kW.
- Wärtsilä 9L32, con una potencia de 5220 kW.

Aunque ambas opciones son válidas en términos de potencia, el factor decisivo para la selección es el Consumo Específico de Combustible (SFOC), que mide cuántos gramos de combustible consume el motor por kilovatio hora. Un menor SFOC se traduce directamente en un menor consumo y, por tanto, en menores costes operativos y emisiones.

La comparación del SFOC para ambos motores en un régimen de carga cercano al de operación (85%) es la siguiente

Tabla 6: Comparación de consumo específico (SFOC)

Modelo de Motor	Consumo Específico (SFOC) a ~85% de Carga
Wärtsilä 8V31	170,3 g/kWh
Wärtsilä 9L32	184,5 g/kWh

Como se observa en la Tabla 6, el motor Wärtsilä 8V31 presenta una mejor eficiencia de combustible. Esta ventaja se mantiene tanto en el modo de operación estándar (Tier II) como en el modo para Zonas de Control de Emisiones (ECA - Tier III). Por este motivo, se seleccionó el motor Wärtsilä 8V31 como motor principal para el buque de propulsión convencional.

1.2. Balance eléctrico

Además de la potencia requerida para la propulsión, un buque tiene una demanda constante de energía eléctrica para alimentar todos sus sistemas auxiliares, desde los equipos esenciales

en la cámara de máquinas hasta los servicios de habitabilidad para la tripulación. Para calcular esta demanda de forma precisa, se ha desarrollado un balance eléctrico.

El desglose completo y detallado de todos los equipos considerados, sus potencias nominales y los factores de carga aplicados para las diferentes condiciones operativas del buque se puede encontrar en la tabla de balance eléctrico adjunta en los Anexos (pág., 159).

En este capítulo, se procederá a describir el dimensionamiento de los principales grupos de consumidores, que son:

- Consumidores de Cámara de Máquinas
- Sistemas de Agua Salada
- Acomodación y Servicios de Habitabilidad
- Equipos de Manejo de Combustible
- Ventilación
- Equipos de Amarre y Fondeo

Para otros consumidores de menor impacto o cuya carga es más estandarizada, como los equipos de cocina, el sistema de aire acondicionado o las grúas de cubierta, no se ha realizado un dimensionamiento desde cero. En su lugar, se han adoptado valores de potencia de referencia extraídos de trabajos y buques de características similares, los cuales también se encuentran reflejados en la tabla de balance eléctrico final.

Para este balance eléctrico se han considerado cinco condiciones de consumo, las cuales son:

- Navegación
- Maniobra
- Carga y descarga
- Puerto
- Emergencia

Al final de este mismo capítulo y considerando los equipos dimensionados a continuación se presentará la tabla resumen del balance eléctrico junto con la selección de los grupos generadores.

1.3. Sistemas de agua salada

En este apartado se procederá a dimensionar la potencia eléctrica requerida por los equipos que manejan agua salada a bordo. El objetivo es determinar el consumo de los componentes principales que forman parte, directa o indirectamente, de los sistemas de refrigeración, servicios de agua de mar y de los sistemas contraincendios.

El análisis se centra en aquellos equipos que tienen un consumo eléctrico asociado. Esto incluye no solo las bombas que trabajan directamente con agua salada, como las bombas del sistema de refrigeración principal, sino también las bombas de los circuitos cerrados de agua dulce.

1.3.1. Sistema de lastre

El sistema de lastre se dimensionará para poder llenar los tanques de lastre en el mismo tiempo que tarda en cargarse las bodegas de carga, 8 horas. El volumen de los tanques de lastre es de 1000 m^3 por lo que el caudal de la bomba de lastre será de:

$$\boldsymbol{Q} = \frac{V_{lastre}}{t_{lastre}} = \frac{1.000}{8} = 125\ \frac{\mathrm{m}^3}{\mathrm{h}}$$

Se instalarán 3 bombas cubriendo cada una el 100% de la capacidad total. Se supondrá una presión de 2,2 bares, suficiente para superar las perdidas de carga de las tuberías y alcanzar la altura necesaria para el deslastrado.

La potencia de las bombas, suponiendo un rendimiento del 80% será por lo tanto de:

$$Potencia = \frac{\left(\frac{Q}{3600} \cdot P \cdot 100\right)}{0{,}8} = 9{,}54\ \text{kW}$$

Donde :

Q: caudal en $\frac{\mathbf{m^3}}{\mathbf{h}}$

P: presión en bares

1.3.2. Sistema contraincendios

Parte del sistema contraincendios está provisto de bombas de agua salada consumidoras de potencia eléctrica que irán dispuestas en el balance eléctrico. En este apartado se dimensionarán estas mismas para tener un balance eléctrico lo más riguroso posible.

Sistema de rociadores de acomodación

Para el sistema de rociadores en acomodación la normativa SOLAS nos indica que se debe ser capaz de mantener un régimen de $5\frac{\text{l}}{\text{min}\cdot\text{m}^2}$ sobre un área de $280\ \text{m}^2$.

El caudal necesario de la bomba será entonces de:

$$Qroc = 5 \cdot 2800 = 14000\ \text{l/min} = 84\ \text{m}^3/\text{h}$$

Suponiendo que para alcanzar la altura máxima del puente de mando y superar la perdida de carga hasta el punto más lejano del buque se necesitan 3,5 bar se puede calcular la potencia de estas bombas igual que las anteriores como:

$$Potencia = \frac{\left(\frac{Q\left(\frac{\text{m}^3}{\text{h}}\right)}{3600} \cdot P(\text{bar}) \cdot 100\right)}{0{,}8} = 10{,}32\ \text{kW}$$

Sistema de rociadores de cámara de máquinas

Se debe proteger con un caudal de 5 l/min m2 el área de todos aquellos equipos que manejen o quemen combustible, en este caso tendremos los motores generadores y las purificadoras de combustible (dispondremos de dos purificadoras para el combustible MDO).

Sabiendo que el área de los motores 8V31 es 19,1 m2, la de los motores generadores de 6,94 m2 y para las purificadoras consideraremos un área de 10 m2 para las dos, se puede calcular el caudal necesario como:

$$Q = 5 \cdot (19{,}1 + 6{,}94 \cdot 3 + 10) \cdot \frac{3}{50} = 15\ \text{m}^3/\text{h}$$

Suponiendo una presión para alcanzar la altura necesaria y superar las pérdidas de carga de 2 bar se puede calcular la potencia como:

$$Potencia = \frac{\left(\frac{Q\left(\frac{\mathrm{m}^3}{\mathrm{h}}\right)}{3600} \cdot P(\mathrm{bar}) \cdot 100\right)}{0{,}8} = 1\ \mathrm{kW}$$

Bombas principales de contraincendios

Dado que tenemos un buque de carga de más de 1000 toneladas de arqueo bruto y además es un buque de carga, el SOLAS nos obliga a como mínimo disponer de dos bombas de contraincendios, dispondremos de una en cámara de máquinas y otra fuera de la cámara de máquinas.

El caudal de estas bombas debe ser de $Q = \frac{4}{3} Qsent$ y no mayor de 180 m^3/h
El cálculo del caudal de sentinas es el siguiente:

$$Dsent = 1{,}68 \cdot L \cdot (B + D) + 25 = 117\mathrm{mm}$$

La tubería deberá tener unas medidas estándar, por lo tanto utilizaremos un diámetro de 120 mm.

El caudal de sentinas será, por lo tanto:

$$Q = \frac{5{,}75 \cdot D^2}{1000} = 82{,}8\ \mathrm{m}^3/\mathrm{h}$$

$$QCI = \frac{4}{3} \cdot 82{,}8 = 110{,}4\ \mathrm{m}^3/\mathrm{h}$$

Por lo tanto, dispondremos de dos bombas, solo una de ellas en la cámara de máquinas con un caudal de 115 m3/h

La presión del sistema de contraincendios será aquel que permita descargar todas las bocas contra incendios con dos bombas funcionando de manera simultánea descargando a través de una manguera. La presión de descarga para este buque será por SOLAS de $0{,}25\ \mathrm{N/mm^2}$(25 m.c.a.), de acuerdo con la siguiente tabla.

Tabla 7: Presión de descarga en función del tipo de buque y arqueo bruto(López de Asiaín, 2021)

BUQUE PASAJE		BUQUE DE CARGA	
Arqueo bruto < 4000 tons	Arqueo bruto ≥ 4000 tons	Arqueo bruto < 6000 tons	Arqueo bruto ≥ 6000 tons
0,30 N/mm²	0,40 N/mm²	0,25 N/mm²	0,27 N/mm²

Se supondrá por lo tanto una presión de al menos 5 bares para superar las pérdidas de carga de las tuberías y descargar con la presión necesaria.

Se deberá de disponer en el buque de dos bombas principales de contra incendios, una sola de ellas en cámara de máquinas y ambas con la siguiente potencia:

$$Potencia = \frac{\left(\frac{Q\left(\frac{\mathrm{m}^3}{\mathrm{h}}\right)}{3600} \cdot P(\mathrm{bar}) \cdot 100\right)}{0{,}8} = 20\ \mathrm{kW}$$

1.3.3. Sistemas de refrigeración con agua salada

Los sistemas de refrigeración con agua salada podemos dividirlos en sistemas de refrigeración individuales y sistemas de refrigeración centralizados. En este caso el único sistema de refrigeración individual será el del motor propulsor ya que los motores generadores se pueden refrigerar mediante el sistema centralizado puesto que no son de potencias muy elevadas.

Sistemas de refrigeración individuales

El motor propulsor estará provisto de un sistema de refrigeración con agua salada individual. Estos sistemas están formados por una bomba de circulación de agua salada, una bomba de circulación de agua dulce, así como intercambiadores de calor.

El suministrador del motor aporta la información de la bomba de agua dulce, la cual forma el sistema de refrigeración con agua de alta temperatura.

Tabla 8: Datos del sistema de agua de refrigeración de alta temperatura (Wärtsilä, 2025)

High Temperature Cooling Water System	
Pressure at engine, after pump, nom (PT401) (excluding static pressure)	365 kPa
Pressure at engine, after pump, alarm	200 kPa
Pressure at engine, after pump, max	600 kPa
Temperature before cylinders, approx. (TE401)	83 °C
Temperature after cylinders, nom (TE402)	88 °C
Temperature after cylinders without CAC, approx. (TE402)	96 °C
Temperature after charge air cooler, nom (TE432)	96 °C
Temperature after engine, alarm	99 °C
Temperature after engine, stop	110 °C
Capacity of engine driven pump, nom	80 m³/h

Como se puede apreciar en la Tabla 8 el caudal necesario de la bomba de agua dulce es de 80 m3/h. Suponiendo una presión de 2,5 bar se puede obtener la potencia de la bomba, que será de:

$$Potencia = \frac{\left(\frac{Q\left(\frac{\mathrm{m}^3}{\mathrm{h}}\right)}{3600} \cdot P(\mathrm{bar}) \cdot 100\right)}{0{,}8} = 7\ \mathrm{kW}$$

Con los datos de la Tabla 8 también se puede calcular la potencia de calor a disipar:

$$P(\mathrm{kW}) = Q\left(\frac{\mathrm{m}^3}{\mathrm{h}}\right) \cdot (T_{sal} - T_{ent}) \cdot C \cdot \rho \cdot 1/3600 \ = 1207\ \mathrm{kW}$$

Q : caudal de agua ($80\ \mathrm{m}^3/\mathrm{h}$)
P: calor a disipar (kW)
T_{sal}: temperatura de salida del agua (96 ºC)
T_{ent}: temperatura de entrada del agua (83ºC)
C: calor específico del agua dulce(4,18 kJ/kg K)
ρ: densidad del agua dulce(kg/m^3)

Se conocen también datos del sistema de refrigeración con agua de baja temperatura, los cuales se pueden apreciar en la

Tabla 9: Datos del sistema de agua de refrigeración de baja temperatura (Wärtsilä, 2025)

Low Temperature Cooling Water System	
Temperature before engine, nom (TE451)	45 °C
Temperature LT-water outlet, nom (TE472)	69 °C

Con estos datos y conociendo la potencia calorífica que se debe de disipar del motor propulsor podemos calcular el caudal necesario de agua salada para su correcta refrigeración.

$$Q\left(\frac{\mathrm{m}^3}{\mathrm{h}}\right) = \frac{P}{(T_{sal} - T_{ent}) \cdot C \cdot \rho} \cdot 3600 = 93\ \mathrm{m}^3/\mathrm{h}$$

Q : caudal de agua salada(m^3/h)
P: calor a disipar (kW)
T_{sal}: temperatura de salida del agua (40 ºC)
T_{ent}: temperatura de entrada del agua (29ºC)
C: calor específico del agua de mar (4,18 kJ/kg K)
ρ: densidad del agua de mar (kg/m^3)

Suponiendo una presión de 2,5 bares se puede calcular la potencia de esta bomba como:

$$Potencia = \frac{\left(\frac{Q\left(\frac{\mathrm{m}^3}{\mathrm{h}}\right)}{3600} \cdot P(\mathrm{bar}) \cdot 100\right)}{0{,}8} = 8\ \mathrm{kW}$$

Se instalarán dos bombas de refrigeración de agua dulce y otras dos bombas de refrigeración de agua salada.

Sistema de refrigeración centralizado

El sistema de refrigeración centralizada se encargará de la refrigeración de equipos más pequeños que necesiten un sistema de refrigeración con agua. Se dispondrá de dos circuitos, uno abierto de agua salada y uno cerrado de agua dulce. La siguiente tabla muestra los principales equipos que aportan calor al sistema de refrigeración centralizado, así como el número, la potencia calorífica, el factor de simultaneidad y el factor de utilización.

Tabla 10: Balance de calor del buque convencional

	Nº	Hgen (kW)	fsim	fut	Htot
Compresor de gambuza	1	80	1	0,8	64
Compresores HVAC	2	200	0,5	0,8	160
Unidad hidráulica fondeo	1	100	1	0,2	20
Motor 265 kW	3	185	1	0,8	444

Los motores auxiliares que se definirán al final de este balance eléctrico generan un calor de 185 kW (Rolls Royce, 2021). El calor final total que se debe disipar en el sistema es de 688 kW.

La bomba de agua dulce necesitará un caudal de:

$$Q\left(\frac{\mathrm{m}^3}{\mathrm{h}}\right) = \frac{P}{(T_{sal} - T_{ent}) \cdot C \cdot \rho} \cdot 3600 = 93\ \mathrm{m}^3/\mathrm{h}$$

Q : caudal de agua dulce(m^3/h)
P: calor a disipar (kW)
T_{sal}: temperatura de salida del agua (70 ºC)
T_{ent}: temperatura de entrada del agua (41ºC)
C: calor específico del agua dulce (4,18 kJ/kg K)
ρ: densidad del agua dulce(kg/m^3)

Suponiendo una presión necesaria de 2,5 bar la potencia de la bomba será de:

$$Potencia = \frac{\left(\frac{Q\left(\frac{\mathrm{m}^3}{\mathrm{h}}\right)}{3600} \cdot P(\mathrm{bar}) \cdot 100\right)}{0{,}8} = 1{,}8\ \mathrm{kW}$$

La bomba de agua salada se calculará de la siguiente forma:

$$Q\left(\frac{\mathrm{m}^3}{\mathrm{h}}\right) = \frac{P}{(T_{sal} - T_{ent}) \cdot C \cdot \rho} \cdot 3600 = 55\ \mathrm{m}^3/\mathrm{h}$$

Q : caudal de agua salada(m^3/h)
P: calor a disipar (kW)
T_{sal}: temperatura de salida del agua (40 ºC)
T_{ent}: temperatura de entrada del agua (29ºC)
C: calor específico del agua de mar (4,18 kJ/kg K)
ρ: densidad del agua de mar (kg/m^3)

Suponiendo una presión de al menos 2,5 bares la potencia de la bomba será de:

$$Potencia = \frac{\left(\frac{Q\left(\frac{\mathrm{m}^3}{\mathrm{h}}\right)}{3600} \cdot P(\mathrm{bar}) \cdot 100\right)}{0{,}8} = 4{,}6\ \mathrm{kW}$$

Se instalarán por lo tanto dos bombas de agua dulce y dos bombas de agua salada para el sistema de refrigeración centralizado.

1.4. Sistemas de acomodación

Para el sistema de acomodación se dimensionarán la bomba para el sistema de agua fría sanitaria, la bomba de agua dulce para el mineralizador, la bomba de agua caliente y el calentador eléctrico de agua caliente.

Bomba de agua fría sanitaria

El caudal será el necesario para dar 180 litros a 22 tripulantes en media hora.

$$Q = nº\ tripulantes \cdot 180\,\mathrm{l} \cdot 0{,}5 \cdot \frac{1}{1000} = 1{,}98\ \mathrm{m}^3/\mathrm{h}$$

Con una presión de 4 bares se necesitará una potencia de 0,3 kW.

Bomba de agua dulce del mineralizador

Se instalará una bomba para el mineralizador con las mismas características que la bomba de agua fría sanitaria, de 2 m3/h, 4 bares de presión y por lo tanto, 0,3 kW de potencia.

Calentador de agua sanitaria

De acuerdo con la Tabla 11 y conociendo el número de personas a bordo se establece una potencia calorífica de 12 kW.

Tabla 11: Potencia calorífica para el agua caliente sanitaria (López de Asiaín, 2021)

Numero de personas a bordo	Volumen (l)	Potencia calorifica (kW)
1-9	200	6
10-29	300	12
30-49	450	47
50-69	750	70
70-99	750	116
100-149	1000	163
150-200	1500	233

Bomba de agua caliente sanitaria

En la Tabla 11 se puede apreciar la cantidad de agua necesaria para 22 tripulantes por hora, por lo que se necesitará un caudal de $0{,}3\ \mathrm{m^3/h}$ y con una presión de 3 bar se obtiene una bomba de una potencia inferior a 0,1 kW.

Bomba de aguas residuales

La bomba de aguas residuales se dimensionará para vaciar el tanque de aguas residuales en 4 horas como máximo. Por lo tanto, debemos calcular la capacidad del tanque de aguas residuales para obtener la potencia de las bombas.

El tanque de aguas residuales se dimensiona para almacenar 300 l de aguas por persona durante un periodo de 2 días para todos los tripulantes a bordo.

$$C\ aguas\ residuales = \frac{C \cdot Nº\ per \cdot D}{1000} = \frac{300 \cdot 21 \cdot 2}{1000} = 13{,}2\ \mathrm{m^3}$$

C: generación de aguas residuales por persona y día
$Nº\ per$: número de tripulantes a bordo del buque
D: días que se debe almacenar los residuos, días de estancia en puerto

La capacidad de las bombas de aguas residuales será de:

$$Q = \frac{Vaguas\ residuales}{t} = \frac{13{,}2}{4} = 3{,}3\ \mathrm{m^3/h}$$

Suponiendo una presión necesaria de 7 bar para superar las pérdidas de carga y la presión mínima de descarga de 1,5 bar se calcula una potencia de las bombas de 0,8 kW. Se instalarán dos bombas de aguas residuales.

1.5. Equipos del motor principal

En este apartado se van a dimensionar los equipos tanto de combustible como de aceite necesarios para un correcto funcionamiento del motor principal. Todos los cálculos se han hecho de acuerdo a los datos y fórmulas provistos por el fabricante en la guía de funcionamiento

del motor (Wärtsilä, 2025). No se va a hacer hincapié en las fórmulas o datos específicos puesto que son independientes de cada motor y fabricante.

Los equipos de manejo de combustible y sus características necesarias para el funcionamiento del motor principal son los siguientes:

Tabla 12: Equipos de manejo de combustible del motor principal

Equipos de manejo de fuel	Nº	Caudal (l/h)	Presión (bar)	P (kW)
Separator feed pump	1	1292,04706	5	0,22
circulation pumps LFO	2	2700	16	1,5
Circulation pumps booster unit	1	2970	16	1,65
Fuel feed pump booster unit	1	908,574118	16	0,50
Separator pre heater	1			15,20

Los equipos para una correcta lubricación y manejo de aceite del motor principal son los siguientes:

Tabla 13: Equipos de manejo de aceite del motor principal

Lubrication Oil system	Nº	Caudal (l/h)	Presión (bar)	P (kW)
Separator feed pump	1	1526,08696	2	0,10
Lubrication Oil pump	1	92000	8	25,55
Prelubricationg oil pump	1	92000	2	6,38

1.6. Sistemas de aceite

Los equipos consumidores de energía eléctrica del sistema de aceite son la bomba de trasiego de aceite y la bomba de circulación de aceite. La bomba de circulación de aceite ya se ha calculado anteriormente para el motor principal (Tabla 13). La bomba de trasiego de aceite, encargada de suministrar el aceite que se consume al motor es la que se calculará en este apartado.

Tabla 14: Datos del sistema de lubricación del motor principal (Wärtsilä, 2025)

Lubricating Oil System	
Pressure before bearings, nom (PT201)	420 kPa
Pressure before bearings, alarm	300 kPa
Pressure before bearings, stop	200 kPa
Suction ability, including pipe loss, max	40 kPa
Priming pressure, nom (PT201)	100 kPa
Priming pressure, alarm	80 kPa
Temperature before bearings, nom (TE201)	75 °C
Temperature before bearings, alarm	85 °C
Temperature after engine, approx.	82 °C
Pump capacity (main), engine driven	144 m^3/h
Pump capacity (main), electrically driven	92 m^3/h
Oil volume in separate system oil tank	5.6 m^3
Maximum oil volume in wet sump (engine stopped)	2.8 m^3
Minimum oil volume in wet sump (engine stopped)	2.4 m^3
Filter fineness	30 microns
Filter difference pressure alarm	120 kPa
Oil consumption at 100% load, approx	0.45 g/kWh

Para calcular la capacidad de la bomba de trasiego de aceite de cilindros es necesario primero calcular la capacidad del tanque de servicio diario de este mismo aceite.

El fabricante nos da la información del consumo de aceite al 100% de la capacidad del motor, supondremos que es el mismo consumo para nuestra condición de funcionamiento. Sabiendo que la potencia consumida por el motor en navegación es de 4388 kW, el consumo de aceite en g/kw h, la densidad del aceite ($0{,}89\ \mathrm{ton/m^3}$)y el tiempo en que se tarda en llenar el tanque de servicio diario de aceite de cilindros (8 horas) podemos calcular la capacidad del tanque de la siguiente forma:

$$C = \frac{Consumo\left(\frac{\mathrm{g}}{\mathrm{kW \cdot h}}\right) * horas * Potencia(\mathrm{kW})}{densidad\left(\frac{\mathrm{g}}{\mathrm{m^3}}\right)} = 0{,}02\ \mathrm{m^3}$$

La bomba de trasiego de aceite se dimensionará para llenar la bomba de trasiego de aceite en 2 horas.

$$Q\left(\frac{\mathrm{m^3}}{\mathrm{h}}\right) = \frac{C(\mathrm{m^3})}{2\ (\mathrm{horas})} = 0{,}009\ \mathrm{m^3/h}$$

Esta bomba con una presión de 1 bar hace una potencia de menos de 1 kW.

Otros equipos necesarios de dimensionar son los tanques de aguas aceitosas y los tanques de aceite sucio. Aunque estos no consuman potencia eléctrica ya que son simples tanques de almacenamiento sin ellos no seríamos capaces de dimensionar la bomba de lodos, la cual se obtendrá más adelante.

Tanque de aguas aceitosas

Para un buque con una potencia de motor principal de 5200 kW la capacidad del tanque de aguas aceitosas se define mediante la siguiente fórmula:

$$C = 20 \cdot D \cdot \frac{P}{10^6} = 1{,}56\ \mathrm{m^3}$$

D: días de navegación (15 días)
P: potencia del motor principal (5200 kW)

Tanque de aceite sucio

El tanque de aceite sucio se dimensiona en función de la necesidad de aceite de cada motor, la cual se puede apreciar en la siguiente tabla.

Tabla 15: Capacidad de aceite de los motores

Modelo de motor	Volumen de aceite (m3)
Wartsila 8V31	5,6
***mtu* 6R0150 DS300**	0,04
***mtu* 4R0113 DS100**	0,012

La capacidad del tanque de aceite sucio será por lo tanto de 5,73 m3

1.7. Sistema de sentinas

En el apartado de sistemas de agua salada se calcula el caudal de sentinas, el cual es de 82,8 m3/h, suponiendo una presión de 3,5 bar se obtiene una potencia de la bomba de 10 kW

La bomba auxiliar de sentinas debe tener un 5% de la capacidad de la bomba principal de sentinas, por lo tanto tendrá un caudal de 4,14 m3/h y una presión igual de 3,5 bar, lo que hace una potencia de 0,5 kW

El tanque de sentinas, puesto que nuestro buque lleva una potencia total instalada de 6095 kW se calculará mediante la siguiente fórmula:

$$C = 1{,}5 + \frac{(P - 1000)}{1500} = 4{,}89 \text{ m}^3$$

P: potencia instalada (kW)

1.8. Sistema de lodos

El sistema de lodos es el encargado de almacenar los residuos de la purificadora de combustibles y del separador de sentinas que se extrae en la limpieza del combustible, por lo que para calcular la capacidad del tanque de lodos así como la potencia de la bomba es necesario calcular los consumos en cada condición de operación. Dado que esto forma parte del final de este capítulo se dejará el desarrollo para entonces y en este apartado se usarán los valores que se verán más adelante.

Por lo tanto, suponiendo una purificación del combustible (k=0,01) y 15 días de navegación la siguiente tabla nos muestra la capacidad necesaria del tanque para los lodos.

Tabla 16: Generación de lodos por condición de operación

Condición	Consumo T/día	C lodos (m3)	V lodos (m3)
Navegación	20,1	3	3,14
Maniobra	2,44	0,024	0,025
C/D	3,49	0,069	0,072
Puerto	1,66	0,033	0,034

La capacidad total del tanque de lodos será de 2,3 m3.

La capacidad de la bomba se calculará entonces de la siguiente forma:

$$Q = \frac{Vsent + Vlodos + Vaacei + Vaces}{t} = 3{,}93 \text{ m}^3/\text{h}$$

$Vsent$: volumen del tanque de sentinas (m3)
$Vaacei$: volumen del tanque de aguas aceitosas (m3)
$Vaces$: volumen del tanque de aceite sucio (m3)
t: tiempo de descarga (h)

Suponiendo una presión de la bomba de lodos de 5 bares la potencia de la bomba será de 0,68 kW.

1.9. Ventilación

El sistema de ventilación de la cámara de máquinas se dimensionará para poder suministrar el aire necesario para la combustión y dispersar el calor generado por los diferentes equipos dispuestos en la cámara de máquinas, además de para garantizar una renovación adecuada del aire en el interior de la cámara de máquinas para crear una atmósfera respirable. Todas las entradas de aire al sistema contarán con rejillas regulables, además el sistema contará con

equipos de cierre de accionamiento remoto contra incendios como fire dampers en todas las entradas y salidas de aire.

El sistema estará compuesto por un circuito de impulsión que introducirá aire en el interior del local gracias a unos ventiladores axiales de doble velocidad. También contará con un circuito de extracción que extraerá el aire caliente del local de manera natural por el guardacalor.

Para dimensionar el sistema realizaremos el cálculo de los ventiladores, para ello deberemos estudiar las diferentes condiciones de funcionamiento del buque, navegación, maniobra, carga y descarga y finalmente la condición de puerto y de esta manera poder suministrar aire para la condición más exigente. El cálculo se realizará de acuerdo a la norma ISO 8861, para ello deberemos calcular en cada condición el caudal de aire necesario para la combustión (qc) y el caudal de aire necesario para disipar el calor generado por los equipos (qh), motores generadores, alternadores, motores eléctricos, tuberías de escape y la instalación eléctrica. Al no disponer de calderas no deberemos calcular el calor generado por estas ni el de las tuberías de vapor. Dado que no se utilizan combustibles que deban ser calentados para su bombeo los tanques no estarán calefactados y tampoco emitirán calor a la cámara de máquinas. Una vez determinados dichos caudales se elegirá el valor mayor de los siguientes valores:

$$Q = qc + qh$$
$$Q = 1{,}5 \cdot qc$$

Como se explicará mas adelante la generación de potencia eléctrica estará definida por tres grupos generadores de 265 kW y un grupo generador de 100 kW para la condición de emergencia el cual no irá instalado en la cámara de máquinas.

1.9.1. Caudal de aire necesario para la combustión

La siguiente tabla detalla el cálculo del caudal de aire de combustión requerido por los grupos generadores en cada condición operativa, utilizando los datos de consumo de aire especificados por el fabricante del motor.

Tabla 17: Caudal de aire necesario para la combustión en grupos auxiliares

Caudal de aire necesario para la combustión en auxiliares				
265 kW al 100%	23	m3/min	0,44	kg/s
100 kW al 100%	8,2	m3/min	0,16	kg/s
Condición de operación	Navegación	Maniobra	C/D	Puerto
Potencia consumida	4783,89	447,70	641,32	287,81
Porcentaje de potencia	75%	84%	81%	79%
Caudal para la combustión 265 kW	0,66	0,74	1,04	0,35
Caudal para la combustión 100 kW				
Caudal total necesario (m3/s)	0,57	0,65	0,91	0,30

A continuación, en la Tabla 18, se presenta el cálculo del caudal de aire necesario para la combustión del motor principal. Los valores utilizados para este cálculo provienen de las especificaciones técnicas del fabricante.

Tabla 18: Caudal de aire necesario para la combustión del motor principal

Caudal de aire necesario para la combustión en principal		
5200 kW al 100%	7,3	kg/s
Condición de operación	Navegación	Maniobra

Potencia consumida	4388	
Porcentaje de potencia	84%	0%
Caudal para la combustión principal	7,3	0
Caudal total necesario (m3/s)	6,34	0

Dado que los motores auxiliares son refrigerados por aire, la necesidad de ventilación debe ser absorbida por el sistema de ventilación. La siguiente tabla detalla el caudal de aire necesario para esta función.

Tabla 19: Necesidad de ventilación de los motores auxiliares

Necesidades de ventilación motores auxiliares				
265 kW al 100%	517,7	m3/min	9,92	kg/s
100 kW al 100%	187	m3/min	3,58	kg/s
Condición de operación	Navegación	Maniobra	C/D	Puerto
Caudal (m3/s)	17,26	17,26	25,89	8,63

A continuación, en la Tabla 20, se detalla el calor irradiado por el motor principal. Este valor se basa en los datos proporcionados por la ficha técnica del fabricante del motor.

Tabla 20: Calor irradiado por el motor prinicpal

Calor emitido por el motor principal		
calor al 100%	168	kW
Condición de operación	Navegación	
Calor (kW)	141,12	

Además del motor principal, los motores auxiliares son una fuente de calor importante. En la Tabla 21 se detalla el calor irradiado por los generadores de 265 kW en funcionamiento.

Tabla 21: Calor emitido por los motores generadores (265 kW)

Calor emitido por motor auxiliar 265 kW				
Calor al 100%	36,9	kW		
Condición de operación	Navegación	Maniobra	C/D	Puerto
Calor (kW)	73,8	73,8	110,7	36,9

La siguiente tabla cuantifica el calor generado por las tuberías de escape de los motores en funcionamiento.

Tabla 22: Calor emitido por las tuberías de escape

Calor emitido por las tuberías de escape				
Condición de operación	Navegación	Maniobra	C/D	Puerto
L tub esc Rolls Royce	30	30	45	15
L tub esc wartsila	20			
Calor radiado por m Rolls Royce (	0,1	0,1	0,1	0,1
Calor radiado por m wartsila (700 mm, t=250 K)	0,35	0,35	0,35	0,35
Calor (kW)	10	3	4,5	1,5

Finalmente, se calcula el calor disipado por la propia instalación eléctrica. Para ello, se ha estimado que esta carga térmica equivale al 20% de la potencia eléctrica total consumida por los equipos situados en la cámara de máquinas. La siguiente tabla muestra el resultado de este cálculo.

Tabla 23: Calor emitido por la instalación eléctrica

Calor emitido por la instalación eléctrica				
Condición de operación	Navegación	Maniobra	C/D	Puerto
P no dedicada a la prop en ccmm	395,89	447,70	641,32	208,96
Calor	79,18	89,54	128,26	41,79

Una vez calculadas todas las fuentes de calor y el aire de combustión, la siguiente tabla resume los caudales de aire requeridos para cada condición operativa. Según la normativa ISO 8861, el caudal de diseño para la ventilación de la cámara de máquinas debe ser el valor máximo entre la suma del caudal necesario para la combustión y el caudal necesario para la disipación de calor y 1,5 veces el caudal de la combustión

Tabla 24: Resumen de caudales de ventilación requeridos

Caudal necesario para disipar el calor				
Condición de operación	Navegación	Maniobra	C/D	Puerto
Calor total	304,10	180,95	265,20	87,67
Caudal para combustión (m3/s)	6,92	0,65	0,91	0,30
Caudal para calor (m3/s)	35,43	29,46	43,79	14,55
Q=qc+qh (m3/s)	42,35	30,11	44,70	14,85
Q=qc+qh (m3/h)	152474,75	108389,63	160902,28	53453,19
Q=1,5*qc (m3/s)	10,38	0,97	1,36	0,45

A partir del caudal máximo requerido en la condición más exigente, se ha procedido a la selección final de los equipos. Se instalará un sistema compuesto por cuatro ventiladores axiales de 60,000 m³/h cada uno, con una potencia asociada de 11.9 kW. Estos ventiladores serán de doble velocidad para poder ajustar el caudal a las necesidades de cada condición y optimizar el consumo energético. Además, al menos uno de ellos será reversible, permitiendo su uso como extractor en caso de necesidad o emergencia.

1.10. Selección de motores generadores y consumo de combustible

1.10.1. Selección de motores generadores

A partir del dimensionamiento de los consumidores eléctricos detallado en los apartados anteriores y de los demás equipos que se pueden encontrar en el balance del anexo (pág., 159), se ha elaborado la tabla resumen del balance de cargas eléctricas. Esta tabla cuantifica la potencia eléctrica total requerida por el buque en cada una de las cinco condiciones de operación

Tabla 25: Resumen del balance eléctrico del buque con propulsión convencional

Condición de operación	Navegación	Maniobra	Carga y Descarga	Puerto	Emergencia
Potencia (kW)	395	447	641	287	98

Con la demanda eléctrica ya definida se procede a la selección de los grupos electrógenos. Para garantizar la redundancia operativa y la seguridad, se ha optado por una configuración de múltiples generadores.

La planta eléctrica estará compuesta por tres generadores principales de 265 kW cada uno, modelo Rolls Royce MTU 6R0150 DS300. Adicionalmente, se instalará un generador de emergencia de 100 kW, modelo Rolls Royce MTU 4R0113 DS100, que, en cumplimiento con la normativa SOLAS, irá dispuesto fuera de la cámara de máquinas.

Para asegurar una operación eficiente y una larga vida útil de los equipos, es importante verificar que los motores operen dentro de su rango de carga óptimo, que generalmente se sitúa entre el 70% y el 90% de su capacidad. La siguiente tabla muestra el número de generadores en servicio y el porcentaje de carga de cada uno para satisfacer la demanda en cada condición.

Tabla 26: Régimen de funcionamiento de los motores auxiliares

Régimen de funcionamiento	Navegación	Maniobra	C y D	Puerto	Emergencia
265 kW	75 %	85 %	80 %	78 %	0 %
265 kW	75 %	85 %	80 %	0 %	0 %
265 kW	0 %	0 %	80 %	0 %	0 %
100 kW	0 %	0 %	0 %	78 %	98 %

Como se puede observar en la tabla, en las condiciones más habituales, el número de generadores en funcionamiento se ajusta para que la carga sobre ellos se mantenga dentro del rango óptimo. En el caso del generador de emergencia, este opera a un régimen superior, lo cual es aceptable dado que su uso es esporádico y de corta duración, primando la capacidad de entregar la potencia necesaria sobre la eficiencia a largo plazo.

1.10.2. Consumo de combustible

Una vez seleccionados tanto el motor principal como los generadores auxiliares, el paso final es calcular el consumo total de combustible del buque de referencia. Este consumo se desglosa en el del motor principal, que opera únicamente en la condición de navegación, y el de los motores auxiliares, que varía según la condición operativa del buque.

El mayor consumidor de combustible es, con diferencia, el motor principal durante la navegación en alta mar. Como se estableció en el apartado anterior, se seleccionó un motor Wärtsilä 8V31 que opera a una potencia de 4368 kW para mantener la velocidad de servicio. Con un consumo específico (SFOC) de HFO de 170,3 g/kWh (Wärtsilä, 2025) a ese régimen, el consumo diario para la propulsión es:

$$Consumo_{propulsión} = 4368\ \text{kW} \cdot 170{,}3\frac{\text{g}}{\text{kWh}} \cdot 24\frac{\text{h}}{\text{día}} \cdot \frac{1}{0{,}85}\frac{\text{m}^3}{\text{ton}} \cdot \frac{1}{10^6}\frac{\text{ton}}{\text{g}} = 21{,}1\frac{\text{m}^3}{\text{día}}$$

Puesto que el consumo de LFO es de 172,38 g/kWh haciendo el mismo cálculo se obtiene que el consumo día es de $21{,}4\ \frac{m^3}{día}$.

Este consumo solo se aplica a la condición de "Navegación". En el resto de las condiciones (maniobra, puerto, etc.), se considera que el motor principal está apagado.

El consumo de los generadores auxiliares depende directamente de la demanda eléctrica en cada una de las cinco condiciones operativas definidas en el balance de cargas. Utilizando la potencia requerida para cada condición y el consumo específico (SFOC) de los generadores

Rolls Royce MTU seleccionados, se calcula el combustible necesario para la generación eléctrica.

De acuerdo con los datos del fabricante los motores tienen los siguientes consumos en sus respectivos regímenes de operación:

Tabla 27: Consumos de los motores auxiliares (Rolls Royce, 2021)

Régimen de funcionamiento	Consumo del motor 265 kW	Consumo del motor 100 kW
100 %	71,7 l/h	31 l/h
75 %	52,9 l/h	25 l/h

Basándonos en el número de generadores en servicio y su régimen de funcionamiento en cada condición podemos calcular el consumo en m3/día de los generadores auxiliares:

- Condición de Navegación: Para satisfacer la demanda eléctrica en alta mar, se requiere el funcionamiento de dos de los generadores principales de 265 kW. En esta condición, ambos motores operan a un 75% de su carga nominal, un punto de alta eficiencia. Utilizando los datos de consumo del fabricante, esto se traduce en un consumo diario de 2.53 m^3/día.

- Condición de Maniobra: Durante la entrada y salida de puerto, la demanda aumenta por el uso de equipos como las hélices de maniobra. Para cubrir este pico, se utilizan dos generadores de 265 kW, operando a un régimen superior del 85%. Interpolando los datos de consumo del fabricante, el combustible requerido en esta condición asciende a 2.88 m^3/día.

- Condición de Carga y Descarga: Este es el escenario de mayor demanda eléctrica, al tener que alimentar la maquinaria de cubierta. Para ello, se ponen en servicio los tres generadores principales de 265 kW, trabajando a un 80% de su capacidad. El consumo de combustible resultante para esta operación es el más elevado, alcanzando los 4.11 m^3/día.

- Condición de Puerto: Durante la estancia en puerto, la demanda eléctrica es menor. Para optimizar la eficiencia, se utiliza una configuración mixta de un generador de 265 kW junto con el generador auxiliar de 100 kW, operando ambos a un 78% de su carga respectiva. Esta configuración resulta en el consumo más bajo, con un valor de 1.95 m^3/día.

La siguiente tabla presenta un resumen del consumo total de combustible en toneladas por día para cada una de las condiciones operativas analizadas. Este es el perfil de consumo final que se utilizará como base de comparación en el estudio económico.

Tabla 28: Consumo total de combustible diario por condición de operación

Condición operativa	Consumo motor principal (m^3/día)	Consumo motores auxiliares (m^3/día)	Consumo total (m^3/día)
Navegación	21,1 HFO/21,4 LFO	2,53	23,65
Maniobra	0	2,88	2,88
C y D	0	4,11	4,11
Puerto	0	1,95	1,95

2. Buque propulsado a vela

El perfil energético del buque propulsado a vela es diferente al de su homólogo convencional. La diferencia más significativa y la principal ventaja de este diseño es la eliminación completa del consumo de combustible para la propulsión, ya que esta es provista íntegramente por el viento.

Por lo tanto, el análisis de su consumo se centra exclusivamente en la demanda de energía eléctrica para los servicios a bordo. Aunque muchos de los equipos son comunes a ambos buques, la ausencia de un motor principal y la inclusión de sistemas para la operación de las velas modifican sustancialmente el balance de cargas eléctricas, como se detallará a continuación.

2.1. Diferencias principales en el balance eléctrico

A continuación, en la Tabla 29, se presenta una comparativa de las potencias eléctricas instaladas para los equipos cuyos cambios son más directos. Sistemas más complejos como la ventilación, sistemas de aceite o sistemas de lodos, que requieren un rediseño completo, se analizarán en el siguiente apartado.

Tabla 29: Cambios del balance eléctrico (Buque a vela vs. Convencional)

Sistema/ Equipo	Potencia buque convencional (kW)	Potencia buque a vela	Observaciones
Sistemas de agua salada			
Bombas de lastre	9,54	9,54	Se mantienen igual
Contraincendios acomodación	10,3	10,3	Se mantienen igual
Rociadores cc. mm.	1	0,4	Se reduce el area de los equipos que manejan combustible de 50 a 17 m2
Bombas ppales de CI	19	19	Se mantiene igual
Refrigeración individual			No hay sistemas de refrigeración indi
Refrigeración centralizada			No hay cambios, en las especificaciones de los motores ambos irradian el mismo calor al refrigerante.
Sistemas de acomodación			Los sistemas de acomodación se mantienen iguales
Equipos del motor principal			Los equipos del motor principal se eliminan

2.2. Rediseño del sistema de aceite

La eliminación del motor principal simplifica los sistemas de aceite a bordo. En primer lugar, se suprimen todos los equipos dedicados al manejo de aceite de cilindros del motor propulsor, incluyendo sus bombas de trasiego y alimentación.

Como consecuencia directa, la generación de residuos oleosos es mucho menor. Esto permite redimensionar a la baja la capacidad de los tanques de aguas aceitosas y del tanque de aceite sucio. A su vez, una menor capacidad de almacenamiento de residuos implica un menor sistema de tratamiento de lodos, con el correspondiente ahorro en potencia eléctrica instalada.

Tanque de aguas aceitosas

Para dimensionar el tanque de aguas aceitosas del buque a vela, se ha utilizado la misma fórmula de cálculo que en el caso del buque convencional. Sin embargo, los parámetros de entrada se adaptan a las nuevas características de la propulsión.

Como número de días sin posibilidad de descarga, se ha tomado el valor más conservador posible: el tiempo de viaje máximo registrado en las simulaciones del capítulo anterior, correspondiente a 48 días. En cuanto a la potencia instalada, al no existir motor principal, se considera únicamente la de la planta eléctrica auxiliar, que suma un total de 730 kW. Con estos nuevos datos, se procede a recalcular la capacidad requerida para el tanque.

$$C = 20 \cdot D \cdot \frac{P}{10^6} = 0{,}73\ \mathrm{m}^3$$

D: días de navegación (48 días)
P: potencia del motor principal (730 kW)

Tanque de aceite sucio

Para el dimensionamiento del tanque de aceite sucio, se ha seguido el criterio de que este debe ser capaz de albergar el volumen total de aceite contenido en los cárteres de todos los motores diésel a bordo.

Considerando una capacidad de 40 litros para cada uno de los tres generadores principales de 210 kW y 12 litros para el generador de emergencia de 100 kW, la capacidad mínima requerida para el tanque se calcula como la suma de estos volúmenes:

$$Vtotal = (3 \times 40\mathrm{L}) + 12\mathrm{L} = 132\ \mathrm{L}$$

La capacidad del tanque de aceite sucio será de 132 litros.

2.3. Rediseño del sistema de sentinas

El sistema de sentinas del buque a vela es análogo al del buque convencional en cuanto a sus bombas, pero presenta una diferencia en el volumen de almacenamiento.

Dado que el caudal de sentinas a evacuar no se ve modificado por el tipo de propulsión, se mantendrán las mismas bombas de sentinas, tanto la principal como la auxiliar, con las mismas capacidades que las dimensionadas para el buque de referencia.

Sin embargo, la capacidad del tanque de sentinas sí debe ser recalculada. Según la normativa, este volumen depende de la potencia total instalada en la cámara de máquinas. Al no contar con un motor propulsor, la potencia total en este espacio es significativamente menor. Por tanto, se procederá a recalcular la capacidad requerida para el tanque utilizando la misma fórmula que para el buque convencional, pero introduciendo el nuevo y reducido valor de potencia instalada.

$$C = 1{,}5 + \frac{(P - 1000)}{1500} = 1{,}32\ \mathrm{m}^3$$

P: potencia instalada (730 kW)

2.4. Rediseño del sistema de lodos

Como se ha explicado, la capacidad del tanque de lodos se calcula considerando el consumo de combustible, un factor k de 0.01 al tratarse de combustible purificado y un ciclo operativo total entre descargas.

Suponiendo un periodo combinado de navegación, puerto, maniobra y carga/descarga de 53 días, en la siguiente tabla se presenta el volumen de lodos generado para cada condición. Los consumos de combustible específicos se determinarán en apartados posteriores. La suma de estos volúmenes definirá la capacidad total requerida para el tanque.

Tabla 30: Capacidad del tanque de lodos del buque a vela

Condición	Consumo T/día	C lodos (m3)	V lodos (m3)
Navegación	2,16	1,04	1,08
Maniobra	2,45	1,18	1,23
C/D	3,45	1,66	1,72
Puerto	0,98	0,47	0,49
Total		4,34	4,52

La capacidad de la bomba se calculará entonces de la siguiente forma:

$$Q = \frac{Vsent + Vlodos + Vaacei + Vaces}{t} = 1{,}68\ \mathrm{m^3/h}$$

$Vsent$: volumen del tanque de sentinas (m3)
$Vaacei$: volumen del tanque de aguas aceitosas (m3)
$Vaces$: volumen del tanque de aceite sucio (m3)
t: tiempo de descarga (h)

Suponiendo una presión de la bomba de lodos de 5 bares la potencia de la bomba será de 0,3 kW.

2.5. Rediseño del sistema de ventilación

El rediseño del sistema de ventilación para el buque a vela se realiza de forma análoga al del buque convencional, siguiendo la misma normativa ISO. Sin embargo, la ausencia del motor principal modifica drásticamente los requerimientos de aire de la cámara de máquinas.

Las principales diferencias en el cálculo son dos:

- El caudal de aire para la combustión (qc) se reduce de forma muy significativa, ya que ahora solo debe satisfacer la demanda de los generadores auxiliares en funcionamiento.

- La carga térmica a disipar (qh) es mucho menor, al eliminarse la principal fuente de calor del local: el motor propulsor.

El calor emitido por el resto de equipos (generadores, sistemas eléctricos, etc.) se calcula utilizando la misma metodología que en el apartado anterior. A continuación, las siguientes tablas presentan los nuevos cálculos del calor total a refrigerar y del aire de combustión necesario para cada condición operativa.

La siguiente tabla resume el calor necesario para la combustión en cada condición.

Tabla 31: Caudal de aire necesario para la combustión de los motores auxiliares de 210 kW

Caudal de aire necesario para la combustión en auxiliares

210 kW al 100%	23	m3/min	0,44	kg/s
100 kW al 100%	8,2	m3/min	0,15	kg/s
	Navegación	Maniobra	C/D	Puerto
Potencia consumida	4700,386	355,292	334,478	284,416
Porcentaje de potencia	74%	85%	80%	68%
Caudal para la combustión 210 kW	0,65	0,74	0,89	0,29
Caudal para la combustión 100 kW				
Caudal total necesario (m3/s)	0,57	0,64	0,77	0,25

A continuación, se muestra la ventilación necesaria para los motores generadores

Tabla 32: Necesidad de ventilación de los motores auxiliares de 210 kW

Necesidades de ventilación motores auxiliares				
210 kW al 100%	517,7	m3/min	9,92	kg/s
100 kW al 100%	187	m3/min	3,58	kg/s
Condición de operación	Navegación	Maniobra	C/D	Puerto
Caudal (m3/s)	17,25	17,25	17,25	8,63

En la siguiente tabla se determina el calor irradiado en cada condición por los motores auxiliares.

Tabla 33: Calor emitido por los motores auxiliares de 210 kW

Calor emitido por motor auxiliar 210 kW				
Calor irradiado por un motor	36,9	kW		
Condición de operación	Navegación	Maniobra	C/D	Puerto
Calo (kW)	73,8	73,8	73,8	73,8

Se puede apreciar a continuación el calor emitico por las tuberías de escape.

Tabla 34: Calor emitido por las tuberías de escape en el buque propulsado a vela

Calor emitido por las tuberías de escape				
	Navegación	Maniobra	C/D	Puerto
L tub esc Rolls Royce	30	30	30	30
Calor radiado por m Rolls Royce (	0,1	0,1	0,1	0,1
Calor radiado por m wartsila (700 mm, t=250 K)	0,35	0,35	0,35	0,35
Calor (kW)	3	3	3	3

El calor emitido por la instalación eléctrica que se aprecia en la siguiente tabla se puede determinar igual que para el buque convencional.

Tabla 35: Calor emitido por la instalación eléctrica en cc. mm. del buque propulsado a vela

Calor emitido por la instalación eléctrica				
	Navegación	Maniobra	C/D	Puerto
P no dedicada a la prop en ccmm	312,386	355,292	334,478	284,416

Calor	62,4772	71,0584	66,8956	56,8832

La siguiente tabla resumen muestra el caudal de aire necesario en cada condición para disipar el calor generado, así como introducir el suficiente aire para que se genere una correcta combustión en los motores.

Tabla 36: Caudal de aire necesario en cada condición del buque propulsado a vela

Caudal necesario para disipar el calor				
Condición de operación	Navegación	Maniobra	C/D	Puerto
Calor total	139,28	162,47	158,30	148,29
Caudal para combustión (m3/s)	0,57	0,65	0,78	0,26
Caudal para calor (m3/s)	26,62	28,19	27,85	27,37
Q=qc+qh (m3/s)	27,19	28,84	28,63	27,63
Q=qc+qh (m3/h)	97890,20	103808,93	103057,99	99453,95
Q=1,5*qc (m3/s)	0,86	0,97	1,17	0,39

Se instalarán por lo tanto 3 ventiladores axiales de 55 m3/h y 11 kW de doble velocidad y al menos uno de ellos reversible.

2.6. Selección de motores generadores y consumo de combustible

2.6.1. Selección de motores generadores

De forma análoga al buque convencional, una vez consolidado el balance eléctrico del buque a vela, se procede a seleccionar la planta de generadores que satisfaga la nueva demanda de potencia, la cual se resume en la siguiente tabla.

Tabla 37: Balance eléctrico del buque propulsado a vela

Condición de operación	**Navegación**	**Maniobra**	**Carga y Descarga**	**Puerto**	**Emergencia**
Potencia (kW)	312	355	334	284	98

Dada la menor demanda eléctrica máxima en comparación con el buque convencional (al no existir los consumidores asociados al motor principal, así como las grúas de carga y descarga que son sustituidas por velas rígidas), la planta de generación se puede optimizar significativamente.

Para el buque a vela, se ha seleccionado una planta eléctrica compuesta por tres generadores principales de 210 kW (modelo Rolls Royce MTU 6R0150 DS230), en lugar de los tres de 265 kW del buque de propulsión convencional. Se mantiene, eso sí, el mismo generador de emergencia de 100 kW (modelo Rolls Royce MTU 4R0113 DS100), situado fuera de la cámara de máquinas.

La siguiente tabla analiza cómo esta nueva configuración responde a las demandas de cada condición, mostrando el régimen de funcionamiento de los generadores.

Tabla 38: Régimen de funcionamiento de los motores del buque a vela

Régimen de funcionamiento	**Navegación**	**Maniobra**	**C y D**	**Puerto**	**Emergencia**
210 kW	74 %	84 %	79 %	91 %	0 %
210 kW	74 %	84 %	79 %	0 %	0 %

210 kW	0 %	0 %	0 %	0 %	0 %
100 kW	0 %	0 %	0 %	91 %	98 %

2.6.2. Consumo de combustible

A diferencia del buque de referencia, el perfil de consumo del buque a vela es mucho más simple, ya que la principal fuente de consumo, el motor propulsor, ha sido eliminada.

Por tanto, el único consumo de combustible a bordo es el de los generadores auxiliares, que deben suministrar la energía eléctrica requerida por el resto de los sistemas. Como se detalló, la planta eléctrica de este buque se compone de tres generadores principales de 210 kW y uno de emergencia de 100 kW.

Utilizando los mismos datos de consumo del fabricante (Tabla 27, puesto que los consumos son iguales para los motores seleccionados en esta opción que para los motores de 265 kW) y el perfil de carga para el buque a vela (Tabla 38), se calcula el consumo diario de combustible para cada condición operativa:

Basándose en el número de generadores en servicio y su régimen de funcionamiento en cada condición, se puede calcular el consumo en m³/día de la planta eléctrica auxiliar:

- Condición de Navegación: Para satisfacer la demanda eléctrica en navegación, se requiere el funcionamiento de dos de los generadores de 210 kW. En esta condición, ambos motores operan a un 74% de su carga nominal, lo que se traduce en un consumo diario de 2.53 m³/día.

- Condición de Maniobra: Durante la entrada y salida de puerto, la demanda aumenta. Para cubrir este pico, se utilizan dos generadores de 210 kW, operando a un régimen superior del 84%. El combustible requerido en esta condición asciende a 2.88 m³/día.

- Condición de Carga y Descarga: En este escenario, la demanda es cubierta por dos generadores de 210 kW trabajando a un 79% de su capacidad, con un consumo de 2.7 m³/día.

- Condición de Puerto: Durante la estancia en puerto, se emplea una configuración mixta de un generador de 210 kW y el de 100 kW, operando ambos a un 91% de su carga, lo que resulta en un consumo total de 2.2 m³/día.

La siguiente tabla muestra el resumen de los consumos en cada condición para el buque propulsado a vela.

Tabla 39: Consumo de combustible en cada condición del buque a vela

Condición operativa	Consumo total (m³/día)
Navegación	2,53
Maniobra	2,88
C y D	2,7
Puerto	2,2

Capítulo 6
Disposición general y Arquitectura naval

En ESTE capítulo se establece la disposición general del buque, tomando como punto de partida las formas generadas a partir de la librería ShipSim (Basilio Puente & M. Dolores Fernandez, 2021). Se detalla la compartimentación interna de la embarcación en función de las dimensiones principales de los espacios clave: bodegas de carga, piques de proa y popa, cámara de máquinas, zona de habilitación y tanques principales. Para ello, se han considerado como referencia diversas disposiciones generales de buques recogidas en publicaciones como *Significant Ships*, así como trabajos académicos de fin de grado, máster y proyectos finales de carrera. Todo el diseño se desarrolla conforme a la normativa de la sociedad de clasificación seleccionada, Bureau Veritas, específicamente su reglamento *Rules for the Classification of Steel Ships, Part B: Hull and Stability* (BUREAU VERITAS, 2025), y siguiendo también las *Common Structural Rules for Bulk Carriers and Oil Tankers* publicadas por la IACS (AICS, 2024).

En una primera etapa, se definen la eslora de líneas de carga, así como los elementos estructurales principales del buque: separación entre cuadernas, número y ubicación de los mamparos transversales, posición del mamparo de colisión, y configuración del doble fondo y costado.

Posteriormente, se procede al dimensionamiento de los diferentes espacios internos a partir de la distribución de los mamparos transversales, lo que permite delimitar áreas como la zona de habilitación, la zona de cámara de máquinas, además de establecer los accesos y sistemas de comunicación entre compartimentos.

Finalmente, con la compartimentación ya definida, los datos se incorporan al software Maxsurf, con el objetivo de obtener un cálculo detallado del volumen útil de las bodegas de carga. Este resultado será fundamental para llevar a cabo un análisis económico posterior con el mayor grado de precisión posible.

1. Eslora de líneas de carga

La eslora de líneas de carga (L_{LL}, de sus siglas en inglés,“Load line length”) de acuerdo con la sociedad de clasificación será la mayor de las siguientes dos medidas:

- El 96 % de la eslora total del buque, medida sobre una línea de flotación situada al 85 % del calado. Dado que nuestra embarcación dispone de bulbo, la medición de la eslora no se realiza hasta el punto más saliente del bulbo, sino que se debe tomar la proyección vertical hacia abajo del punto más retrasado de la curva del contorno, hasta intersectar la línea de flotación

$$L_{LL} = 0{,}96 * L_T(0{,}85D) = 0{,}96 * 97{,}28 = 93{,}38 \text{ m}$$

- La distancia desde el costado de la roda (fore side of the stem) hasta el eje del timón (axis of the rudder stock) sobre esa misma línea de flotación.

$$L_{LL} = 99 \text{ m}$$

En este caso la mayor es la medida desde la roda hasta el eje del timón, 99 m

2. Mamparos transversales

De acuerdo con los requisitos establecidos en la Sociedad de clasificación, todos los buques deben estar provistos, como mínimo, de los siguientes mamparos estancos transversales:

- Un mamparo de colisión (en proa);
- Un mamparo de pique de popa;
- Dos mamparos que delimiten el espacio de máquinas, en buques con máquinas situadas a popa, se requiere un mamparo por delante del espacio de máquinas.

Además de estos mamparos mínimos, para buques que no están sujetos a requisitos específicos de subdivisión, se deben instalar mamparos adicionales correctamente espaciados, de manera que el número total no sea inferior al que se indica en la Tabla 1 de la normativa.

Length (m)	Number of bulkheads for ships with aft machinery **(1)**	Number of bulkheads for other ships
L < 65	3	4
65 ≤ L < 85	4	5
85 ≤ L < 105	4	5
105 ≤ L < 120	5	6
120 ≤ L < 145	6	7
145 ≤ L < 165	7	8
165 ≤ L < 190	8	9
L ≥ 190	to be defined on a case by case basis	
(1) After peak bulkhead and aft machinery bulkhead are the same.		

Figura 41: Mínimo número de mamparos estancos (BUREAU VERITAS, 2025)

Para un buque con eslora de 100 metros, corresponde aplicar el intervalo:

85 ≤ L < 105 m

Según la tabla correspondiente, el número mínimo de mamparos es de 4 mamparos estancos transversales puesto que el buque dispone de maquinaria situada en popa.

En consecuencia, para el buque considerado, se requerirá un mínimo de 4 mamparos estancos transversales, además del cumplimiento de los requisitos específicos para los espacios de maquinaria y mamparos extremos. Para este caso específico se han considerado 7 mamparos estancos transversales, los mamparos de pique de popa, colisión, proa de cámara de máquinas y uno por cada vela rígida. Los de las velas rígidas junto con el mamparo de colisión definirán las bodegas de carga.

3. Mamparo de colisión

En el diseño estructural y de compartimentación de los buques, uno de los elementos fundamentales en términos de seguridad es el mamparo de colisión. Este mamparo, que debe ser estanco al agua hasta la cubierta de francobordo en buques de carga, se ubica en la zona de proa con el fin de limitar la inundación en caso de colisión frontal. Los criterios generales para la ubicación del mamparo de colisión se seguirán de acuerdo con la normativa “Pt B, Ch 2, Sec 1, 3 Collision bulckhead” (BUREAU VERITAS, 2025).

El mamparo de colisión debe instalarse a una distancia medida desde la perpendicular de proa de la eslora de línea de carga (FP_{LL}), dentro de los siguientes límites:

- Distancia mínima: no menor a 5 % de la eslora de líneas de carga (L_{LL})o 10 metros, lo que sea menor.
- Distancia máxima: no mayor a 8 % de la L_{LL} o 5 % de la L_{LL} más 3 metros, lo que sea mayor.

Estos límites tienen como objetivo garantizar tanto una protección efectiva ante impactos frontales como la optimización del espacio interior del buque.

Tabla 40: Distancia mínima y máxima del mamparo de colisión

Condición	Distancia del mamparo
Distancia mínima	**4,95 m** o 10 m
Distancia máxima	7,92 m o **7,95 m**

Cuando el buque dispone de un bulbo de proa, que se extiende hacia adelante por debajo de la línea de flotación existe una serie de particularidades, los valores anteriores no se miden directamente desde la perpendicular de proa, sino desde un nuevo punto de referencia adelantado, determinado de la siguiente manera:

La distancia debe medirse desde el punto que resulte más próximo a la FP_{LL} entre los siguientes tres:

- El punto medio de la longitud total del bulbo (medida bajo la línea de flotación);
- Un punto situado a 1,5 % de la eslora L_{LL} hacia proa desde la perpendicular de proa;
- Un punto situado a 3 metros hacia proa desde la perpendicular de proa.

Tabla 41: Distancia del mamparo de colisión

Condición	Distancia a la perpendicular de proa(m)
Punto medio de la longitud del bulbo	**0,845**
1,5% L_{LL}	1,48
3 m a proa de la perpendicular de proa	3

De estos tres valores, se toma el que represente la menor extensión hacia proa, ya que ello implica una referencia más conservadora desde la cual aplicar los límites mencionados anteriormente.

Por tanto, el mamparo de colisión debe ubicarse a una distancia entre 4,95 m y 7,95 m desde el punto situado 0,845 m por delante de la perpendicular de proa, en la cuaderna 119.

4. Doble fondo

El doble fondo se debe extender desde el mamparo de colisión hasta el mamparo del pique de popa, tan lejos como sea compatible con el diseño y la forma de trabajo del buque.

El doble fondo permite proteger el fondo del buque del fondo marino. La altura vertical del mismo se mide desde la quilla y debe mantenerse a lo largo de toda la manga y eslora en la que esté instalado, esta distancia vertical viene determinada por la sociedad de clasificación (BUREAU VERITAS, 2025) y se calculada a partir de la siguiente fórmula:

$$h = \frac{B}{20} = \frac{20}{20} = 1\text{m}$$

B: manga (m)

Esta distancia no debe ser menor de 760 mm ni mayor de 2 m. El resultado obtenido de 1 m se encuentra entre estos parámetros por lo que es el que se tendrá en consideración.

5. Doble casco

La distancia de doble casco viene establecido en la normativa “Common Structural Rules for Bulk Carriers and Oil Tankers” (AICS, 2024) donde se define que en los Bulk Carriers no

debe ser inferior a 1 m de distancia medido perpendicular al costado del buque. Este es construido en los costados del buque conectando mediante un mamparo longitudinal el doble fondo con la cubierta. Por otro lado, en el interior del doble casco debe de existir una mínima separación interna de 600 mm para permitir el paso de personal de forma que se pueda realizar una correcta revisión y mantenimiento.

6. Eslora de la cámara de máquinas

En este apartado se procede a definir la eslora de la cámara de máquinas con el objetivo de delimitar la distribución de espacios a bordo del buque. Esta definición es fundamental para establecer el volumen útil destinado a las bodegas de carga y calcular de forma fiable la capacidad total de carga de carbón. Disponer de una estimación sólida del volumen de las bodegas es esencial para los posteriores análisis económicos, ya que este dato condiciona la cantidad de mercancía transportada por viaje y la rentabilidad de la operación.

Con el objetivo de mantener un criterio conservador y homogéneo en la evaluación de las dos alternativas (buque convencional y buque a vela) se adoptarán las mismas dimensiones de cámara de máquinas para ambos casos. Esta decisión se justifica en términos prácticos: si bien el buque propulsado a vela dispondrá únicamente de motores generadores de pequeño tamaño, se estima que parte del espacio disponible en cubierta y en el casco se verá comprometido por la instalación de las velas rígidas y los mecanismos asociados a su orientación y despliegue. Por lo tanto, asumir una dimensión común para la cámara de máquinas en ambas configuraciones permite evitar una infraestimación del volumen útil destinado a carga, al tiempo que proporciona una base comparable para el análisis económico posterior.

La eslora de la cámara de máquinas se obtendrá, por lo tanto, en función de los elementos principales de la propulsión, los cuales son la línea de ejes, la reductora, el motor principal y el colector de tomas de mar.

Las dimensiones del motor principal se pueden apreciar en la siguiente figura:

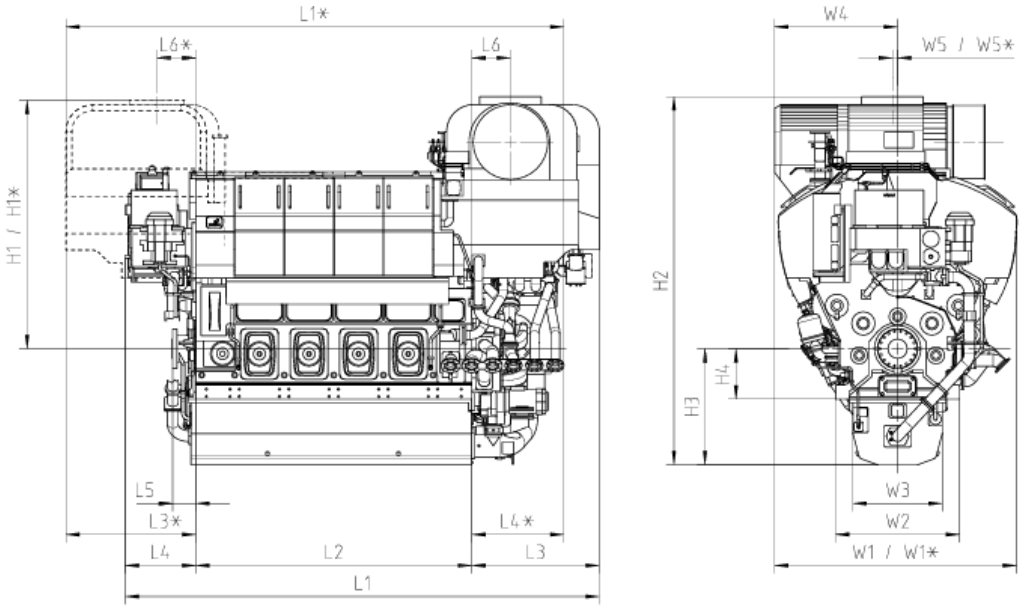

Fig 1-1 W8V31 & W10V31 Main engine dimensions (DAAF336230D)

Engine	L1	L1*	L2	L3	L3*	L4	L4*	L5	L6	L6*
W8V31	6121	6417	3560	1650	1650	911	1207	300	500	500
W10V31	6761	7057	4200	1650	1650	911	1207	300	500	500

Figura 42: Dimensiones del motor propulsor (Wärtsilä, 2025)

Haciendo uso de software CAD definiremos la eslora mínima de la cámara de máquinas. Incluyendo el modelo de motor correspondiente, una reductora de 1 me de largo y un espacio de 4 metros detrás del motor para colocar el colector de tomas de mar se obtiene una eslora de aproximadamente 12800 mm, lo equivalente a 16 claras de cuadernas de 800 mm. Coincidiendo el mamparo de proa de cámara de máquinas con la cuaderna 21.

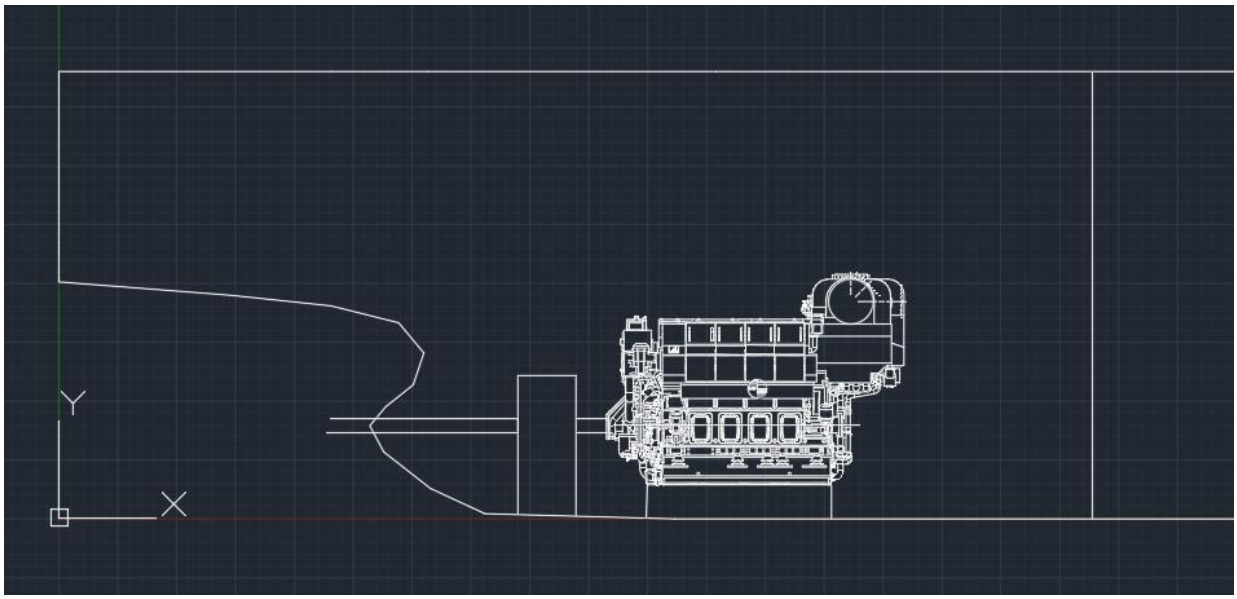

Figura 43: Esquema simplificado de la cámara de máquinas del buque convencional

7. Habilitación

La zona de acomodación se sitúa en la sección de popa del buque, ubicada sobre la cubierta principal y directamente por encima de la cámara de máquinas. Esta disposición permite una conexión funcional entre ambas zonas, facilitando el acceso del personal técnico mediante troncos de escaleras internos. La altura total del bloque de habilitación se diseña cumpliendo con los requisitos de visibilidad establecidos por el Convenio SOLAS (Capítulo V, Regla 22), garantizando así condiciones adecuadas para la navegación segura desde el puente de gobierno. Las distintas cubiertas del bloque de alojamiento han sido proyectadas en función de los requerimientos operativos, de descanso y de servicio del personal a bordo.

La dotación total del buque se ha establecido en 22 personas, distribuidas entre los distintos departamentos de a bordo según las funciones operativas y de mantenimiento requeridas. La tripulación se divide de la siguiente manera:

Tabla 42: Tripulación a bordo

Roles a bordo	Nº tripulantes
Departamento de cubierta	
Capitán	1
Primer oficial	1
Segundo oficial	2
Tercer oficial	1
Bosun/ Contramaestre	1
AB (able seaman)/ marinero	3
Cadetes	1
Departamento de máquinas	
Jefe de máquinas	1
Primer oficial	1
Segundo oficial	1
Tercer oficial	1
Oiler	2
Electricista	1
Fitter	1
Cadete	1
Departamento de fonda	
Cocinero	1
Ayudante de cocina	1
Camarero/mozo	1

7.1. Distribución del bloque de habilitación por cubiertas

El bloque de habilitación se distribuye verticalmente en varias cubiertas, con una altura libre entre cubiertas de 2750 mm, a excepción del puente de gobierno, que cuenta con una altura mayor de 3000 mm para mejorar la visibilidad y operatividad en la navegación.

El acceso a estas cubiertas se realiza a través de un tronco de escaleras interno, complementado con escaleras exteriores situadas en la superestructura. Estos accesos también se utilizarán como rutas de evacuación hacia las estaciones de abandono en caso de emergencia.

Todos los espacios de acomodación estarán equipados con iluminación artificial adecuada, ventilación forzada y un sistema centralizado de aire acondicionado, garantizando condiciones óptimas de habitabilidad durante toda la operación del buque.

En la cubierta principal se dispondrán la mayor parte de los camarotes, ocupará toda la manga del buque y 19,2 metros de eslora, toda la habilitación se encontrará tras el mamparo de proa de la cámara de máquinas adicionalmente en esta cubierta se encontrará el comedor y la cocina.

Las diferentes cubiertas de habilitación sobre la cubierta principal contarán con una manga de 11,5 metros y con una eslora de8,8 metros, se dispondrán de tres cubiertas diferentes que albergarán principalmente los camarotes de los oficiales y pañoles y una adicional para el puente de mando con una geometría diferente.

8. Bodegas de carga

Se define el volumen de las bodegas de carga a partir de los límites y superficies de contorno definidos anteriormente, los cuales son: doble fondo, doble casco, mamparo de colisión y mamparo de proa de cámara de máquinas. Estas se han generado en Maxsurf para obtener la geometría lo más ajustada posible.

En la Figura 44 se muestra una imagen de los tanques de carga en maxsurf Stability. En la que se muestran al 98 % de su capacidad.

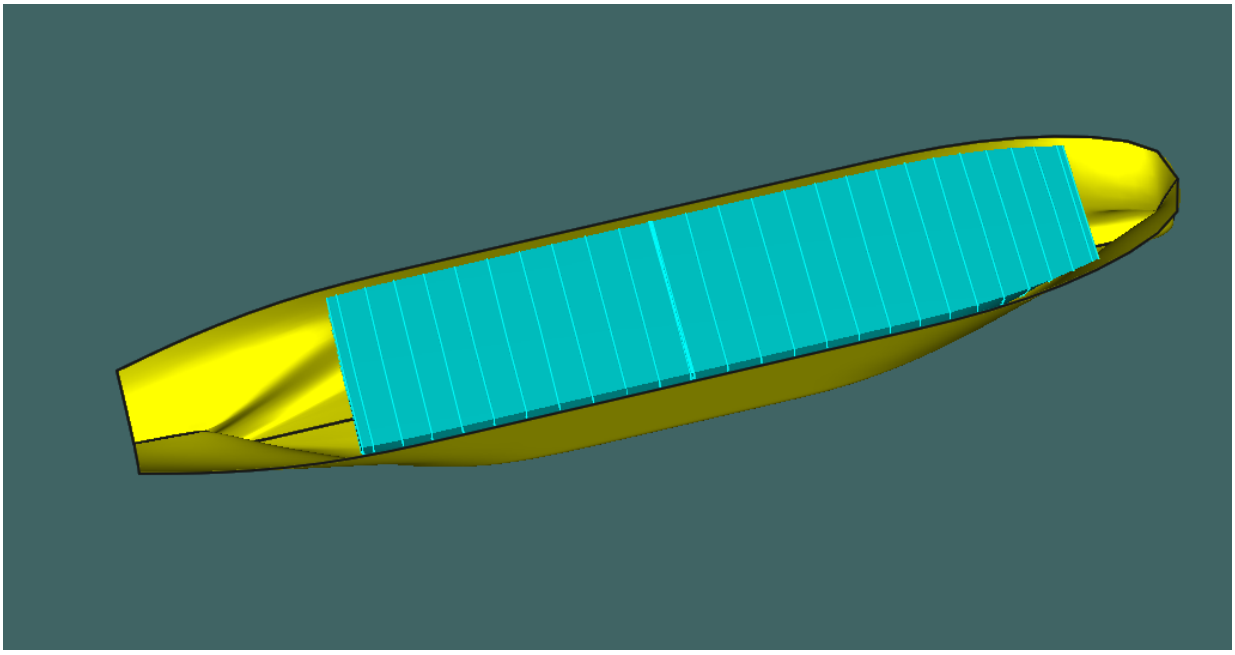

Figura 44: Bodegas de carga en Maxsurf Stability

La capacidad de las bodegas de carga es de $9387\,m^3$ para un 100% de capacidad. Este será el volumen utilizado posteriormente en el estudio económico. Suponiendo un factor de estiba de $1\,m^3\,/ton$ podemos suponer que la capacidad máxima de carga es de $10387\,ton$,

dado que el desplazamiento de nuestro buque es menor, se supondrá que las bodegas de carga no irán completamente cargadas.

9. Arqueo bruto y neto

El arqueo de un buque constituye una medida normalizada que permite determinar tanto su dimensión global como su implicación fiscal. Esta medición está regulada por el Convenio Internacional sobre Arqueo de Buques (IMO, 1969), el cual establece los procedimientos y criterios técnicos necesarios para su cálculo.

Desde un punto de vista técnico y normativo, el arqueo se descompone en dos valores fundamentales:

- Arqueo bruto (GT): Representa una medida adimensional que refleja el volumen total interior del buque, incluyendo todos los espacios cerrados, y se utiliza como referencia del tamaño general del buque.

- Arqueo neto (NT): Indica la capacidad útil del buque destinada al transporte de carga o pasajeros, excluyendo aquellos volúmenes que no contribuyen directamente a la función comercial del buque.

Para llevar a cabo el cálculo del arqueo de acuerdo con los principios establecidos en el convenio mencionado, es necesario emplear una serie de parámetros geométricos y operacionales que se detallan en los apartados siguientes.

Para asegurar el análisis más riguroso posible, se procederá a calcular de forma individual los volúmenes de los diferentes espacios para cada una de las dos alternativas de buque: la convencional y la propulsada a vela.

9.1. Arqueo Bruto

El arqueo bruto (GT) constituye una medida representativa del volumen total interno del buque, incluyendo todos los espacios cerrados, y se calcula mediante la siguiente expresión matemática:

$$GT = V * K_1$$

V: volumen total de todos los espacios cerrados del buque. (m^3) Este volumen se ha obtenido a partir del modelado tridimensional generado con el software Maxsurf, así como con programa CAD Rhinoceros.

Los volúmenes del buque convencional y del buque a vela se resumen en la siguiente tabla.

Tabla 43: Volumen de los espacios del buque propulsado a vela

Espacio	Buque convencional	Buque a vela
Bajo cubierta	15218	15218
Superestructura y chimenea	2378	1537
Volumen de carga	10387	10387
Total	27983	27142

Por otra parte, el coeficiente K_1 es un factor adimensional que depende del volumen V, y se determina aplicando la siguiente fórmula:

$$K_1 = 0.2 + 0.02 \cdot \log_{10} V$$

El arqueo bruto que se usará para cada buque y se empleará en el cálculo de las tasas es el siguiente:

Tabla 44: Arqueo bruto

GT Buque convencional	GT Buque a vela
8085	7835

9.2. Arqueo Neto

El Arqueo Neto se calcula mediante la siguiente expresión matemática:

$$NT = K_2 \cdot V_c \cdot \left(\frac{4 \cdot d}{3 \cdot D}\right)^2 + K_3 \cdot \left(N_1 + \frac{N_2}{10}\right)$$

V_c: volumen de los espacios de carga cerrados (m^3)
d: calado de trazado en el centro del buque ($4\ \mathrm{m}$)
D: puntal a la cubierta inferior ($9.5\ \mathrm{m}$)
K_3: 1.25
N_1: número de pasajeros en camarotes de hasta 8 plazas
N_2: número de pasajeros en espacios con más de 8 plazas (tipo literas)

El coeficiente K_2 es un factor adimensional que depende de V_c, y se determina aplicando la siguiente fórmula:

$$K_2 = 0.2 + 0.02 \cdot \log_{10} V_c$$

Por último, se deben considerar los siguientes criterios:

- El factor $\left(\frac{4 \cdot d}{3 \cdot D}\right)^2$ no será superior a 1.
- El término $K_2 \cdot V_c \cdot \left(\frac{4 \cdot d}{3 \cdot D}\right)^2$ no se tomará inferior a $0.25\ \mathrm{GT}$
- $NT \geq 0.3\ \mathrm{GT}$

Por lo que:

$$\left(\frac{4 \cdot d}{3 \cdot D}\right)^2 = \left(\frac{4 \cdot 4}{3 \cdot 9.5}\right)^2 = 0.3152 < 1$$

$$Buque\ convencional: K_2 \cdot V_c \cdot \left(\frac{4 \cdot d}{3 \cdot D}\right)^2 = K_2 \cdot Vc \cdot 1 = 2911\ NT > 2021 = 0{,}25\mathrm{GT}$$

$$Buque\ a\ vela: K_2 \cdot V_c \cdot \left(\frac{4 \cdot d}{3 \cdot D}\right)^2 = 2911\ NT > 1958 = 0{,}25\mathrm{GT}$$

Puesto que no hay camarotes de ocho plazas ni camarotes con literas el último termino que multiplica K_3 se desprecia. La tabla muestra los valores de arqueo neto que se considerarán para cada caso.

Tabla 45:Arqueo neto

NT Buque convencional	NT Buque a vela
2911	2911

10. Plano de disposición general

El plano de disposición general se puede consultar más adelante en el Anexo (pág., 180)

Capítulo 7
Análisis económico estadístico

Una vez definidos los parámetros técnicos y operativos de ambas alternativas de buque, en este capítulo final se procede a realizar el estudio económico comparativo. El objetivo es traducir el rendimiento técnico (tiempos de viaje, consumo, etc.) en variables económicas para poder evaluar de forma rigurosa la viabilidad del buque propulsado a vela frente a su homólogo convencional.

Para ello, se realizará un análisis de los costes asociados a cada proyecto, abarcando tanto los gastos de capital (CAPEX) como los costes de operación (OPEX) y de viaje (VOYEX). Se tendrán en cuenta todos los factores económicos relevantes, desde la inversión inicial en la construcción del buque hasta los costes recurrentes como el consumo de combustible o las tasas portuarias.

La principal herramienta metodológica para esta comparación será el Valor Actual Neto (VAN), o *Net Present Value (NPV)*. Este indicador financiero es ideal para evaluar proyectos con diferentes flujos de caja a lo largo del tiempo, permitiendo determinar bajo qué condiciones la inversión en un buque a vela empieza a ser rentable.

Finalmente, se realizarán diferentes simulaciones económicas, analizando la sensibilidad del resultado a variables clave como el precio del combustible. El objetivo es determinar el punto de equilibrio y las condiciones de mercado en las que la propulsión eólica no solo es una alternativa técnicamente viable, sino también económicamente interesante.

1. Tasas portuarias

Uno de los componentes más significativos de los costes variables de un viaje (VOYEX) son las tasas y tarifas asociadas a la estancia y operación del buque en los puertos. Para realizar un análisis económico riguroso, es importante cuantificar estos costes

En este apartado se desglosarán y formularán las tasas correspondientes a los dos puertos que definen la ruta de estudio: el puerto de Hampton Roads (Virginia, EE. UU.) como terminal de carga de carbón, y el puerto de Róterdam (Países Bajos) como principal punto de descarga en Europa.

1.1. Tasas portuarias de Hampton Roads

El puerto de Hampton Roads es el principal puerto de exportación de carbón de Estados Unidos y cuenta con varias terminales especializadas en su manejo. Para este estudio, se han considerado las dos instalaciones más importantes:

- La terminal Pier 6 de Norfolk Southern (NS) en Lamberts Point (Norfolk Southern, 2025): Descrita como la instalación de exportación de carbón más grande y rápida del hemisferio norte, es servida exclusivamente por el ferrocarril de Norfolk Southern. Una de sus características principales es que cuenta con almacenamiento en tierra, lo que permite la mezcla de diferentes tipos de carbón para cumplir con las especificaciones del cliente.

- La terminal de Dominion Terminal Associates (DTA): Es otra instalación de gran capacidad, propiedad de varias compañías de carbón y servida por el ferrocarril CSX.

Para el presente estudio, se ha seleccionado la terminal Norfolk Southern (NS) Pier 6 en Lamberts Point (Norfolk Southern, 2025) como el punto de carga de referencia. Esta elección se debe principalmente a la disponibilidad de información pública y detallada sobre sus tarifas. Esta accesibilidad a los datos permite realizar un cálculo de los costes de carga mucho más riguroso y preciso en comparación con otras terminales, siendo fundamental para la validez del análisis económico.

1.1.1. Tasas de manipulación del carbón

La principal tasa asociada a la manipulación de la mercancía en la terminal de Norfolk Southern es la de volcado y asentamiento.

Dumping Charges: Esta tarifa cubre el servicio de transferir el carbón desde las instalaciones de la terminal a las bodegas del buque y, posteriormente, distribuirlo y nivelarlo dentro de ellas para asegurar una estiba correcta y segura. Según la tarifa oficial, el coste de esta operación es de $8,5 por tonelada transportada. (Norfolk Southern, 2025)

$$Coste\ de\ la\ tasa = 8{,}5\frac{\$}{\text{ton}} \cdot tonelada\ de\ carga = \$\ 38250$$

1.1.2. Tasa de atraque y asistencia con remolcador

Esta tarifa cubre la asistencia de remolcadores y personal de amarre para las operaciones de atraque, desplazamiento en el muelle y desatraque del buque. El tarifario especifica un coste concreto para buques que realizan estas maniobras sin propulsión propia ("Vessels without power"), que es el caso que se aplicaría al buque a vela. Según el documento, la tarifa es de $0,83 por Tonelada Neta (N.T.) y para buques con propulsión propia será de $0,79 por Tonelada Neta (N.T.) (Norfolk Southern, 2025). Es necesario diferenciar entre el buque a vela y el buque con propulsión convencional.

Buque a vela:

$$Coste\ de\ la\ tasa = 0{,}83\frac{\$}{\text{N. T.}} \cdot \text{N. T.} = \$\ 2416$$

Buque convencional:

$$Coste\ de\ la\ tasa = 0{,}79\frac{\$}{\text{N. T.}} \cdot \text{N. T.} = \$\ 2299$$

1.1.3. Tasa de muelle (Dockage)

La tasa de "dockage" es el cargo que aplica la terminal por el uso del muelle mientras el buque está amarrado. El cálculo de esta tarifa es variable y depende del tiempo de estancia.

Para las primeras 24 horas, se aplica una tarifa de $0.65 por Tonelada de Registro Neto (NRT), para el tiempo que exceda las 24 horas, se aplica un cargo adicional de $0.02708 por NRT y por hora (o fracción).

Es importante destacar que esta tasa no se aplica durante los periodos en los que el buque ocupa el muelle mientras está esperando para ser cargado.

Para el propósito de este estudio, se asumirá que tanto el buque convencional como el propulsado a vela requieren el mismo tiempo para las operaciones de carga, 2 días. Dado que ambos buques tienen las mismas dimensiones principales y, por tanto, un arqueo neto (NT)

igual, se considera que el coste total por esta tasa será el mismo para ambas alternativas. (Norfolk Southern, 2025)

- Si el Tiempo en Muelle ≤ 24 horas:

$$Coste\,Total = 0.65\ (\$/\mathrm{NT}) \times Arqueo\ Neto\ del\ Buque\ (\mathrm{NT})$$

- Si el Tiempo en Muelle > 24 horas:

$$Coste\,Total = (0.65 \times \mathrm{NT}) + [0.02708 \times \mathrm{NT} \times (Horas\_en\_muelle - 24)]$$

1.1.4. Tasas por suministro de agua dulce

Finalmente, se considera el coste por el suministro de agua dulce al buque durante su estancia en puerto. Esta tarifa se compone de un cargo fijo por el servicio y un cargo variable en función de la cantidad de agua suministrada. Siendo el cargo fijo de $200 por la conexión al servicio y un cargo de $ 1 por cada tonelada de agua neta (Norfolk Southern, 2025)

1.1.5. Tasas de practicaje

El practicaje es un servicio obligatorio que garantiza que un experto local asista en la navegación y maniobra del buque. El coste de este servicio será idéntico para ambas alternativas de buque, ya que depende de sus dimensiones principales. Además, esta tarifa se tendrá en cuenta tanto para el atraque como para el desatraque.

Dado que el acceso a la tarifa oficial de practicaje de Hampton Roads no es factible, para realizar un cálculo lo más riguroso posible, se utilizará el tarifario público de la Canaveral Pilots Association, representativo de los puertos de la Costa Este de EE. UU.

Este tarifario establece una fórmula multicomponente que calcula el coste total del practicaje sumando varias tasas basadas en las características específicas del buque. La fórmula general es la siguiente

$$\begin{aligned} Coste\,Total = {} & (Tasa\_Eslora \times Eslora) + (Tasa\_Manga \times Manga) \\ & + (Tasa\_Calado \times Calado) + (Tasa\_Arqueo \times Arqueo) + Tasa\_Atraque \\ & + Tasa\_Bote\_Práctico \end{aligned}$$

Donde:
$Tasa\_Eslora$: $1.06 por pie de eslora total.
$Tasa\_Manga$**:** $5.26 por pie de manga.
$Tasa\_Calado$**:** $31.52 por pie de calado.
$Tasa\_Arqueo$**:** $13.13 por cada 1,000 Toneladas de Arqueo Bruto.
$Tasa\_Atraque$**:** $315.19 (tarifa fija).
$Tasa\_Bote\_Práctico$**:** $52.53 (tarifa fija).

1.2. Tasas portuarias de Róterdam

A diferencia de otros puertos que aplican una única tasa, el cálculo de los derechos portuarios en Róterdam es un proceso que busca incentivar la eficiencia y la sostenibilidad. El coste total no es una tarifa única, sino la suma de tres componentes principales que se calculan por separado:

- Componente del Buque: Basado en el tamaño del barco.
- Componente de la Carga: Basado en la cantidad de mercancía manipulada, con un posible descuento por eficiencia.

- Componente de Sostenibilidad: Basado en el perfil medioambiental del buque, que puede dar lugar a descuentos.

La información de las tasas portuarias de Róterdam se obtiene del artículo "*General Terms And Conditions. Including Port Tariffs. Port of Rotterdam*" (Quarch et al., 2025)

1.2.1. Componente del buque (vessel component)

La componente del buque se calcula en función del arqueo bruto (GT) del buque. El primer paso es identificar el tipo de buque y la tarifa aplicable (€/GT) según la tabla del tarifario del puerto mostrada a continuación.

Tabla 46: Tasas portuarias de Róterdam en función del arqueo bruto (Quarch et al., 2025)

Ltr.	Type of ship	GT tariff
A	Oil/product tankers	0.303
B	LNG tankers	0.309
C	Chemical/gas tankers	0.303
D	Bulk carriers	0.303
E	Containers ships in Deepsea Service	0.146
F	Container ships in Shortsea/Feeder Service	0.110
G	Container ship not in Scheduled Service	0.303
H	General Cargo ships in Deepsea Service	0.300
I	General Cargo ships in Shortsea/Feeder Service	0.114
J	General Cargo ships not in Scheduled Service	0.303
K	Car Carriers, Ropax and Roll-on/Roll-off ships in Scheduled Service	0.061
L	Car Carriers, Ropax and Roll-on/Roll-off ships not in Scheduled Service	0.130
M	Cruise ships	0.131
N	Offshore ships	0.303
O	Other Vessels/Seagoing Vessels	0.303

Como se puede apreciar en la Tabla 46 la tarifa del buque por tonelaje bruto para un buque Bulkcarrier es de 0,303 €/GT.

$$Tarifa\ portuaria\ del\ buque\ convencional = GT \cdot 0{,}303\,\frac{€}{\mathrm{GT}} = 2449\ €$$

$$Tarifa\ portuaria\ del\ buque\ \ a\ vela = GT \cdot 0{,}303\,\frac{€}{\mathrm{GT}} = 2374\ €$$

1.2.2. Componente de la carga (Cargo Component)

Esta parte se basa en la cantidad de toneladas de carbón que se descargan, pero incluye un sistema de descuento por eficiencia para buques que transportan mucha carga en relación con su tamaño (GT).

Cálculo base de la carga

Primero, se calcula el coste base de la carga multiplicando las toneladas descargadas por la tarifa correspondiente al carbón, la cual se puede apreciar en la Tabla 47

Tabla 47: Tasas portuarias de Róterdam del volumen de carga (Quarch et al., 2025)

No.	Commodity type	Cargo rate
01	Agribulk	0.576
02	Ores	0.576
03	Coal	0.576
04	Other dry bulk	0.576
05	Crude oil	0.737
06	Mineral oil products (including petcoke)	0.576
07	Other liquid bulk	0.576
08	Roll-on/Roll-off	0.390
09	Containers (including flats)	0.597
	-Shortsea/Feeder Service	0.417
	-Deepsea Service	0.543
10	Other general cargo	0.576
	-Shortsea/Feeder Service	0.417
	-Deepsea Service	0.442
11	LNG	0.597
12	Biomass	0.576
13	Scrap	0.560
14	Renewable fuels and chemicals	0.560

La tasa de la carga se calcula de la siguiente manera:

$$Tasa\ de\ la\ carga = Toneladas\ de\ carga \cdot 0{,}576 = 2880\ €$$

Calculo de la tasa máxima de la carga

A continuación, se calcula un "tope" o límite máximo para este coste. Este tope se calcula multiplicando el Arqueo Bruto (GT) del buque por un porcentaje de descuento (Tabla 48: Descuento por eficiencia (Quarch et al., 2025)) y por la misma tarifa de carga.

Tabla 48: Descuento por eficiencia (Quarch et al., 2025)

Ltr.	Type of ship	Deepsea (in %)	Shortsea (in %)
A	Container ships	35	45
B	General Cargo ships	45	55
C	Bulk carriers	133.3	
D	Oil-/product tanker	133.3	
E	Chemical/gas tankers	133.3	
F	General Cargo ships (no liner service)	133.3	

Por lo tanto, la tasa máxima de la carga se calculará como:

$$Tasa\ máxima\ de\ carga\ convencional = Arqueo\ bruto\ (\mathrm{GT}) \cdot 133{,}3\% \cdot 0{,}576 = 6207\ €$$
$$Tasa\ máxima\ de\ carga\ a\ vela = Arqueo\ bruto\ (\mathrm{GT}) \cdot 133{,}3\% \cdot 0{,}576 = 6015\ €$$

Tasa final de la carga

El coste final que se paga por la carga es el valor más bajo entre el cálculo base y el máximo. De esta forma, si el buque es muy eficiente (mueve muchas toneladas para su GT), su coste de carga estará limitado por el máximo, obteniendo así un descuento efectivo.

1.2.3. Componente de sostenibilidad y descuentos ambientales

Finalmente, se calcula un componente basado en la sostenibilidad, que también puede dar lugar a importantes descuentos.

Componente de sostenibilidad

Se calcula una tasa base multiplicando el Arqueo Bruto por una "tarifa de usuario" de sostenibilidad, la cual se puede apreciar en la Tabla 49.

Tabla 49: Componente de sostenibilidad (Quarch et al., 2025)

Ltr.	Type of ship	GT tariff
A	Oil/product tankers	0.065
B	LNG tankers	0.065
C	Chemical/gas tankers	0.065
D	Bulk carriers	0.065
E	Containers ships in Deepsea Service	0.065
F	Container ships in Shortsea/Feeder Service	0.065
G	Container ship not in Scheduled Service	0.065
H	General Cargo ships in Deepsea Service	0.065
I	General Cargo ships in Shortsea/Feeder Service	0.065
J	General Cargo ships not in Scheduled Service	0.065
K	Car Carriers, Ropax and Roll-on/Roll-off ships in Scheduled Service	0.043
L	Car Carriers, Ropax and Roll-on/Roll-off ships not in Scheduled Service	0.043
M	Cruise ships	0.087
N	Offshore ships	0.065
O	Other Vessels/Seagoing Vessels	0.065

El componente base de sostenibilidad para un Bulk Carrier será, por lo tanto:

$$Componente\ de\ sostenibilidad\ convencional = Arqueo\ Bruto\ (\mathrm{GT}) \cdot 0{,}065 = 525\ €$$

$$Componente\ de\ sostenibilidad\ a\ vela = Arqueo\ Bruto\ (\mathrm{GT}) \cdot 0{,}065 = 509\ €$$

Sobre esta tasa base se pueden aplicar descuentos por sostenibilidad, cuyo valor final diferirá entre el buque convencional y el propulsado a vela debido a sus distintos perfiles medioambientales.

Descuento de sostenibilidad ESI (Environmental Ship Index)

Se aplica un porcentaje de descuento en función de la puntuación ESI del buque, el cual se encuentra en la Tabla 50

Tabla 50: Descuento en función del ESI (Quarch et al., 2025)

ESI score from	ESI score to	NOX score	Discount percentage (in %)
1	21	n/a	5
21	31	n/a	10
31	41	n/a	60
41	61	n/a	80
61	81	n/a	100
81	and higher	to 60	100
81	and higher	from 60	120

Para el buque cuya propulsión es convencional se supondrá una puntuación ESI no muy baja ni muy alta, de aproximadamente 25, lo cual otorga un descuento de 10 %. En cambio para el buque propulsado a vela se supondrá la puntuación máxima de ESI que ofrece un descuento del 120 %, haciendo que la tarifa de sostenibilidad sea negativa para este caso.

Descuento de sostenibilidad "Green Award"

Adicionalmente, se contempla un descuento por el certificado Green Award. Para este estudio, se asume que el buque convencional no dispone de esta certificación, mientras que el buque propulsado a vela sí la obtendría.

Sin embargo, en la práctica, este descuento adicional resulta irrelevante para el buque a vela en este análisis. La razón es que, como se ha establecido, el buque a vela ya se beneficia de un descuento por sostenibilidad (ESI) del 120%. Este primer descuento por sí solo ya convierte el componente de sostenibilidad en un crédito negativo. Dado que la tasa base ya ha sido completamente bonificada y superada, las normativas del puerto no permiten aplicar descuentos adicionales sobre un valor ya negativo.

Por lo tanto, aunque el buque a vela posee la certificación, el único descuento efectivo que se aplica en el cálculo es el del ESI, que es el más favorable posible.

1.2.4. Tasa por gestión de residuos (Waste Fee)

Adicionalmente a los derechos portuarios, se aplica una tasa obligatoria por la gestión de los residuos del buque, en cumplimiento con la normativa MARPOL. El cálculo de esta tasa en el Puerto de Róterdam se basa en una estructura de tres componentes:

2. Un importe fijo por buque. 210 €
3. Un importe variable que depende del Arqueo Bruto (GT) del buque. 0,037 €/GT
4. Un importe máximo que actúa como límite. 1800 €

$$Tasa\ residuos\ convencional = 210 + GT \cdot 0{,}037 = 509\ €$$
$$Tasa\ residuos\ a\ vela = 210 + GT \cdot 0{,}037 = 499\ €$$

1.3. Coste total de las tasas portuarias por viaje

Una vez analizadas las diferentes tarifas en los puertos de carga y descarga, en este apartado se procede a consolidar todos los costes para obtener una cifra final del gasto en tasas portuarias para un viaje completo de ida y vuelta (Hampton Roads - Róterdam - Hampton Roads).

El cálculo se hará de forma separada para el buque convencional y para el buque propulsado a vela, destacando así las diferencias económicas entre ambas alternativas.

Para estimar el coste de un viaje de ida y vuelta, se han seguido las siguientes asunciones:

- Las tasas de la terminal de Hampton Roads se aplican una vez (durante la operación de carga).

- Las tasas del puerto de Róterdam se aplican una vez (durante la operación de descarga).

- El servicio de practicaje, que es necesario tanto para entrar como para salir de cada puerto, se calcula aplicando la tarifa proxy de la Costa Este de EE. UU. un total de cuatro veces (dos en Hampton Roads y dos en Róterdam, al no disponer de datos específicos para este último).

1.3.1. Tasas portuarias buque convencional

La siguiente tabla resume la suma de todos los costes portuarios para la alternativa convencional

Tabla 51: Coste total de tasas portuarias por viaje (Buque convencional)

Concepto	Base de cálculo	Coste estimado
Tasas de Hampton Roads		
Tasas de manipulación	$8{,}5\frac{\$}{ton}\cdot ton\ de\ carga$	$ 38250
Tasas de atraque	$0{,}79\frac{\$}{N.T.}\cdot N.T.$	$ 2299
Tasas de muelle	$(0.65 \times NT) + [0.02708 \times NT \times (H - 24)]$	$ 3784
Tasas de practicaje	$Tarifa\ sobre\ las\ dimensiones \cdot 4 = 1580 \cdot 4$	$ 6320
Tasas de Róterdam		
Componente del buque	$GT \cdot 0{,}303\frac{€}{\mathrm{GT}}$	2449 €
Componente de la carga		2880 €
Componente de sostenibilidad		472 €
Tasa por gestión de residuos	$210 + GT \cdot 0{,}037$	509 €
Total		**49850** €

1.3.2. Tasas portuarias buque a vela

A continuación, se presenta la misma tabla para el buque propulsado a vela. La mayoría de las tasas son idénticas al depender de las dimensiones del buque, pero el coste varía significativamente en los Derechos Portuarios de Róterdam debido a los descuentos por sostenibilidad.

Tabla 52: Coste total de tasas portuarias por viaje (Buque a vela)

Concepto	Base de cálculo	Coste estimado
Tasas de Hampton Roads		
Tasas de manipulación	$8{,}5\frac{\$}{ton}\cdot ton\ de\ carga$	$ 38250
Tasas de atraque	$0{,}79\frac{\$}{N.T.}\cdot N.T.$	$ 2416
Tasas de muelle	$(0.65 \times NT) + [0.02708 \times NT \times (H - 24)]$	$ 3784
Tasas de practicaje	$Tarifa\ sobre\ las\ dimensiones \cdot 4 = 1576 \cdot 4$	$ 6307
Tasas de Róterdam		
Componente del buque	$GT \cdot 0{,}303\frac{€}{\mathrm{GT}}$	2374 €
Componente de la carga		2880 €
Componente de sostenibilidad		−101 €
Tasa por gestión de residuos	$210 + GT \cdot 0{,}037$	499 €
Total		**49303** €

2. CAPEX y OPEX

Para poder realizar un análisis económico completo, es fundamental estimar los dos principales componentes de coste de un proyecto naval: el coste de inversión inicial o capital (CAPEX) y los costes de operación (OPEX).

El CAPEX representa la inversión que se debe realizar para la construcción y adquisición de cada buque. Por su parte, el OPEX agrupa todos los gastos recurrentes necesarios para

mantener el buque en funcionamiento a lo largo de su vida útil, como los costes de tripulación, mantenimiento, seguros o gestión.

En los siguientes apartados se detallará la metodología utilizada para calcular el CAPEX de ambas alternativas de buque, así como el desglose de sus costes de operación anuales, que servirán como datos de entrada para el modelo de simulación económica.

2.1. CAPEX

Para hacer una correcta estimación del CAPEX primero deberemos calcular el coste de la construcción de cada una de las opciones de buque. Para ello se utilizará la siguiente fórmula (Liviano & Rodríguez, n.d.).

$$CC = tps \cdot WST + CEc + cep \cdot BP + chf \cdot nch \cdot NT + cpe \cdot tps \cdot P_{er} + cva \cdot CC$$

Donde

- tps, es el coeficiente de coste unitario de la estructura montada (€ / t).
- WST es el peso de la estructura (t).
- CEc, es el coste de los equipos de manipulación y contención de la carga y de su montaje.
- cep, es el coeficiente del coste unitario de la maquinaria instalada (€/kW).
- BP, es la potencia del equipo propulsor (kW).
- chf, es el coste unitario de la habilitación y fonda (€/trip).
- NT, es el número de tripulantes.
- cpe, es el coeficiente de comparación del coste del equipo restante.
- P_{er}, es el peso del equipo restante, excluido el de carga (t).
- cva, es el coeficiente de costes varios aplicados.
- nch, es el nivel de calidad de la habilitación.

Siendo el valor de los coeficientes:

- $tps = 2800$ € / t.
- $cep = 360$ € / kW.
- $chf = 40000$ € /tripulante.
- $nch = 1{,}2$
- $cpe = 1{,}35$
- $cva = 0{,}1$
- $NT = 22$ tripulantes
- $CEc = 1500000$ €

Las incógnitas por calcular son el peso del acero (WST), la potencia propulsora (kW), y el peso del equipo restante (P_{er} , sin el equipo de manipulación y contención de la carga).

Para el peso del acero laminado (casco y superestructura) se ha hecho uso de la fórmula de J.L. Garcés, además se tomará un valor del 4,6% más debido al aumento de acero para graneleros de doble casco. Esta fórmula y consideraciones se han obtenido del Proyecto de finde carrera Nº 59 (Liviano & Rodríguez, n.d.)

$$WST = (0{,}024 \cdot L_{PP}^{1{,}5} \cdot B \cdot D \cdot 0{,}5) \cdot 1{,}046$$

Para el cálculo del peso del equipo restante (P_{er}) se ha utilizado la expresión de los apuntes de la asignatura de proyectos a la que hacen referencia en el PFC (Liviano & Rodríguez, n.d.) mencionado anteriormente.

$$P_{er} = 0{,}035 \cdot L_{PP}^{1{,}3} \cdot B^{0{,}8} \cdot D^{0{,}3}$$

Los precios de los diferentes elementos variarán desde el 2010 que es la referencia hasta el día de hoy. Por ello se calculará la inflación del 2010 al 2025 para poder hacer uso de los datos más rigurosos posibles. Se usará el Índice HIPC para ver la variación de precio.

Tabla 53: Índice HIPC 2010 y 2025 (Federal Reserve Bank of ST. Louis, 2025)

Fecha	Índice HIPC
2010	93,13
2025	129,04

Por lo que:

$$Precio_{ahora} = Precio_{2010} \cdot \left(\frac{HIPC_{ahora}}{HIPC_{ahora}}\right)$$

El coste de adquisición e instalación de una vela del sistema de velas es un componente fundamental del CAPEX para el buque propulsado a vela. La estimación de este coste se basa en los datos presentados en la siguiente tabla, que muestra una horquilla de precios en función del área de la vela.

Tabla 54: CAPEX y OPEX de las velas rígidas (Laursen, 2023)

	WASP	Rotor sail		Suction wing		Hard sail		Kite	
	Costs (EUR 1,000)	min	max	min	max	min	max	min	max
CAPEX	Asset costs	560	1,050	200	900	438	876	340	2,345
	Installation costs (newbuild)	84	158	30	135	66	130	51	351
	Installation costs (retrofit)	140	263	50	225	109	219	85	586
One-off costs	Training	10	10	10	10	10	10	10	10
OPEX	Annual maintenance & repair	12	22	4	18	8	18	17	117
	Annual energy consumption WAPS	26	79	26	53	No data available			

Dado que la tabla proporciona un rango de costes (un límite mínimo y uno máximo), para realizar un cálculo concreto en este estudio se ha optado por utilizar el valor medio de dicha horquilla.

Según las anteriores fórmulas y teniendo en cuenta la inflación puesto que los datos son del 2010, las diferentes opciones del buque, tanto convencional como a vela, tendrán el siguiente coste:

Tabla 55:Resumen de costes buque convencional y a vela

Variable	Buque convencional	Buque a vela
Potencia instalada	6095 kW	730 kW
Peso del acero laminado	1757 ton	1757 ton
Peso de equipo restante	274 ton	274 ton
tps_{ahora}	3879 €	
cep_{ahora}	498 €	
chf_{ahora}	55423 €	
CEc_{ahora}	2078400 €	
Velas Coste	0 €	4 · 657000 €
Velas instalación	0 €	4 · 98000 €
Estimación del coste	16314000 €	16395000 €

Una vez estimado el coste de adquisición de cada buque, es necesario anualizar esta inversión para poder incluirla en el análisis de flujos de caja del proyecto. Para asegurar una comparación equitativa, se han aplicado las mismas consideraciones de financiación y amortización a ambas alternativas de buque.

Es importante destacar que las condiciones y tasas utilizadas a continuación se basan en modelos de ejercicios académicos y se establecen como una hipótesis de cálculo para este estudio, en lugar de reflejar condiciones de mercado específicas.

Financiación del Buque Un 25% del coste de adquisición se cubre con fondos propios, mientras que el 75% restante se financia mediante un préstamo a 10 años con un tipo de interés fijo del 6%. El método de devolución del préstamo es el de cuotas de principal constante (sistema alemán), donde la porción de capital amortizado es la misma cada año.

Para la amortización contable del buque, se han establecido las siguientes condiciones:

- El activo se amortiza en un periodo de 15 años.
- Se considera un valor residual del 15% del precio de compra al final de su vida útil.
- El método de amortización es el lineal, aplicando una tasa anual del 7% sobre el valor del activo.

A continuación, en los siguientes apartados, se aplicarán estas condiciones a los costes de adquisición de cada buque para desglosar los flujos de caja anuales asociados al CAPEX

2.1.1. CAPEX buque convencional

El cálculo del CAPEX anualizado para el buque convencional se detalla en las siguientes tablas, que muestran los principales cálculos para los tres primeros años del proyecto.

Tabla 56: Datos principales de la financiación (buque convencional)

FINANCIACIÓN	
Valor buque	16.313.469 €
Fondos propios (25%)	4.078.367 €
Importe préstamo (75%)	12.235.102 €
Duración préstamo (años)	10
Forma devolución	Cuota ppal. Constante
Pago anual ppal	1.223.510 €
Interés (anual)	6%

Tabla 57: Amortización (Buque convencional)

AMORTIZACIÓN (sistema lineal)	
Precio buque	16313469 €
Valor residual del 15%	2447020 €
Valor amortizable	13866448 €
Amortización anual (15 años)	**924429** €

Tabla 58: Financiación anual (Buque convencional)

FINANCIACIÓN			
AÑO	PPAL. PENDIENTE INICIAL	DEVOLUCIÓN PPAL.	INTERESES (6% sobre Pend. Inicial)
2020	12.235.102 €	1.223.510 €	734.106 €
2021	11.011.592 €	1.223.510 €	660.696 €
2022	9.788.082 €	1.223.510 €	587.285 €
2023	8.564.571 €	1.223.510 €	513.874 €

Tabla 59: CAPEX anual (Buque convencional)

INTERESES (€)	AMORTICACIÓN (€)	CAPEX (€)
660.696	924.430	1.585.125
587.285	924.430	1.511.715
513.874	924.430	1.438.304

2.1.2. CAPEX buque a vela

Aplicando la misma metodología de financiación y amortización que en el apartado anterior, a continuación se desglosa el cálculo del CAPEX anual para el buque propulsado a vela.

Tabla 60: Datos principales de la financiación (Buque a vela)

FINANCIACIÓN	
Valor buque	16.394.522 €
Fondos propios (25%)	4.098.631 €
Importe préstamo (75%)	12.295.892 €
Duración préstamo (años)	10
Forma devolución	Cuota ppal. Constante
Pago anual ppal	1.229.589 €
Interés (anual)	6%

Tabla 61: Amortización (Buque a vela)

AMORTIZACIÓN (sistema lineal)	
Precio buque	16394522,32
Valor residual del 15%	2459178,347
Valor amortizable	13935343,97
Amortización anual (15 años)	**929022,93**

Tabla 62: Financiación anual (Buque a vela)

FINANCIACIÓN			
AÑO	PPAL. PENDIENTE INICIAL	DEVOLUCIÓN PPAL.	INTERESES (6% sobre Pend. Inicial)
2020	12.295.892 €	1.223.510 €	737.754 €
2021	11.072.382 €	1.223.510 €	664.343 €
2022	9.848.871 €	1.223.510 €	590.932 €
2023	8.625.361 €	1.223.510 €	517.522 €

Tabla 63: CAPEX anual (Buque a vela)

INTERESES (€)	AMORTICACIÓN (€)	CAPEX (€)
664.343	924.430	1.588.773
590.932	924.430	1.515.362
517.522	924.430	1.441.952

2.2. OPEX

A continuación, se definirán los costes de operación (OPEX) para cada una de las alternativas de buque. La estimación de estos costes se basa en los datos presentados en la Tabla 64 que muestra el OPEX para buques graneleros en función de su tamaño y ofrece un rango con un valor superior e inferior.

Dado que el buque de estudio tiene un tamaño inferior al de las categorías presentadas en la tabla de referencia, se ha optado por un enfoque conservador, seleccionando el valor inferior de dicha horquilla como el OPEX base para este análisis.

Este OPEX base se considera común para ambas alternativas de buque, ya que cubre gastos como la tripulación, los seguros y la gestión general, que se asumen idénticos. La principal diferencia entre ambos es que el buque propulsado a vela incurre en un coste operativo adicional que el buque convencional no tiene: el mantenimiento y la reparación del sistema vélico, aun así, el buque convencional cuenta con los costes operativos de mantenimiento y reparación del motor principal.(Tabla 54).

Por lo tanto, el OPEX anual final para cada buque se define de la siguiente manera:

- OPEX Buque Convencional: Se corresponde directamente con el OPEX base.
- OPEX Buque a Vela: Es la suma del OPEX base más el coste anual estimado para el mantenimiento de las velas.

Tabla 64: OPEX anuales Bulkcarriers (Moore Greece, 2022)

BULK CARRIERS

Handysize (10,000 - 39,999 dwt)

Daily KPIs	TCE	Crew Wages	Provisions	Crew Other	Lubricants	Stores	R & M	Spares	Insurance costs	Admin	Total Opex*
Observations	**136**	**135**	**129**	**133**	**133**	**134**	**132**	**126**	**135**	**135**	**135**
Average	$17,417	$2,176	$194	$481	$277	$408	$276	$358	$386	$991	$5,509
Lower bound	$10,202	$1,982	$145	$202	$155	$209	$75	$151	$264	$677	$4,438
Higher bound	$22,569	$2,454	$223	$669	$388	$560	$456	$541	$497	$1,362	$6,408

Tabla 65: OPEX Convencional frente a OPEX vela

OPEX Convencional	OPEX vela
$\$\,\mathbf{4438} = \mathbf{3816}\,€$	16816 €

3. Fuel EU

Además del análisis de costes como gastos de capital (CAPEX), operativos (OPEX) y las tasas portuarias, se incorporará, una estimación del impacto económico derivado de la normativa medioambiental, específicamente la regulación FuelEU Maritime, cuya entrada en vigor está prevista para 2025.

Este marco regulatorio introduce un coste adicional que afectará la competitividad del buque de propulsión convencional frente a la alternativa de propulsión a vela. Dado que la ruta transcurre entre un puerto de la Unión Europea y Estados Unidos, el cálculo de las emisiones y las posibles penalizaciones se aplicará sobre el 50% de la energía consumida, conforme a lo estipulado en la regulación. La proyección de este sobrecoste considerará la evolución previsible de los combustibles a emplear durante los próximos 35 años, periodo que define el horizonte del proyecto.

Para cuantificar el impacto económico de la regulación FuelEU Maritime en el buque convencional, se ha modelado su coste como una penalización variable a lo largo de la vida útil del proyecto. Este sobrecoste se ha calculado en euros por cada tonelada de combustible consumida (€/tonelada), diferenciando entre HFO (Heavy Fuel Oil) y LFO (Light Fuel Oil).

Los valores reflejan la penalización creciente a medida que los objetivos de reducción de emisiones de la Unión Europea se vuelven más estrictos con el tiempo.

A continuación, se presentan los costes anuales por tonelada de combustible quemada que se incorporarán al modelo de simulación económica para el buque convencional durante los 35 años de horizonte del proyecto, obtenidos a partir de la herramienta FuelEU Calculator de BetterSea (BetterSea, 2025).

Tabla 66: Penalización del FuelEU a combustibles convencionales (BetterSea, 2025)

Año	HFO €/ton	LFO €/ton
2025	62,21	53,98
2026-2029	62,21	53,98
2030-2034	156,43	149,74
2035-2039	355,66	353,22
2040-2044	745,34	748,21
2045-2040	1475,59	1490,32
+2050	1899,6	1921,22

4. Sistema de evaluación económica: Simulación de Montecarlo

Una vez definidos todos los parámetros técnicos y operativos de ambas alternativas de buque (tiempos de viaje, consumos, CAPEX y OPEX), en este apartado final se procede a realizar la evaluación económica comparativa. El objetivo es determinar las condiciones bajo las cuales la inversión en un buque propulsado a vela es económicamente viable frente a su homólogo convencional.

La viabilidad de este proyecto a largo plazo no depende de un único escenario, sino de su capacidad para ser rentable en un entorno de incertidumbre. Variables como la variabilidad del

precio del combustible, la incertidumbre sobre el coste de capital (la tasa de descuento) y, sobre todo, la variabilidad de los tiempos de viaje del buque a vela, introducen un nivel de riesgo que un cálculo simple cálculo del VAN no puede determinar.

Por este motivo, para la evaluación se ha optado por un enfoque probabilístico mediante la Simulación de Montecarlo. Esta metodología permite modelar la incertidumbre de las variables clave para analizar no solo un único resultado, sino una distribución de miles de futuros posibles. El objetivo no es obtener una única respuesta de "sí" o "no", sino calcular la probabilidad de que el buque a vela sea una inversión más rentable que la alternativa convencional.

En los siguientes apartados se detallará la configuración del modelo de simulación, la definición de los parámetros fijos y las variables estocásticas, y finalmente, se presentarán y analizarán los resultados de la simulación.

4.1. Flete requerido

El objetivo de cada simulación es calcular el flete requerido (FR) para cada flota. Esta métrica representa el flete mínimo (€/tonelada) que cada flota necesitaría cobrar para cubrir todos sus costes (CAPEX, OPEX, combustible, tasas, etc.) a lo largo de su vida útil y alcanzar un Valor Actual Neto (VAN) de cero. La flota que obtenga un flete requerido más baja en un escenario determinado se considerará la alternativa económicamente más eficiente.

4.2. Parámetros del modelo

El modelo se alimenta de dos tipos de variables: parámetros fijos, calculados en capítulos anteriores, y variables libres, que representan las principales incertidumbres.

- Parámetros fijos: son los valores que se mantienen constantes en todas las simulaciones. Incluyen el CAPEX y OPEX de cada buque, las tasas portuarias, la capacidad de carga, etc.

- Variables libres: son las variables inciertas, modeladas con distribuciones de probabilidad:

    - Precio del combustible: siendo una de las mayores incertidumbres del mercado, se ha modelado mediante una distribución triangular. Esta distribución se elige cuando se tiene una idea clara del valor más probable, pero se quiere tener en cuenta escenarios más extremos. Se ha definido con un rango de 450 €/tonelada (mínimo), 650 €/tonelada (valor más probable) y 1000 €/tonelada (máximo).

    - Tasa de descuento: para reflejar la incertidumbre sobre el coste de capital a lo largo de la vida del proyecto, se ha utilizado una distribución uniforme entre un 7% y un 11%. Se opta por una distribución uniforme cuando se puede definir un rango plausible, pero no hay razones para suponer que un valor dentro de ese rango sea más probable que otro. El rango elegido representa desde un entorno de financiación favorable y de bajo riesgo (7%) hasta un escenario con tipos de interés más altos y una mayor prima de riesgo exigida por los inversores (11%).

    - Tiempos de viaje (buque a vela): esta es la variable más crítica del modelo. En lugar de asumir una distribución teórica (como la normal), se ha utilizado la distribución empírica real obtenida de los más de 8,100 resultados exitosos de las simulaciones del algoritmo de *weather routing*. En cada iteración de Montecarlo, el modelo toma un tiempo de viaje de ida y otro de vuelta al azar directamente de este conjunto de datos. Este enfoque es el más robusto posible,

ya que asegura que la variabilidad utilizada en el análisis económico no es una suposición, sino una consecuencia del rendimiento del buque calculado previamente.

- Coste de emisiones (EU ETS): para el buque convencional, se ha incluido un coste por emisiones de CO_2. Se ha fijado un valor constante en términos reales de 70 € por tonelada de CO_2 emitida, basándose en la cotización media del mercado de futuros de derechos de emisión.(ICE, 2025)

- Caso base de análisis: los resultados detallados se presentarán para un caso base, definido por una flota de referencia de 2 buques convencionales y una vida útil del proyecto de 35 años.

4.3. Caso de estudio

Para comprender el funcionamiento interno del modelo de Montecarlo, es útil desglosar el proceso que se sigue en una única iteración de las miles que componen la simulación completa. El objetivo de cada iteración es comparar ambas flotas en un único futuro posible, definido por un conjunto de variables aleatorias.

El primer paso es definir el escenario. Para esta iteración en particular, el modelo toma un valor al azar para cada una de las variables inciertas, basándose en las distribuciones de probabilidad que hemos definido:

- Se elige un precio del combustible (ej: 700 $/tonelada).
- Se elige una tasa de descuento (ej: 9%).
- Se elige un tiempo de viaje de ida para el buque a vela de la base de datos de resultados calculados (ej: 35 días).
- Se elige un tiempo de viaje de vuelta para el buque a vela (ej: 42 días).

Con este conjunto de valores, el futuro para esta simulación ha quedado fijado. Ahora, el problema es determinar qué flota es más económica en estas condiciones específicas.

Mientras que la flota convencional es fija (2 buques), la flota a vela debe adaptarse a las condiciones de tiempo de viaje de este escenario:

Primero, se calcula la capacidad de transporte anual de un único buque a vela con los tiempos de viaje de este escenario. Un ciclo completo tardaría:

$$2\,(puerto) + 35\,(ida) + 42\,(vuelta) + 2\,(puerto) = 81\ días$$

lo que le permitiría transportar una cantidad X de toneladas al año.

A continuación, se compara esa capacidad X con el objetivo de carga anual fijado por la flota convencional. Si para mover la carga objetivo se necesitaran, por ejemplo, 4,3 buques a vela, el modelo determina que es necesario construir 5 buques enteros, ya que no es posible construir una fracción de un buque. Este es el paso clave donde la variabilidad del viaje se traduce en un coste de inversión (CAPEX).

Una vez determinado el tamaño de ambas flotas para este escenario, el modelo calcula todos los costes a lo largo de los 35 años de vida útil del proyecto:

- Costes anuales: para cada flota, se calculan los costes operativos anuales, que son la suma del OPEX, el coste del combustible (usando el precio de 700 €/tonelada de este escenario) y, para el buque convencional, el coste de las emisiones ETS.

- Costes financieros: se calculan los flujos de caja de la devolución del préstamo para el CAPEX total de cada flota.

- Carga anual: se calcula la cantidad total de carga que cada flota moverá anualmente.

Finalmente, se aplican las fórmulas del Valor Actual Neto (VAN) a estos flujos de 35 años, utilizando la tasa de descuento del 9% de este escenario, para obtener el coste total del proyecto y la carga total transportada en dinero de hoy. En la simulación no solo se actualizan los costes al presente, sino también la carga transportada. Esto se hace para que ambos se midan en la misma escala temporal. Transportar una tonelada en el primer año no tiene el mismo valor que hacerlo al final de la vida útil del proyecto, ya que lo que ocurre antes siempre tiene más peso que lo que sucede dentro de veinte o treinta años. Si solo se descontasen los costes y la carga se dejase sin ajustar, se estaría comparando magnitudes que no son equivalentes y el resultado quedaría distorsionado. Por eso, el modelo calcula el Valor Actual Neto de la carga, de forma que tanto los costes como la carga reflejan su valor presente. Así, al dividir uno entre otro, el ratio de rentabilidad que se obtiene es coherente y permite comparar de forma justa los dos tipos de flota.

Dividiendo los costes entre la carga, se obtiene el Flete Requerido (FR) para cada flota.

$$FR = \frac{VAN_{carga}}{VAN_{costes}}$$

El resultado de esta única simulación podría ser:

- FR Flota Convencional: 100 €/tonelada
- FR Flota a Vela: 99 €/tonelada

En esta iteración concreta, la flota a vela ha sido la opción más económica. Este proceso completo se repite miles de veces, y la acumulación de estos miles de resultados individuales es lo que permite construir los histogramas y calcular la probabilidad final de rentabilidad que se presentan en los siguientes apartados.

5. Resultados del caso base

Para comenzar el análisis, se presenta en detalle el resultado de la simulación para el escenario base, definido por una flota de referencia de 2 buques convencionales y una vida útil del proyecto de 35 años. Este análisis en profundidad nos permitirá entender la dinámica fundamental de la comparación entre ambas flotas.

La primera figura muestra la distribución del Flete Requerido (FR) para ambas flotas a lo largo de las 16,000 simulaciones. Esta es la métrica central del análisis.

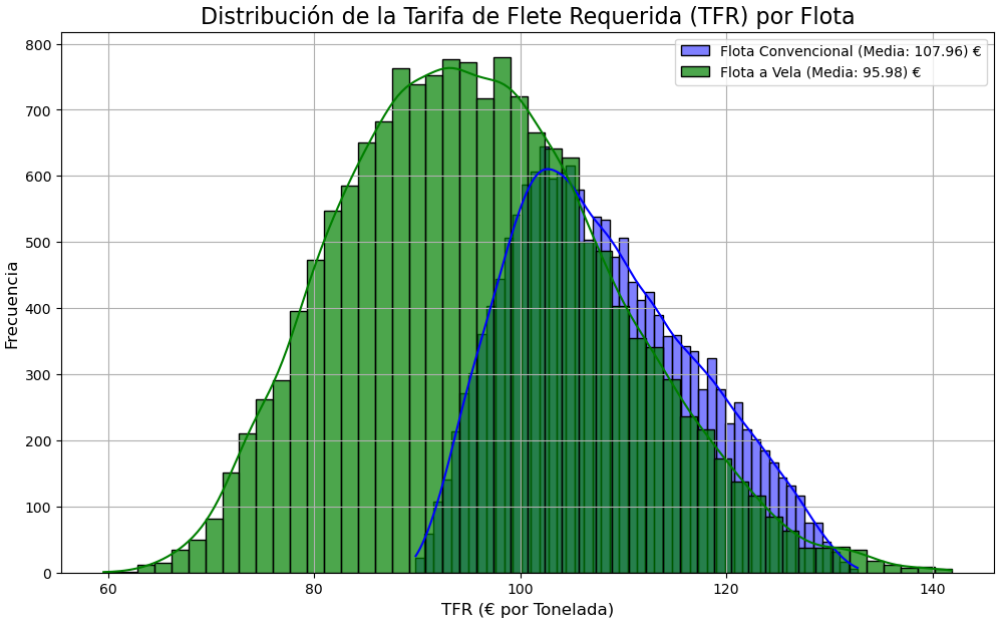

Figura 45: Histograma de la tarifa de flete requerida del caso base

La siguiente figura ilustra uno de los resultados más importantes de la simulación, la distribución del número de buques a vela que fueron necesarios en cada escenario para igualar la capacidad de transporte de la flota convencional. Este gráfico visualiza directamente el riesgo de la inversión asociado a la variabilidad de los vientos.

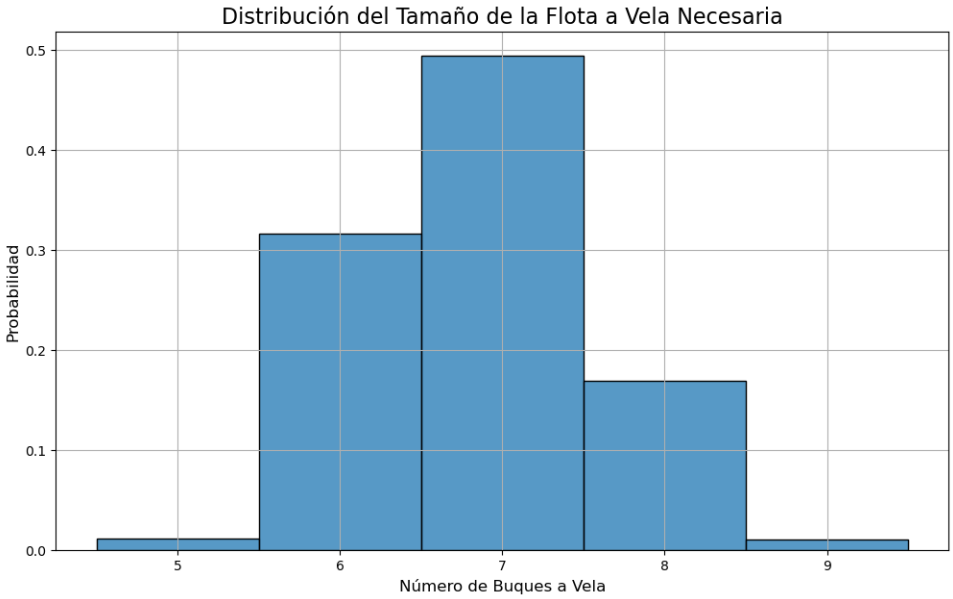

Figura 46: Distribución del tamaño de la flota a vela necesaria

Los resultados numéricos de la simulación para el caso base se resumen a continuación:

Tabla 67: Resultados simulación de montecarlo caso base

Valor	Flota Convencional	Flota a Vela
FR Medio	107 € / tonelada	95 € / tonelada
Nº Medio de Buques	2 (Fijo)	6,85
Probabilidad de ser más rentable	19,49%	80,51%

Como se puede observar en los resultados, en el escenario base, la flota propulsada a vela es más competitiva para el caso de estudio, necesitando un flete requerido de 95 € por tonelada, frente a los 107 € de la flota convencional. Esto se traduce en una probabilidad del 80,51% de que la inversión en la flota a vela sea más rentable que la alternativa convencional a lo largo de los 35 años de vida útil. El principal factor que ayuda a la rentabilidad de la flota a vela son los costes añadidos por las normativas europeas sobre los combustibles fósiles del buque convencional, sin tener en cuenta estas normativas el resultado estaría más ajustado.

Sin embargo, el análisis del tamaño de la flota a vela revela el principal riesgo del proyecto. Para cumplir con el objetivo de carga, se necesitaron de media 6,85 buques a vela. Como muestra el histograma, aunque en escenarios de vientos muy favorables se necesitaron menos buques, también hubo una probabilidad significativa de necesitar 8 o incluso más, lo que incrementaría notablemente el CAPEX y el riesgo de la inversión. La conclusión del caso base es que, aunque la alternativa a vela presenta una ligera ventaja económica, esta es muy ajustada y está sujeta a una considerable incertidumbre operativa.

6. Análisis de incertidumbre y sensibilidad

El histograma de resultados del caso base muestra una amplia distribución de posibles resultados, lo que refleja la incertidumbre inherente al proyecto. El propósito de un análisis de sensibilidad es entender cuáles de las variables libres (precio del combustible, tiempo de viaje, tasa de descuento) son las que más contribuyen a esta dispersión, es decir, cuáles son los factores de riesgo y de oportunidad más importantes del proyecto.

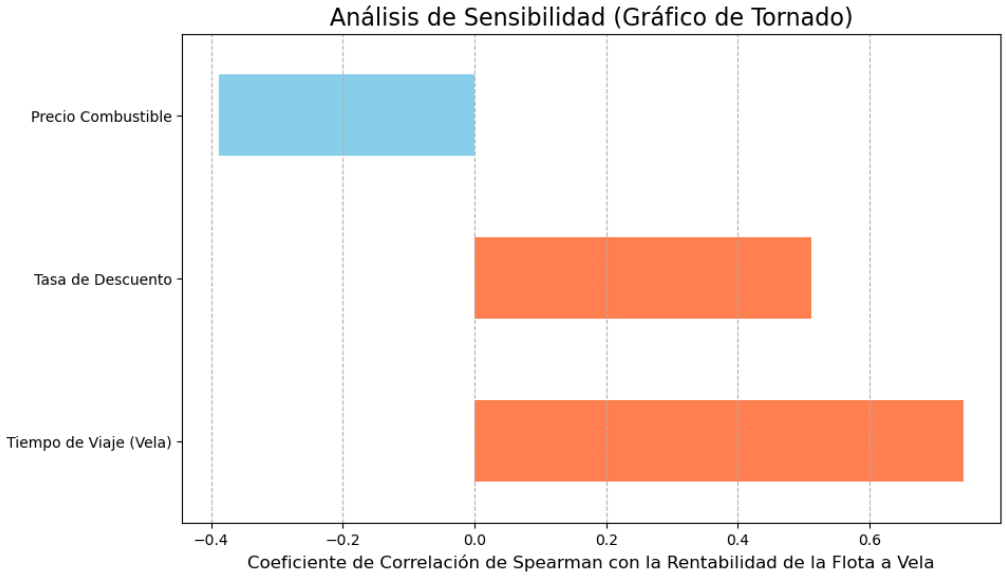

Figura 47: Gráfico de sensibilidad de FR

El análisis arroja tres conclusiones principales, ordenadas por su nivel de impacto:

Tiempo de viaje (Vela), correlación: +0.74 (positiva y muy fuerte): esta es, con diferencia, la variable más influyente. Una fuerte correlación positiva significa que a medida que el tiempo de viaje del buque a vela aumenta, su competitividad empeora drásticamente. Esto es lógico: viajes más largos implican que se pueden realizar menos ciclos al año, lo que obliga a construir una flota más grande (y por tanto, más cara en CAPEX y OPEX) para transportar la misma cantidad de carga anual. Este factor representa el principal riesgo operativo y económico del proyecto.

Tasa de descuento, correlación: +0.51 (positiva y moderada): el segundo factor en importancia es el coste del capital. Una correlación positiva indica que a medida que la tasa de

descuento sube, la flota a vela se vuelve menos competitiva. La razón es que una tasa de descuento alta penaliza en mayor medida a los proyectos que requieren una inversión inicial (CAPEX) más elevada. Dado que la flota a vela a menudo necesita un mayor número de buques, su CAPEX total suele ser superior, haciéndola más vulnerable a un entorno financiero con financiación más cara.

Precio del combustible, correlación: -0.39 (negativa y moderada): finalmente, el precio del combustible es el tercer factor clave y representa la principal oportunidad de mercado para el buque a vela. Una correlación negativa significa que a medida que el precio del combustible sube, la competitividad de la flota a vela mejora significativamente. Esto es el núcleo del valor del buque a vela, que es prácticamente inmune a la volatilidad del precio del combustible, mientras que esta penaliza severamente la rentabilidad de la flota convencional. Cuanto más caro sea el combustible, más rentable será la alternativa a vela.

Dado que el precio del combustible ha sido identificado como el aliado principal del buque a vela, se ha realizado un análisis más profundo de su impacto. La siguiente figura muestra un diagrama de dispersión que mapea la rentabilidad de la flota a vela (puntos verdes si es más rentable, rojos si no lo es) para cada una de las simulaciones, en función del precio del combustible y la tasa de descuento.

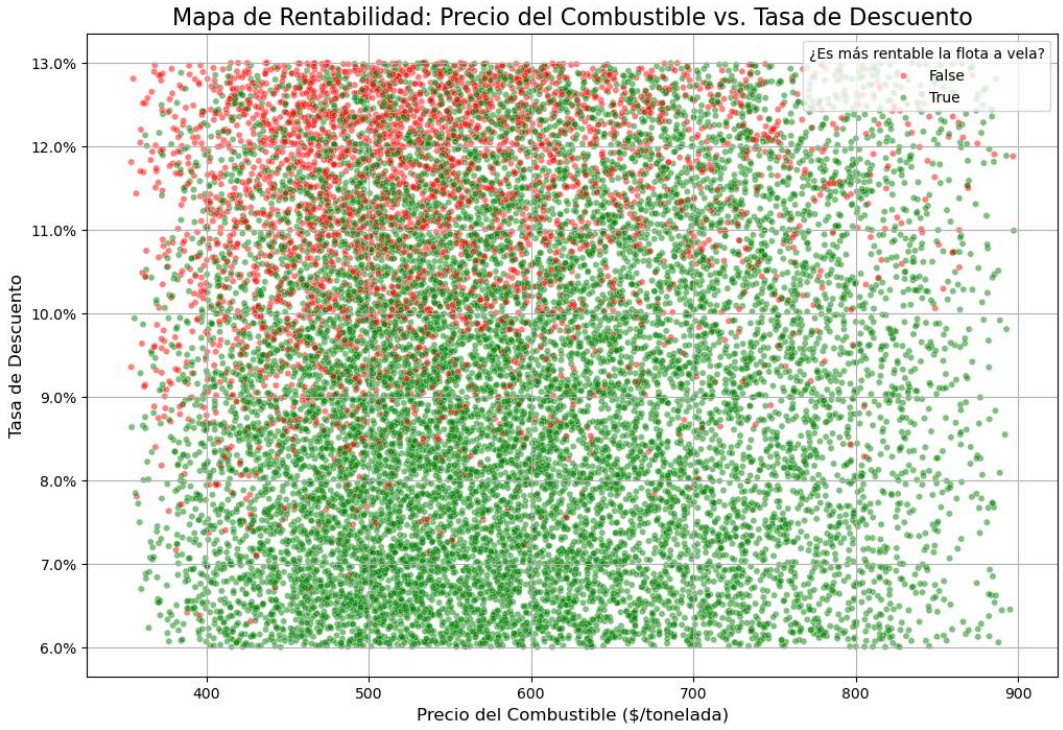

Figura 48: Mapa de rentabilidad en función del precio del combustible y la tasa de descuento

El gráfico muestra cada simulación como un punto, coloreado en verde si la flota a vela resultó más rentable (FR más bajo) y en rojo si no lo fue. El análisis de esta distribución permite extraer varias conclusiones clave:

- Influencia dominante del precio del combustible: se observa un claro gradiente de color de izquierda a derecha en la parte superior. En escenarios con precios de combustible bajos (por debajo de 600 €/tonelada), la mayoría de los resultados son desfavorables para la flota a vela (puntos rojos). A medida que el precio del combustible aumenta, la densidad de puntos verdes se incrementa drásticamente, hasta convertirse en la opción dominante en escenarios de combustible caro. Esto confirma visualmente la conclusión del gráfico de tornado: el precio del combustible es el principal motor económico del proyecto.

- Influencia secundaria de la tasa de descuento: aunque menos pronunciado, también se aprecia un gradiente vertical. Con un mismo precio de combustible, la probabilidad de que un punto sea verde es mayor en la parte inferior del gráfico (tasas de descuento bajas) que en la superior. Esto confirma que, como se esperaba, un coste de capital más bajo favorece la inversión en la flota a vela debido a su mayor CAPEX.

- Impacto de la variabilidad del viaje: La frontera entre la zona roja y la verde no es una línea nítida, sino una zona difusa con puntos de ambos colores mezclados. Esta mezcla es la evidencia del impacto de la tercera variable libre, el tiempo de viaje.

    - Los puntos verdes en la zona de combustible barato representan aquellos escenarios afortunados en los que, a pesar del bajo precio del combustible, los vientos fueron tan favorables que los tiempos de viaje del buque a vela fueron excepcionalmente cortos, necesitando una flota pequeña y, por tanto, rentable.

    - Por el contrario, los puntos rojos en la zona de combustible caro representan los escenarios desafortunados en los que, a pesar de la ventaja del combustible, los vientos fueron tan malos que se necesitaron demasiados buques para cumplir el objetivo de carga, haciendo que la inversión no fuera rentable.

En conclusión, este mapa de rentabilidad no solo confirma los resultados del análisis de sensibilidad, sino que también ilustra la interacción entre el riesgo de mercado (precio del combustible) y el riesgo operativo (variabilidad del viaje) que define la viabilidad de este proyecto.

7. Análisis de escala y temporal

Una vez analizado el caso base y la sensibilidad a las variables de mercado, este último apartado explora cómo cambia la viabilidad del proyecto al modificar dos decisiones estratégicas clave: la escala de la operación (el tamaño de la flota de referencia) y el horizonte temporal de la inversión (la vida útil del proyecto). Para ello, se ha ejecutado la simulación de Montecarlo completa para múltiples escenarios, variando el tamaño de la flota convencional entre 2, 4 y 6 buques, y la vida útil del proyecto entre 20 y 35 años.

La siguiente figura resume los resultados de este análisis completo, mostrando la evolución de la probabilidad de que la flota a vela sea la opción más económica.

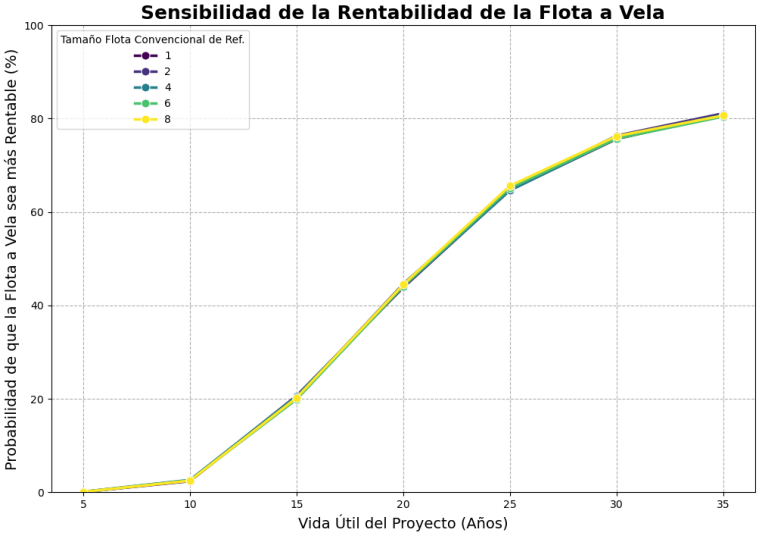

Figura 49: Evolución de la rentabilidad de la flota a vela en función de la vida útil y el tamaño de la flota convencional

El análisis de esta gráfica revela dos conclusiones fundamentales y muy distintas para cada una de las variables estudiadas, que son clave para entender el modelo de negocio de la propulsión a vela.

El primer resultado notable, como se observa en la superposición de las líneas de diferentes colores en el gráfico, es que la probabilidad de rentabilidad de la flota a vela es independiente del tamaño de la flota de referencia.

La razón de esta invarianza reside en la forma de evaluación utilizada, el FR no mide el beneficio total de una flota en millones de euros, sino la eficiencia económica, expresada en €/tonelada. Es el precio mínimo que se necesita cobrar por cada tonelada para que la inversión se pague a sí misma.

Al escalar la operación (por ejemplo, al pasar de una flota de 2 a 4 buques), tanto los costes totales de la flota (CAPEX, OPEX, etc.) como la cantidad total de carga que esta puede transportar se duplican. Como el FR se calcula dividiendo los costes totales entre la carga total, el factor de escala se cancela en el numerador y el denominador, resultando en el mismo valor de FR. Por tanto, la conclusión no es que una flota pequeña sea más o menos rentable que una grande, sino que la tecnología de propulsión a vela es, bajo las condiciones estudiadas, más eficiente que la convencional en un 50% de los escenarios a partir de los 23 años, independientemente de la flota de referencia de la operación.

En contraste con lo anterior, el análisis revela una dependencia muy fuerte y positiva entre la rentabilidad de la flota a vela y el tiempo de operación del proyecto. Como muestra la pendiente ascendente de las curvas en la gráfica, la viabilidad de la inversión en propulsión a vela mejora a medida que se alarga su vida útil.

Analizando la evolución de la probabilidad:

- Con una vida útil de 10 años, la probabilidad de que la flota a vela sea rentable es inferior al 10%.
- A los 15 años, esta probabilidad aumenta hasta aproximadamente el 20%.
- A los 20 años, alcanza el 42%.
- El punto de equilibrio, donde ambas flotas tienen la misma probabilidad de ser rentables, se sitúa en torno a los 23 años.
- Finalmente, en el caso base de 35 años, la probabilidad se consolida en torno al 80%, convirtiéndose en la opción económicamente más probable.

Esta tendencia, que se asemeja a una curva de crecimiento logarítmico, se explica por la estructura de costes fundamental del proyecto. El buque a vela se caracteriza por un alto coste de inversión inicial (CAPEX), cuyos flujos de caja negativos (la devolución del préstamo) se concentran en los primeros 10 años. Su gran ventaja, el masivo ahorro en combustible y las tasas de emisiones, son un beneficio operativo que se va acumulando de forma constante año tras año.

En un horizonte de inversión corto (como 10 o 15 años), simplemente no hay tiempo suficiente para que los ahorros acumulados lleguen a compensar la gran inversión inicial. Sin embargo, a medida que el proyecto se alarga más allá del periodo de financiación, cada año adicional es un año de grandes ahorros operativos. Son estos años extra los que decantan la

balanza, permitiendo que los beneficios a largo plazo del bajo consumo superen el coste inicial de la tecnología. Esta dinámica demuestra que la inversión en propulsión a vela no debe ser evaluada como una solución a corto plazo, sino como una estrategia a largo plazo, cuya viabilidad económica se consolida y fortalece a medida que la vida útil del proyecto se expande. El código encargado de mostrar esta gráfica se muestra en el Anexo (pág. 172)

8. Coste de variabilidad asumido por el fletador

Hasta ahora, el análisis económico se ha centrado en la perspectiva del armador, determinando el flete requerido para asegurar la rentabilidad de la inversión en el buque. Sin embargo, para que una solución de transporte marítimo sea verdaderamente viable, también debe serlo para el cliente que contrata el servicio, en este caso, el fletador. El principal desafío que introduce la propulsión a vela es la variabilidad en los tiempos de viaje, un factor que no afecta de la misma manera a los buques de propulsión convencional, como ya se ha comentado en numeradas ocasiones anteriormente.

Esta irregularidad en las entregas representa un riesgo para el fletador. Una industria que depende de un suministro constante de materia prima, como el carbón en este caso, no puede permitirse una interrupción en su cadena de suministro. Un retraso inesperado en la llegada de un buque podría agotar sus reservas y forzar una parada en la producción, con los consecuentes costes económicos. Para mitigar este riesgo, el fletador se ve obligado a mantener un volumen de inventario adicional, conocido como stock de seguridad.

El propósito de este apartado es calcular el coste derivado de mantener dicho stock de seguridad. Para ello, en primer lugar, se determina la dimensión del stock de seguridad en toneladas. Se necesita conocer el consumo diario de la fábrica y el máximo retraso inesperado que se pretende cubrir. El consumo diario se obtiene relacionando la carga que se transporta anualmente por la flota convencional con los 365 días del año que estará operativa la fábrica.

$$Consumo\ Diario = \frac{Carga\ Anual\ Objetivo}{365}$$

El máximo retraso inesperado (Δt) se puede definir de acuerdo a dos enfoques de visión del riesgo, un enfoque razonable es cubrir la desviación desde el tiempo de ciclo promedio hasta el máximo, mientras que un enfoque más conservador y con mayor miedo al riesgo cubrirá todo el rango de variabilidad posible, entre el tiempo máximo y el tiempo mínimo de viaje que se puede obtener de las simulaciones de tiempo de viaje.

$$\Delta t_{prudente} = T_{ciclo\ máximo} - T_{ciclo\ promedio}$$
$$\Delta t_{conservador} = T_{ciclo\ máximo} - T_{ciclo\ mínimo}$$

Con estos valores se podría calcular el stock de seguridad como el consumo a lo largo de ese retraso inesperado. Una vez dimensionado el stock se pude calcular su valor económico, el cual está principalmente asociado con el capital inmovilizado. Para calcularlo, primero se determina el valor del inventario de carbón adicional, fijando un precio de aproximadamente 100 €, valor de referencia del mercado durante el estudio realizado. (Trading Economics, 2025)

$$Valor\ Stock = Consumo\ diario \cdot\ \Delta t \cdot Precio\ del\ carbón$$

Este valor es un capital que la empresa no puede destinar a otras inversiones, el coste de esta condición es el coste de oportunidad, el cual se podría definir como el rendimiento que se podría a ver obtenido con ese dinero en otras inversiones. Para este análisis, dicho coste se representa mediante una tasa de descuento k del 8%.

Finalmente, para que el coste sea comparable y se pueda tener en cuenta en la toma de decisiones se divide por el total de la mercancía transportada al año, pudiendo obtener así el sobrecoste por tonelada.

$$Sobrecoste = \frac{Valor\ Stock \cdot k}{Carga\ Anual\ Objetivo}$$

Para el caso de estudio de una flota de dos buques convencionales, suponiendo dicha carga anual objetivo (139000 toneladas) y con un enfoque prudente el stock de seguridad necesario para garantizar la operación se calcula de 12,6 toneladas. Esto se traduce en un sobrecoste como se ha comentado debido al capital inmovilizado de 0,8 €/ton.

Optando por la postura más conservadora frente al riesgo el stock de seguridad se establece en 21, 5 toneladas y por lo tanto un sobrecoste de 1,36 €/ton. Estos resultados para el caso base se obtienen a partir del código dispuesto en el Anexo (pág. 178)

Este valor es el precio que el fletador tendría que asumir para convertir una opción de transporte variable, aunque sostenible, en la solución logísticamente tan segura como la convencional. Este sobrecoste debe ser considerado junto al flete del armador para realizar una comparación equitativa entre ambas alternativas.

Al comparar este sobrecoste por tonelada con los valores medios del Flete Requerido (FR) obtenidos en la simulación principal, se observa que el impacto es limitado. El coste adicional para el fletador representa una variación inferior al 2% del flete total.

Esto sugiere que, si bien el coste de la irregularidad es una que el fletador debe tener en cuenta en su planificación, no altera de forma significativa la competitividad económica de la flota a vela. Los factores determinantes en la rentabilidad del proyecto, como el ahorro de combustible y el impacto de las normativas de emisiones (ETS y FuelEU), tienen un peso considerablemente mayor en la estructura de costes final.

Por este motivo este análisis se presenta de forma separada. El cálculo principal del FR se mantiene estrictamente desde la perspectiva del armador (sus costes e inversión), mientras que este apartado calcula un riesgo específico del fletador.

El planteamiento de este análisis considera que el stock de seguridad funciona como un colchón permanente, en lugar de una cantidad que se gasta y desaparece. Se ha pensado el inventario total como un sistema de dos niveles: un stock operativo, previsto para el consumo habitual entre entregas, y el stock de seguridad, una reserva que permanecería intacta a la espera de un imprevisto. La idea es que esta reserva solo se utilice en situaciones excepcionales, como un retraso extremo, con el objetivo de evitar que la producción se detenga.

Bajo esta lógica, si ocurriera un retraso y se consumiera parte de este colchón de seguridad, la llegada del siguiente buque sería un factor clave, pues se entiende que su cargamento tendría como prioridad reponer el colchón, devolviéndolo a su nivel original. Este mecanismo de reposición permitiría que, tras superar una situación adversa, el sistema se recuperase y estuviera mejor preparado para afrontar otro desafío similar. De esta forma, se asume que el modelo puede ser representativo a nivel anual, al contemplar una recuperación entre posibles eventos. Es importante matizar que este enfoque no está pensado para una secuencia de varios de los peores viajes posibles de forma consecutiva, pues un escenario de esa naturaleza impediría el ciclo de reposición que se ha considerado clave para la estabilidad del sistema.

Capítulo 8
Conclusiones

Este trabajo es un análisis prospectivo cuyo objetivo principal era evaluar la viabilidad técnica y económica de un buque granelero propulsado íntegramente por velas, comparándolo con una alternativa convencional. Para ello, ha sido necesario desarrollar una metodología que abarca desde la simulación física del movimiento del buque en función del viento hasta un análisis económico probabilístico. A partir de los resultados obtenidos en los capítulos anteriores, se pueden extraer las siguientes conclusiones.

Uno de los principales resultados de este trabajo ha sido el desarrollo, desde cero, de una herramienta de software funcional: un algoritmo de *weather routing* capaz de simular travesías para un buque propulsado a vela. Se ha demostrado que es posible integrar fuentes de datos, como los datos de viento reales de MERRA-2 y los diagramas polares de rendimiento del buque, para calcular tiempos de viaje.

Cabe señalar que, para superar las dificultades y los costes computacionales que implica una simulación de esta magnitud, el algoritmo implementa una estrategia de poda agresiva (Búsqueda en Haz) que descarta una gran cantidad de nodos en cada paso. Por este motivo, no se puede afirmar que la herramienta sea un optimizador perfecto o el más realista posible. Es probable que con el uso de equipos informáticos de mayor capacidad o con el desarrollo de heurísticas de poda aún más sofisticadas, se pudiesen obtener resultados de ruta todavía más ajustados a la realidad. Además para poder realizar todas las simulaciones se ha tenido que hacer uso del sistema de super computación CesViMa Margerit-3 sin el cual este trabajo no hubiese sido posible terminar.

Sin embargo, para el objetivo principal de esta tesis, que no era crear el algoritmo de generación de rutas definitivo, sino generar un conjunto de datos estadísticamente significativo de los tiempos de viaje para alimentar un modelo económico, la herramienta desarrollada ha demostrado ser adecuada.

El análisis de los miles de viajes simulados con esta herramienta ha revelado una dependencia estacional en los tiempos de viaje. Los meses de invierno y finales de otoño resultaron ser significativamente más rápidos que los de verano, una variabilidad que se confirma como el principal factor de riesgo operativo de la propulsión eólica. Se ha demostrado que esta incertidumbre en la duración de las travesías tiene un impacto económico directo, ya que en los escenarios con vientos desfavorables, se requiere la construcción de una flota de mayor tamaño para poder garantizar el mismo transporte de carga anual que una flota convencional.

El análisis económico, realizado mediante una Simulación de Montecarlo, no ofrece una respuesta absoluta, sino que permite cuantificar la probabilidad de éxito de la inversión.

Para el caso base estudiado, la flota a vela presenta una probabilidad de ser más rentable que la convencional de aproximadamente el 80% de los casos a partir de los 35 años de vida útil del proyecto, teniendo en cuenta todas las penalizaciones al combustible fósil. El análisis de sensibilidad ha permitido identificar los factores clave que condicionan este resultado. Se ha encontrado que la viabilidad del proyecto es extremadamente sensible al precio del combustible y a la imposición de tasas sobre las emisiones de CO_2, siendo estos los principales motores económicos que favorecen a la alternativa a vela. Por otro lado, el tiempo de viaje se confirma como el mayor riesgo, seguido del coste del capital (la tasa de descuento), que penaliza la mayor inversión inicial que requiere la flota a vela.

Una de las conclusiones más interesantes es la relación de la rentabilidad con el tiempo. El estudio demuestra que la propulsión a vela es una inversión a largo plazo. Su probabilidad de éxito es baja en proyectos con una vida útil corta, pero se incrementa de forma muy notable a medida que el horizonte temporal se alarga, superando el umbral del 50% de probabilidad en proyectos a más de 23 años. Es en estos plazos largos donde los ahorros acumulados en combustible y emisiones logran compensar de forma consistente el mayor CAPEX.

Es importante comentar que si el buque convencional tomase medidas para la reducción del uso de combustibles fósiles la rentabilidad a 35 años seguramente estaría más ajustada. En cálculos previos a la introducción de las penalizaciones por FuelEU la probabilidad de rentabilidad de la inversión se reducía al 50 % en el caso base, un 30% por debajo del obtenido con las penalizaciones.

Por otro lado, la tecnología de la propulsión a vela permite economías de escala, ya que el aumento del tamaño de las mismas no aumenta significativamente ni los CAPEX ni OPEX de las mismas, a diferencia de los motores convencionales donde el aumento del tamaño incrementa significativamente el precio del motor.

Este trabajo aporta una metodología para la evaluación de sistemas de propulsión no convencionales, que podría ser adaptada para otros tipos de buques y rutas. La propulsión a vela, podría ser una alternativa económicamente viable en un futuro con combustibles y derechos de emisión caros.

Como limitaciones del estudio, cabe señalar que se ha centrado en una única ruta y tipo de buque, y que los parámetros económicos son estimaciones. Como posibles líneas de investigación futura, sería de gran interés aplicar esta misma metodología para analizar otras rutas comerciales, comparar diferentes tecnologías de velas, o introducir una variabilidad año a año en los tiempos de viaje dentro de la simulación de Montecarlo para un análisis del riesgo aún más detallado, así como reducir las penalizaciones en el buque convencional por el uso de otros combustibles.

Este trabajo ha seguido, como indica su título, los principios de la industria Lean. El enfoque se ha centrado en optimizar el transporte marítimo mediante el estudio de stock de seguridad y su implicación del flete, sin dejar de cumplir el objetivo de transportar la misma carga anual. De forma simultánea, se ha buscado mejorar la rentabilidad y disminuir las emisiones de gases de efecto invernadero, alineándose así con la filosofía Lean de eliminar desperdicios y maximizar la eficiencia del sistema.

Bibliografía

AICS. (2024). *Common Structural Rules for Bulk Carriers and Oil Tankers*. https://erules.veristar.com/dy/data/bv/pdf/606-NR_2024-01.pdf

Basilio Puente, & M. Dolores Fernandez. (2021). *ShipSIM Package*. https://github.com/BasilioPV/ShipSIM/tree/main/ShipSIM

BetterSea. (2025). *FuelEU Calculator*. https://www.bettersea.tech/fueleu-calculator

BUREAU VERITAS. (2025). *RULES FOR THE CLASSIFICATION OF STEEL SHIPS, PART B, HULL AND STABILITY*. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://erules.veristar.com/dy/data/bv/pdf/467-NR_PartB_2025-07.pdf

Clarksons. (2020). *What is dry bulk cargo ?* https://www.clarksons.com/glossary/what-is-dry-bulk-cargo/

d'Amore-Domenech, R., Leo, T. J., & Pollet, B. G. (2021). Bulk power transmission at sea: Life cycle cost comparison of electricity and hydrogen as energy vectors. *Applied Energy*, *288*, 116625.

EVOLUTION, & Yatch Racing System. (2025). *Polar Curves, what are they?* https://evolution-tactic.com/en/polar-curves/

Federal Reserve Bank of ST. Louis. (2025). *Harmonized Index of Consumer Prices: All-Items HICP for Euro Area*. https://fred.stlouisfed.org/data/CP0000EZ19M086NEST

Franco, R. (2020). *Proyecciones y sistemas de coordenadas*. https://rodolfofrancoweb.com/sig/proyecciones_y_sistemas_de_coordenadas/coordenadas-geograficas/

Global Modeling and Assimilation Office (GMAO). (2015). *MERRA-2 tavg1_2d_flx_Nx: 2d,1-Hourly,Time-Averaged,Single-Level,Assimilation,Surface Flux Diagnostics V5.12.4, Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC)*. 2015. https://doi.org/10.5067/7MCPBJ41Y0K6

Grootendorst, G., Rainone, K., Cullipher, K., McNab, R., & Komarek, T. (2022). *Comprehensive Economic Development Strategy For Hampton Roads*. https://hamptonroadsalliance.com/ceds/

ICE. (2025). *ICE Endex EUA Futures*. https://www.ice.com/products/197/EUA-Futures/data?marketId=7505169&span=3

IEA. (2024). *Coal 2024*. IEA. https://www.iea.org/reports/coal-2024

IEA. (2025). *World Energy Statistics and Balances*. IEA. https://www.iea.org/data-and-statistics/data-product/world-energy-statistics-and-balances

IMO. (1969). *Convenio Internacional Sobre Arqueo de Buques*.

Laursen, R. . P. H. . S. D. . Z. R. . N. D. . V. S. D. . P. E. . (2023). *Potential of Wind-Assisted Propulsion for Shipping*. *October*, 1–271. www.emsa.europa.eu

Liviano, J., & Rodríguez, J. (n.d.). *PFC Nº 59, Granelero 50.000 tpm*.

López de Asiaín, A. (2021). *Apuntes Asignatura Proyecto de Sistemas Auxiliares*.

Moore Greece. (2022). *Moore Maritime Index 2022*. https://www.moore-greece.gr/MediaLibsAndFiles/media/greeceweb.moorestephens.com2020/Documents/Insights/mmi/Moore-Maritime-Index-Brochure-2022.pdf

Norfolk Southern. (2025). *Norfolk Southern Coal Tariff NS #1000*. https://www.norfolksouthern.com/content/dam/nscorp/pdf/NS Coal Coke and Iron Ore Tarrif NS _1000 - Eff. Jul 1, 2025.pdf

Parker Host, T. (2022). *Coal export report, East coast terminals*. https://www.tparkerhost.com/wp-content/uploads/2023/08/USEC-and-USG-Combined-July-2023.pdf

Quarch, B., Geissler, D., & Peters, P. (2025). General Terms and Conditions. *Quick Guide Legal Best Practices for Founders in Germany*, 79–80. https://doi.org/10.1007/978-3-658-47385-3_10

Rolls Royce. (2021). *mtu6R0150 DS300*. 4.

Sun, W., Liu, X., & Yang, L. (2020). An optimization method for economical ship-routing and

ship operation considering the effect of wind-assisted rotors. *Proceedings of the International Conference on Offshore Mechanics and Arctic Engineering - OMAE*, *6A-2020*, 1–8. https://doi.org/10.1115/OMAE2020-18776

Trading Economics. (2025). *Cotización del carbón*. https://es.tradingeconomics.com/commodity/coal

Wärtsilä. (2025). *Marine Engine Configuration*. https://www.wartsila.com/marine/engine-configurator

ANEXO

Algoritmo de Postorcesamiento de diagramas polares

```
import numpy as np
import pandas as pd
from scipy.interpolate import CubicSpline

velocity_wind = .0 # velocidad del viento en m/s
velocity_angle = df_final_para_plotear['velocity_angle']
velocity_magnitude = df_final_para_plotear['velocity_magnitude']

d = 20.0
c = 7.0

alpha = np.zeros(len(velocity_angle))
Sef = np.zeros(len(velocity_angle))
ratio = np.zeros(len(velocity_angle))
Vred = velocity_magnitude.copy()

min_shield_angle_deg = 0.0
max_shield_angle_deg = 80.0

min_shield_angle_rad = np.deg2rad(min_shield_angle_deg)
max_shield_angle_rad = np.deg2rad(max_shield_angle_deg)

for i in range(len(velocity_angle)):
    denominator_for_alpha = np.sqrt(
        velocity_magnitude.iloc[i]**2 +
        velocity_wind**2 -
        2 * velocity_magnitude.iloc[i] * velocity_wind *
np.cos(velocity_angle.iloc[i])
    )

    arcsin_arg = (velocity_magnitude.iloc[i] * np.sin(velocity_angle.iloc[i]))
/ denominator_for_alpha
    alpha[i] = np.arcsin(arcsin_arg)

    Sef[i] = np.sin(alpha[i]) * d
    ratio[i] = (3 * Sef[i] + c) / (4 * c)
    if ratio [i] >1 :
        ratio [i]=1

    if min_shield_angle_rad <= velocity_angle.iloc[i] <= max_shield_angle_rad:
        Vred.iloc[i] = velocity_magnitude.iloc[i] * np.sqrt(ratio[i])
#print (Vred)
#print (velocity_angle)
```

```
x = velocity_angle
y = Vred
cs3 = CubicSpline(x, y, bc_type='natural')
xs = np.linspace(min(x), max(x), 200)
ax = plt.subplot(111, projection='polar')
ax.plot(xs, cs3(xs), 'g-', label="L3")
ax.set_theta_zero_location('S')  # Establecer 0 grados en la parte inferior
(Sur)
ax.set_title('Diagrama Polar del Módulo de la Velocidad vs Ángulo de la
Velocidad')
plt.show()
```

Algoritmo final de weather routing (ida)

```
import pandas as pd
import re
from math import  pi
import numpy as np
import random
from math import radians, degrees, sin, cos, atan2 as math_atan2, asin, sqrt,
acos
from collections import deque
import time
import netCDF4 as nc
from datetime import datetime, timedelta
import csv
import os
print("Directorio actual desde donde corre el script:", os.getcwd())
# ------------------------------------------------------------------
# ------ PARÁMETROS PRINCIPALES PARA LA SIMULACIÓN --------
# ------------------------------------------------------------------

# Fichero donde se guardarán todos los resultados
NOMBRE_ARCHIVO_RESULTADOS = "resultados_simulaciones0107.csv"

# Intervalo en horas entre el inicio de cada nueva simulación
INTERVALO_HORAS_SIMS = 1

# Rango de fechas para las simulaciones
FECHA_INICIO_SIMS_STR = "20240701"
FECHA_FIN_SIMS_STR = "20241231"
HORA_INICIO_VIAJE = 00
# ---------------------------
# Parámetros para 1 simulación
# ---------------------------
inicio = (51.73, 3.49) # Rotterdam
#zona_llegada_rect = {'lat_min': 45.3, 'lat_max':45.5, 'lon_min':-40.2 ,
'lon_max': -39.8} # Pruebas
zona_llegada_rect = {'lat_min': 36.6, 'lat_max': 37.2, 'lon_min': -76.4,
'lon_max': -75.9} # Virginia
profundidad_max = 0
# --- MODIFICACIÓN: Añadir Checkpoint ---
CHECKPOINT_CANAL = {'lat': 49, 'lon': -7.39} # Punto imaginario en la entrada
del Canal de la Mancha

puntos_limite_superior = [
    (-76.8, 40.5), (-70, 41.5), (-65, 43), (-52, 46.1),
    (-45, 55), (-20, 52), (-9.52, 51.42), (-6.37, 49.89),(-5.14, 49.89), (0.39,
50.76),(1.39, 51.09), (5, 53)
]
puntos_limite_inferior = [
```

```
    (-76.8, 25), (-70, 25), (-65, 25), (-56, 25),
    (-45, 25), (-30, 25), (-20, 25),(-5.53,48.61),(-3.44, 49.06),(-2.37,
49.81),(0.33, 49.89), ( 1.57, 50.93), ( 3.16, 51.46), (3.87, 51.90 ), (5, 50)
]
zonas_criticas = [{'lat_min': 43.75, 'lat_max': 41.1, 'lon_min': -60.4,
'lon_max': -59.4}]
VISITED_DECIMAL_PLACES = 3

PODA_COLA_CADA_N_HORAS = 12
GRADOS_LONGITUD_MAX_RETRASO = 3.0

BEAM_WIDTH = 2000
PODA_BEAM_CADA_N_EXPANDIDOS = 1000

FECHA_INICIO_VIAJE_STR = "20240101"
## La siguiente linea ha sido introducida para poder leer los archivos en
margerit
carpeta_script = os.path.dirname(os.path.abspath(__file__))
RUTA_BASE_ARCHIVOS_VIENTO = carpeta_script
NOMBRE_ARCHIVO_VIENTO_PLANTILLA =
"MERRA2_400.tavg1_2d_flx_Nx.{fecha}_reducido.nc4"

cached_wind_file_path = None
cached_wind_dataset = None
cached_latitudes = None
cached_longitudes = None
#Nombres de archivos de diagramas polares
nombres_de_archivos_csv = [
    'dp_5_nudos.csv', 'dp_10_nudos.csv', 'dp_15_nudos.csv', 'dp_20_nudos.csv',
    'dp_25_nudos.csv', 'dp_30_nudos.csv', 'dp_35_nudos.csv'
]
# ----------------------------
# Funciones para diagramas polares
# ----------------------------

#Cargamos los diagramas polares y poder generar rutas realistas

def cargar_datos_polares(lista_de_archivos):
    polar_data = {}
    for archivo in lista_de_archivos:
        ruta_completa = os.path.join(carpeta_script, archivo)
        try:
            match = re.search(r'\d+', archivo)
            if not match: continue
            velocidad_viento = int(match.group(0))
            df = pd.read_csv(ruta_completa)
            if 'velocity_angle' in df.columns and 'velocity_magnitude' in
df.columns:
```

```
                polar_data[velocidad_viento] = list(zip(df['velocity_angle'],
df['velocity_magnitude']))
    return polar_data

#Esta función calcula la velocidad del buque para un ángulo dado sabiendo la
velocidad del viento de los diagramas polares
#(en este caso aquí entraría con 5,10,15,20,25,30 o 35 )
def _get_speed_at_angle(speed_data, angle):
    speed_data.sort(key=lambda x: x[0])
    angles = [point[0] for point in speed_data]
    speeds = [point[1] for point in speed_data]
    return np.interp(angle, angles, speeds)

#Esta función calcula la velocidad (haciendo uso de la función
_get_speed_at_angle) para la velocidad de viento específica
#Los inputs necesarios son la velocidad del viento, el ángulo de origen del
viento y los datos de diagramas polares cargados
#anteriormente
def calcular_velocidad_barco(velocidad_viento, angulo_viento, polar_data):
    if not polar_data: return 0.0

    angulo_base = angulo_viento % (2 * pi)

    if 2.61799 <= angulo_base <= 3.66519: return 0.0

    if angulo_base > pi:
        angulo_calculo = 2 * pi - angulo_base
    else:
        angulo_calculo = angulo_base

    available_wind_speeds = sorted(polar_data.keys())
    if not available_wind_speeds or velocidad_viento >
max(available_wind_speeds) or velocidad_viento < min(available_wind_speeds):
        return 0.0

    if velocidad_viento in available_wind_speeds:
        return _get_speed_at_angle(polar_data[velocidad_viento],
angulo_calculo)
    else:
        for i in range(len(available_wind_speeds) - 1):
            lower_wind, upper_wind = available_wind_speeds[i],
available_wind_speeds[i+1]
            if lower_wind < velocidad_viento < upper_wind:
                speed_lower = _get_speed_at_angle(polar_data[lower_wind],
angulo_calculo)
                speed_upper = _get_speed_at_angle(polar_data[upper_wind],
angulo_calculo)
                return (speed_lower + speed_upper) / 2
    return 0.0
```

```
# ----------------------------
# Funciones para Datos de Viento
# ----------------------------
def get_wind_data_from_file(file_path, lat, lon, hour_in_day):
    global cached_wind_file_path, cached_wind_dataset, cached_latitudes,
cached_longitudes
    if file_path != cached_wind_file_path or cached_wind_dataset is None:
        try:
            if cached_wind_dataset: cached_wind_dataset.close()
            dataset = nc.Dataset(file_path)
            cached_wind_dataset = dataset
            cached_wind_file_path = file_path
            cached_latitudes = dataset.variables['lat'][:]
            cached_longitudes = dataset.variables['lon'][:]
            # print(f"   INFO Viento: Abierto y cacheado:
{file_path.split('/')[-1]}")
            if cached_wind_dataset: cached_wind_dataset.close()
            cached_wind_dataset = None; cached_wind_file_path = None
            return None, None
    else:
        dataset = cached_wind_dataset
    try:
        lat_idx = (np.abs(cached_latitudes - lat)).argmin()
        lon_idx = (np.abs(cached_longitudes - lon)).argmin()
        u_wind = dataset.variables['ULML'][hour_in_day, lat_idx, lon_idx]
        v_wind = dataset.variables['VLML'][hour_in_day, lat_idx, lon_idx]
        if hasattr(u_wind, 'mask') and u_wind.mask: u_wind = u_wind.filled(0.0)
        if hasattr(v_wind, 'mask') and v_wind.mask: v_wind = v_wind.filled(0.0)
        return float(u_wind), float(v_wind)
    except IndexError:
        # print(f"   ERROR Viento: Índices fuera de rango para
{file_path.split('/')[-1]} en ({lat:.2f},{lon:.2f},h{hour_in_day}).")
        return None, None
        return None, None

def get_wind_at_node(lat, lon, current_datetime_utc):
    fecha_str = current_datetime_utc.strftime("%Y%m%d")
    hora_del_dia = current_datetime_utc.hour
    file_name = NOMBRE_ARCHIVO_VIENTO_PLANTILLA.format(fecha=fecha_str)
    file_path = os.path.join(carpeta_script,file_name)
    return get_wind_data_from_file(file_path, lat, lon, hora_del_dia)

# ----------------------------
# Funciones Geográficas y Auxiliares
# ----------------------------
#Calcúla la distancia en km de un punto a otro
def haversine_km(lat1, lon1, lat2, lon2):
```

```
    R_KM = 6371.0  # Radio de la Tierra en kilómetros
    lat1_rad, lon1_rad = radians(lat1), radians(lon1)
    lat2_rad, lon2_rad = radians(lat2), radians(lon2)

    dlon = lon2_rad - lon1_rad
    dlat = lat2_rad - lat1_rad

    a = sin(dlat / 2)**2 + cos(lat1_rad) * cos(lat2_rad) * sin(dlon / 2)**2
    c = 2 * asin(sqrt(a))

    distancia = R_KM * c
    return distancia

def is_node_valid_against_boundary(node_lat, node_lon, boundary_points,
is_upper_boundary):
    if not boundary_points or len(boundary_points) < 2: return True
    min_lon_boundary = boundary_points[0][0]; max_lon_boundary =
boundary_points[-1][0]; epsilon = 1e-9
    if node_lon < min_lon_boundary - epsilon or node_lon > max_lon_boundary +
epsilon: return False
    for i in range(len(boundary_points) - 1):
        p1_lon, p1_lat = boundary_points[i]; p2_lon, p2_lat =
boundary_points[i+1]
        if (p1_lon - epsilon) <= node_lon <= (p2_lon + epsilon) or \
           (p2_lon - epsilon) <= node_lon <= (p1_lon + epsilon): # Permitir
invertir el orden de puntos del límite
            if abs(p1_lon - p2_lon) < epsilon: # Segmento vertical
                if abs(node_lon - p1_lon) < epsilon: # El nodo está en la
vertical del segmento
                    return not (is_upper_boundary and node_lat > max(p1_lat,
p2_lat) + epsilon) and \
                               not (not is_upper_boundary and node_lat <
min(p1_lat, p2_lat) - epsilon)
            else: # Segmento no vertical
                lat_on_line = p1_lat + (p2_lat - p1_lat) * (node_lon - p1_lon)
/ (p2_lon - p1_lon)
                return not (is_upper_boundary and node_lat > lat_on_line +
epsilon) and \
                           not (not is_upper_boundary and node_lat < lat_on_line -
epsilon)
    return False

def en_corredor_lineal(lat_nuevo, lon_nuevo, upper_boundary_points,
lower_boundary_points):
    if not is_node_valid_against_boundary(lat_nuevo, lon_nuevo,
upper_boundary_points, True): return False
    return is_node_valid_against_boundary(lat_nuevo, lon_nuevo,
lower_boundary_points, False)
```

```
def en_zona_llegada(lat, lon):
    return (zona_llegada_rect['lat_min'] <= lat <= zona_llegada_rect['lat_max']
and
            zona_llegada_rect['lon_min'] <= lon <=
zona_llegada_rect['lon_max'])

def en_zona_critica(lat, lon):
    for zona in zonas_criticas:
        if zona['lon_min'] <= lon <= zona['lon_max'] and zona['lat_min'] <= lat
<= zona['lat_max']: return True
    return False


def mover(lat, lon, rumbo_grados, distancia_km):
    R_KM = 6371.0; rumbo = radians(rumbo_grados)
    lat1, lon1 = radians(lat), radians(lon)
    lat2 = asin(sin(lat1) * cos(distancia_km / R_KM) + cos(lat1) *
sin(distancia_km / R_KM) * cos(rumbo))
    lon2 = lon1 + math_atan2(sin(rumbo) * sin(distancia_km / R_KM) * cos(lat1),
cos(distancia_km / R_KM) - sin(lat1) * sin(lat2))
    return degrees(lat2), degrees(lon2)

# ----------------------------
# Funciones de generación de rutas basadas en diagramas polares
# ----------------------------
def obtener_rutas(lat, lon, current_datetime_utc, polar_data):
    # 1. Obtener datos de viento del modelo (Merra)
    u_wind, v_wind = get_wind_at_node(lat, lon, current_datetime_utc)
    if u_wind is None or v_wind is None:
        return []

    # 2. Calcular velocidad y dirección del viento
    wind_speed_mps = sqrt(u_wind**2 + v_wind**2)
    wind_speed_knots = wind_speed_mps * 1.94384

    # La dirección DE LA QUE VIENE el viento, en radianes estándar (0=Este,
pi/2=Norte).
    wind_from_direction_rad = math_atan2(v_wind, u_wind) + pi

    # 3. Definir rumbos deseados por el usuario
    angulos_rumbo_usuario = [-30, -60, -90, -120, -150]
    rutas_generadas = []

    # 4. Para cada rumbo posible, calcular la velocidad y distancia
    for rumbo_usuario_deg in angulos_rumbo_usuario:

        # Lógica de ángulos
        rumbo_estandar_rad = radians(rumbo_usuario_deg + 90)
        diff = rumbo_estandar_rad - wind_from_direction_rad
```

```
        angulo_viento_barco_rad = abs(diff)
        if angulo_viento_barco_rad > pi:
            angulo_viento_barco_rad = 2 * pi - angulo_viento_barco_rad

        # Variables finales que se usarán para el cálculo y guardado
        angulo_final_para_calculo = angulo_viento_barco_rad
        rumbo_final_a_guardar_deg = rumbo_usuario_deg

        # --- LÓGICA DE CORRECCIÓN DE RUMBO (ANTI-APROADA) ---
        MIN_ANGLE_OFF_WIND_RAD = radians(32) # Nuevo ángulo mínimo: 32 grados
        NO_GO_ZONE_START_RAD = pi - MIN_ANGLE_OFF_WIND_RAD # Empieza a ~148
grados

        if angulo_viento_barco_rad > NO_GO_ZONE_START_RAD:
            # 1. El ángulo para calcular la velocidad será el mejor posible.
            angulo_final_para_calculo = NO_GO_ZONE_START_RAD

            # 2. Calcular el nuevo rumbo corregido.
            wind_to_direction_rad = wind_from_direction_rad - pi # Hacia dónde
sopla el viento

            # Determinar si el rumbo deseado era a babor o estribor del viento
            # para elegir el lado correcto de la virada.
            angular_diff_from_downwind = (rumbo_estandar_rad -
wind_to_direction_rad + pi) % (2*pi) - pi

            if angular_diff_from_downwind > 0: # Virada a babor (respecto al
viento)
                rumbo_corregido_estandar_rad = wind_to_direction_rad +
MIN_ANGLE_OFF_WIND_RAD
            else: # Virada a estribor
                rumbo_corregido_estandar_rad = wind_to_direction_rad -
MIN_ANGLE_OFF_WIND_RAD

            # 3. Convertir el rumbo corregido de vuelta al sistema del usuario.
            rumbo_final_a_guardar_deg = (degrees(rumbo_corregido_estandar_rad)
- 90 + 360) % 360

        # 5. Usar la función del diagrama polar
        velocidad_barco_mps = calcular_velocidad_barco(wind_speed_knots,
angulo_final_para_calculo, polar_data)

        # 6. Calcular la distancia
        distancia_km_por_hora = (velocidad_barco_mps * 3600) / 1000

        rutas_generadas.append((rumbo_final_a_guardar_deg,
distancia_km_por_hora))
    #print(u_wind, v_wind) Era un Prueba
    return rutas_generadas
```

```
def reconstruct_path(end_node_id, nodes_store): # Para las coordenadas del plot
    path_coords = []; current_id = end_node_id
    while current_id is not None:
        node_info = nodes_store[current_id]
        path_coords.append(node_info['pos'])
        current_id = node_info['parent_id']
    return path_coords[::-1]

# ---- MODIFICACIÓN: Nueva función para reconstruir con detalles ----
def reconstruct_detailed_path(end_node_id, nodes_store):
    path_details = []
    current_id = end_node_id
    while current_id is not None:
        node_info = nodes_store[current_id]
        path_details.append({
            'pos': node_info['pos'],
            'steps': node_info['steps'],
            'costo_acumulado': node_info['costo_acumulado']
        })
        current_id = node_info['parent_id']
    return path_details[::-1] # De inicio a fin

# ----------------------------
# Algoritmo tipo BFS con Podas y Viento Real
# ----------------------------
def routing_bfs_real_wind(inicio_lat, inicio_lon, fecha_hora_inicio_utc,
profundidad_max_param, polar_data, ax=None):
     # --- MODIFICACIÓN: Calcular punto objetivo y distancia inicial ---
    lat_centro_llegada = (zona_llegada_rect['lat_min'] +
zona_llegada_rect['lat_max']) / 2
    lon_centro_llegada = (zona_llegada_rect['lon_min'] +
zona_llegada_rect['lon_max']) / 2
    distancia_inicial_a_destino = haversine_km(inicio_lat, inicio_lon,
lat_centro_llegada, lon_centro_llegada)
   # Calculamos también la distancia al checkpoint
    distancia_inicial_a_checkpoint = haversine_km(inicio_lat, inicio_lon,
CHECKPOINT_CANAL['lat'], CHECKPOINT_CANAL['lon'])

    print(f"Objetivo Final: Hampton Roads ({lat_centro_llegada:.2f},
{lon_centro_llegada:.2f})")
    print(f"Checkpoint Intermedio: Canal ({CHECKPOINT_CANAL['lat']:.2f},
{CHECKPOINT_CANAL['lon']:.2f})")

    queue = deque()
    nodes_store = {}; next_node_id = 0

    start_node_id = next_node_id; next_node_id += 1
    costo_inicio = 0.0
```

```
    nodes_store[start_node_id] = {'pos': (inicio_lat, inicio_lon), 'parent_id':
None,
                                  'costo_acumulado': costo_inicio, 'steps': 0}
    queue.append((costo_inicio, inicio_lat, inicio_lon, start_node_id, 0))

    mejor_tiempo_global = float('inf'); mejor_nodo_final_id = None
    min_steps_to_target_found = float('inf')

    nodos_expandidos_count = 0

    start_time_search = time.time()
    explorados_lons_para_scatter = []
    explorados_lats_para_scatter = []

    while queue:
        costo_actual, lat, lon, current_node_id, steps_actuales =
queue.popleft()
        nodos_expandidos_count += 1
        current_datetime = fecha_hora_inicio_utc +
timedelta(hours=steps_actuales)


        if steps_actuales >= mejor_tiempo_global:
            continue
        if steps_actuales > min_steps_to_target_found:
            continue
        if profundidad_max_param > 0 and steps_actuales >=
profundidad_max_param:
            continue


        for angulo, dist_segmento_efectiva in obtener_rutas(lat, lon,
current_datetime, polar_data):
           #print(lat, lon)
           #print (angulo)
           #print (dist_segmento_efectiva)
            lat_nuevo, lon_nuevo = mover(lat, lon, angulo,
dist_segmento_efectiva)
           #print(lat_nuevo,lon_nuevo)
            costo_nuevo = costo_actual + dist_segmento_efectiva
           #print(costo_nuevo)
            nuevos_steps = steps_actuales + 1

            if nuevos_steps > min_steps_to_target_found:
                continue
            if profundidad_max_param > 0 and nuevos_steps >
profundidad_max_param:
                continue
```

```
            if not en_corredor_lineal(lat_nuevo, lon_nuevo,
puntos_limite_superior, puntos_limite_inferior):
                continue
            if en_zona_critica(lat_nuevo, lon_nuevo):
                 continue

            new_node_id = next_node_id; next_node_id += 1
            nodes_store[new_node_id] = {'pos': (lat_nuevo, lon_nuevo),
'parent_id': current_node_id,
                                        'costo_acumulado': costo_nuevo,
'steps': nuevos_steps}
            explorados_lons_para_scatter.append(lon_nuevo);
explorados_lats_para_scatter.append(lat_nuevo)

            if en_zona_llegada(lat_nuevo, lon_nuevo):
                if nuevos_steps < mejor_tiempo_global:
                    print(f"   ** Destino alcanzado en {nuevos_steps}h.
Optimizando costo en este nivel de horas. **")
                    mejor_tiempo_global = nuevos_steps; mejor_nodo_final_id =
new_node_id
                    print(f"     Nueva mejor ruta ({nuevos_steps}h). Costo:
{mejor_tiempo_global:.2f} MN, Nodo: {new_node_id}")
            else:
                if (profundidad_max_param == 0 or nuevos_steps <
profundidad_max_param) and \
                   nuevos_steps < mejor_tiempo_global:
                    queue.append((costo_nuevo, lat_nuevo, lon_nuevo,
new_node_id, nuevos_steps))

        # --- PODA BEAM SEARCH DIVERSIFICADA (Disparador por Nodos Expandidos)
---
        if nodos_expandidos_count > 0 and nodos_expandidos_count %
PODA_BEAM_CADA_N_EXPANDIDOS == 0:

            # --- Parámetros de la Poda Diversificada ---
            N_MEJORES = 1900      # Nº de rutas "élite" a mantener
(Explotación)
            N_ALEATORIOS = 100    # Nº de rutas "comodín" a mantener
(Exploración)
            BEAM_WIDTH = N_MEJORES + N_ALEATORIOS

            # Solo podar si la cola supera el tamaño total deseado
            if len(queue) > BEAM_WIDTH:


                # --- MODIFICACIÓN: Nueva función de promesa basada en VMG a
destino ---
                def calcular_promesa(nodo_tupla):
                    lat_actual = nodo_tupla[1]
```

```
                    lon_actual = nodo_tupla[2]
                    horas_viaje = nodo_tupla[4]

                    if horas_viaje == 0:
                        return -float('inf')

                    # --- Lógica IF/ELSE para cambiar de objetivo ---
                    if lon_actual >= CHECKPOINT_CANAL['lon']:
                        # Si aún no hemos pasado la longitud del checkpoint,
nuestro objetivo es el checkpoint.
                        distancia_actual_al_objetivo = haversine_km(lat_actual,
lon_actual, CHECKPOINT_CANAL['lat'], CHECKPOINT_CANAL['lon'])
                        progreso_hacia_objetivo =
distancia_inicial_a_checkpoint - distancia_actual_al_objetivo
                    else:
                        # Si ya hemos pasado el checkpoint, nuestro objetivo es
Rotterdam.
                        distancia_actual_al_objetivo = haversine_km(lat_actual,
lon_actual, lat_centro_llegada, lon_centro_llegada)
                        progreso_hacia_objetivo = distancia_inicial_a_destino -
distancia_actual_al_objetivo

                        # La puntuación es la "velocidad de acercamiento" al
objetivo actual
                    return progreso_hacia_objetivo / horas_viaje

                # Ordenamos toda la cola de mayor a menor "promesa"
                sorted_queue_list = sorted(list(queue), key=calcular_promesa,
reverse=True)

                # 1. Nos quedamos con los N_MEJORES nodos (Explotación)
                supervivientes_mejores = sorted_queue_list[:N_MEJORES]

                # 2. Seleccionamos candidatos para la parte aleatoria
                pool_aleatorio = sorted_queue_list[N_MEJORES:N_MEJORES + 1000]

                supervivientes_aleatorios = []
                if pool_aleatorio:
                    num_a_escoger = min(N_ALEATORIOS, len(pool_aleatorio))
                    supervivientes_aleatorios = random.sample(pool_aleatorio,
num_a_escoger)

                # 3. La nueva cola es la combinación de los mejores y los
aleatorios
                queue = deque(supervivientes_mejores +
supervivientes_aleatorios)

                #La siguiente fila se utilizaba para comprobar que el código no
se paraba
```

```
                #if n_descartados_beam_actual > 0:
                    #print(f"   PODA BEAM DIVERSIFICADA (N_Exp
{nodos_expandidos_count}): Cola de {num_antes_beam} a {len(queue)}
({n_descartados_beam_actual} desc.)")


        if nodos_expandidos_count % 20000 == 0: # Aumentado para reducir output
            elapsed_search_time = time.time() - start_time_search
            print(f"Nodos exp: {nodos_expandidos_count}, Cola: {len(queue)},
Mejor Costo: {mejor_tiempo_global:.2f}, "
                  f"Min Horas Dest: {min_steps_to_target_found if
min_steps_to_target_found != float('inf') else 'N/A'}, "
                  f"Tiempo Sim: {elapsed_search_time:.2f}s, Fecha actual sim:
{current_datetime.strftime('%Y-%m-%d %H:%M')}")
    # ---- Preparar y devolver datos detallados de la ruta ----
    detailed_path_result = None
    if mejor_nodo_final_id is not None:

        steps_de_la_mejor_ruta = nodes_store[mejor_nodo_final_id]['steps']
        detailed_path_result = reconstruct_detailed_path(mejor_nodo_final_id,
nodes_store) # Para la tabla
        return  steps_de_la_mejor_ruta, detailed_path_result
    else:
        global cached_wind_dataset
        if cached_wind_dataset:
            cached_wind_dataset.close(); cached_wind_dataset = None
        return None, None # Devolver None para detailed_path_result

# ----------------------------
# Cargamos diagramas polares
# ----------------------------
print("Iniciando la carga de datos polares...")
datos_polares_cargados = cargar_datos_polares(nombres_de_archivos_csv)
# ----------------------------
# Bucle Principal de Simulación por Lotes
# ----------------------------


def ejecutar_simulaciones():

    try:
        fecha_inicio_lote = datetime.strptime(FECHA_INICIO_SIMS_STR, "%Y%m%d")
        fecha_fin_lote = datetime.strptime(FECHA_FIN_SIMS_STR,
"%Y%m%d").replace(hour=23)

    current_start_time = fecha_inicio_lote
    current_start_time += timedelta(hours=HORA_INICIO_VIAJE)
    file_exists = os.path.isfile(NOMBRE_ARCHIVO_RESULTADOS)
```

```
    with open(NOMBRE_ARCHIVO_RESULTADOS, 'a', newline='', encoding='utf-8') as
f:
        writer = csv.writer(f)
        if not file_exists:
            writer.writerow(["fecha_hora_salida", "tiempo_computacion_seg",
"tiempo_viaje_horas", "fecha_hora_llegada"])

        sim_count = 0
        while current_start_time <= fecha_fin_lote:

            tiempo_inicio_sim = time.time()

            mejor_tiempo, detailed_path = routing_bfs_real_wind(
                inicio[0], inicio[1], current_start_time,profundidad_max,
datos_polares_cargados
            )

            tiempo_computacion = time.time() - tiempo_inicio_sim

            if mejor_tiempo is not None:
                fecha_llegada = current_start_time +
timedelta(hours=mejor_tiempo)
                resultado = [current_start_time.strftime('%Y-%m-%d %H:%M:%S'),
round(tiempo_computacion, 2), mejor_tiempo,  fecha_llegada.strftime('%Y-%m-%d
%H:%M:%S')]
                writer.writerow(resultado)
                print(f"-> ÉXITO. T. Viaje: {mejor_tiempo} h | T. Cómputo:
{tiempo_computacion:.1f} s")
            else:
                resultado = [current_start_time.strftime('%Y-%m-%d %H:%M:%S'),
round(tiempo_computacion, 2), "N/A", "N/A", "N/A"]
                writer.writerow(resultado)
                print(f"-> FALLO. No se encontró ruta ")

            f.flush()
            current_start_time += timedelta(hours=INTERVALO_HORAS_SIMS)

    if cached_wind_dataset:
        cached_wind_dataset.close()

ejecutar_simulaciones()
```

Algoritmo final de weather routing (vuelta)

```
import pandas as pd
import re
from math import  pi
import numpy as np
import random
```

```
from math import radians, degrees, sin, cos, atan2 as math_atan2, asin, sqrt,
acos
from collections import deque
import time
import netCDF4 as nc
from datetime import datetime, timedelta
import csv
import os
print("Directorio actual desde donde corre el script:", os.getcwd())
# ------------------------------------------------------------------
# ------ PARÁMETROS PRINCIPALES PARA LA SIMULACIÓN POR LOTES --------
# ------------------------------------------------------------------

# Fichero donde se guardarán todos los resultados
NOMBRE_ARCHIVO_RESULTADOS = "resultados_simulaciones0601.csv"

# Intervalo en horas entre el inicio de cada nueva simulación
# Ej: 6 -> 00:00, 06:00, 12:00, 18:00...
INTERVALO_HORAS_SIMS = 1

# Rango de fechas para las simulaciones
FECHA_INICIO_SIMS_STR = "20240106"
FECHA_FIN_SIMS_STR = "20241231"
HORA_INICIO_VIAJE = 00
# ----------------------------
# Parámetros para 1 simulación
# ----------------------------
inicio = (51.73, 3.49) # Rotterdam
#zona_llegada_rect = {'lat_min': 45.3, 'lat_max':45.5, 'lon_min':-40.2 ,
'lon_max': -39.8} # Pruebas
zona_llegada_rect = {'lat_min': 36.6, 'lat_max': 37.2, 'lon_min': -76.4,
'lon_max': -75.9} # Virginia
profundidad_max = 0
# --- MODIFICACIÓN: Añadir Checkpoint ---
CHECKPOINT_CANAL = {'lat': 49, 'lon': -7.39} # Punto imaginario en la entrada
del Canal de la Mancha

puntos_limite_superior = [
    (-76.8, 40.5), (-70, 41.5), (-65, 43), (-52, 46.1),
    (-45, 55), (-20, 52), (-9.52, 51.42), (-6.37, 49.89),(-5.14, 49.89), (0.39,
50.76),(1.39, 51.09), (5, 53)
]
puntos_limite_inferior = [
    (-76.8, 25), (-70, 25), (-65, 25), (-56, 25),
    (-45, 25), (-30, 25), (-20, 25),(-5.53,48.61),(-3.44, 49.06),(-2.37,
49.81),(0.33, 49.89), ( 1.57, 50.93), ( 3.16, 51.46), (3.87, 51.90 ), (5, 50)
]
zonas_criticas = [{'lat_min': 43.75, 'lat_max': 41.1, 'lon_min': -60.4,
'lon_max': -59.4}]
```

```
VISITED_DECIMAL_PLACES = 3

PODA_COLA_CADA_N_HORAS = 12
GRADOS_LONGITUD_MAX_RETRASO = 3.0

BEAM_WIDTH = 2000
PODA_BEAM_CADA_N_EXPANDIDOS = 1000

FECHA_INICIO_VIAJE_STR = "20240101"
## La siguiente linea ha sido introducida para poder leer los archivos en
margerit
carpeta_script = os.path.dirname(os.path.abspath(__file__))
RUTA_BASE_ARCHIVOS_VIENTO = carpeta_script
NOMBRE_ARCHIVO_VIENTO_PLANTILLA =
"MERRA2_400.tavg1_2d_flx_Nx.{fecha}_reducido.nc4"

cached_wind_file_path = None
cached_wind_dataset = None
cached_latitudes = None
cached_longitudes = None
#Nombres de archivos de diagramas polares
nombres_de_archivos_csv = [
    'dp_5_nudos.csv', 'dp_10_nudos.csv', 'dp_15_nudos.csv', 'dp_20_nudos.csv',
    'dp_25_nudos.csv', 'dp_30_nudos.csv', 'dp_35_nudos.csv'
]
# ----------------------------
# Funciones para diagramas polares
# ----------------------------

#Cargamos los diagramas polares y poder generar rutas realistas

def cargar_datos_polares(lista_de_archivos):
    polar_data = {}
    for archivo in lista_de_archivos:
        ruta_completa = os.path.join(carpeta_script, archivo)
        try:
            match = re.search(r'\d+', archivo)
            if not match: continue
            velocidad_viento = int(match.group(0))
            df = pd.read_csv(ruta_completa)
            if 'velocity_angle' in df.columns and 'velocity_magnitude' in
df.columns:
                polar_data[velocidad_viento] = list(zip(df['velocity_angle'],
df['velocity_magnitude']))
    return polar_data

#Esta función calcula la velocidad del buque para un ángulo dado sabiendo la
velocidad del viento de los diagramas polares
#(en este caso aquí entraría con 5,10,15,20,25,30 o 35 )
```

```
def _get_speed_at_angle(speed_data, angle):
    speed_data.sort(key=lambda x: x[0])
    angles = [point[0] for point in speed_data]
    speeds = [point[1] for point in speed_data]
    return np.interp(angle, angles, speeds)

#Esta función calcula la velocidad (haciendo uso de la función
_get_speed_at_angle) para la velocidad de viento específica
#Los inputs necesarios son la velocidad del viento, el ángulo de origen del
viento y los datos de diagramas polares cargados
#anteriormente
def calcular_velocidad_barco(velocidad_viento, angulo_viento, polar_data):
    if not polar_data: return 0.0

    angulo_base = angulo_viento % (2 * pi)

    if 2.61799 <= angulo_base <= 3.66519: return 0.0

    if angulo_base > pi:
        angulo_calculo = 2 * pi - angulo_base
    else:
        angulo_calculo = angulo_base

    available_wind_speeds = sorted(polar_data.keys())
    if not available_wind_speeds or velocidad_viento >
max(available_wind_speeds) or velocidad_viento < min(available_wind_speeds):
        return 0.0

    if velocidad_viento in available_wind_speeds:
        return _get_speed_at_angle(polar_data[velocidad_viento],
angulo_calculo)
    else:
        for i in range(len(available_wind_speeds) - 1):
            lower_wind, upper_wind = available_wind_speeds[i],
available_wind_speeds[i+1]
            if lower_wind < velocidad_viento < upper_wind:
                speed_lower = _get_speed_at_angle(polar_data[lower_wind],
angulo_calculo)
                speed_upper = _get_speed_at_angle(polar_data[upper_wind],
angulo_calculo)
                return (speed_lower + speed_upper) / 2


# ----------------------------
# Funciones para Datos de Viento
# ----------------------------
def get_wind_data_from_file(file_path, lat, lon, hour_in_day):
    global cached_wind_file_path, cached_wind_dataset, cached_latitudes,
cached_longitudes
```

```
    if file_path != cached_wind_file_path or cached_wind_dataset is None:

            if cached_wind_dataset: cached_wind_dataset.close()
            dataset = nc.Dataset(file_path)
            cached_wind_dataset = dataset
            cached_wind_file_path = file_path
            cached_latitudes = dataset.variables['lat'][:]
            cached_longitudes = dataset.variables['lon'][:]
            return None, None
            if cached_wind_dataset: cached_wind_dataset.close()
            cached_wind_dataset = None; cached_wind_file_path = None
            return None, None
    else:
        dataset = cached_wind_dataset

        lat_idx = (np.abs(cached_latitudes - lat)).argmin()
        lon_idx = (np.abs(cached_longitudes - lon)).argmin()
        u_wind = dataset.variables['ULML'][hour_in_day, lat_idx, lon_idx]
        v_wind = dataset.variables['VLML'][hour_in_day, lat_idx, lon_idx]
        if hasattr(u_wind, 'mask') and u_wind.mask: u_wind = u_wind.filled(0.0)
        if hasattr(v_wind, 'mask') and v_wind.mask: v_wind = v_wind.filled(0.0)
        return float(u_wind), float(v_wind)

def get_wind_at_node(lat, lon, current_datetime_utc):
    fecha_str = current_datetime_utc.strftime("%Y%m%d")
    hora_del_dia = current_datetime_utc.hour
    file_name = NOMBRE_ARCHIVO_VIENTO_PLANTILLA.format(fecha=fecha_str)
    file_path = os.path.join(carpeta_script,file_name)
    return get_wind_data_from_file(file_path, lat, lon, hora_del_dia)

# ----------------------------
# Funciones Geográficas y Auxiliares
# ----------------------------
#Calcúla la distancia en km de un punto a otro
def haversine_km(lat1, lon1, lat2, lon2):
    R_KM = 6371.0  # Radio de la Tierra en kilómetros
    lat1_rad, lon1_rad = radians(lat1), radians(lon1)
    lat2_rad, lon2_rad = radians(lat2), radians(lon2)

    dlon = lon2_rad - lon1_rad
    dlat = lat2_rad - lat1_rad

    a = sin(dlat / 2)**2 + cos(lat1_rad) * cos(lat2_rad) * sin(dlon / 2)**2
    c = 2 * asin(sqrt(a))

    distancia = R_KM * c
    return distancia
```

```
def is_node_valid_against_boundary(node_lat, node_lon, boundary_points,
is_upper_boundary):
    if not boundary_points or len(boundary_points) < 2: return True
    min_lon_boundary = boundary_points[0][0]; max_lon_boundary =
boundary_points[-1][0]; epsilon = 1e-9
    if node_lon < min_lon_boundary - epsilon or node_lon > max_lon_boundary +
epsilon: return False
    for i in range(len(boundary_points) - 1):
        p1_lon, p1_lat = boundary_points[i]; p2_lon, p2_lat =
boundary_points[i+1]
        if (p1_lon - epsilon) <= node_lon <= (p2_lon + epsilon) or \
           (p2_lon - epsilon) <= node_lon <= (p1_lon + epsilon): # Permitir
invertir el orden de puntos del límite
            if abs(p1_lon - p2_lon) < epsilon: # Segmento vertical
                if abs(node_lon - p1_lon) < epsilon: # El nodo está en la
vertical del segmento
                    return not (is_upper_boundary and node_lat > max(p1_lat,
p2_lat) + epsilon) and \
                               not (not is_upper_boundary and node_lat <
min(p1_lat, p2_lat) - epsilon)
            else: # Segmento no vertical
                lat_on_line = p1_lat + (p2_lat - p1_lat) * (node_lon - p1_lon)
/ (p2_lon - p1_lon)
                return not (is_upper_boundary and node_lat > lat_on_line +
epsilon) and \
                           not (not is_upper_boundary and node_lat < lat_on_line -
epsilon)
    return False

def en_corredor_lineal(lat_nuevo, lon_nuevo, upper_boundary_points,
lower_boundary_points):
    if not is_node_valid_against_boundary(lat_nuevo, lon_nuevo,
upper_boundary_points, True): return False
    return is_node_valid_against_boundary(lat_nuevo, lon_nuevo,
lower_boundary_points, False)

def en_zona_llegada(lat, lon):
    return (zona_llegada_rect['lat_min'] <= lat <= zona_llegada_rect['lat_max']
and
            zona_llegada_rect['lon_min'] <= lon <=
zona_llegada_rect['lon_max'])

def en_zona_critica(lat, lon):
    for zona in zonas_criticas:
        if zona['lon_min'] <= lon <= zona['lon_max'] and zona['lat_min'] <= lat
<= zona['lat_max']: return True
    return False
```

```
def mover(lat, lon, rumbo_grados, distancia_km):
    R_KM = 6371.0; rumbo = radians(rumbo_grados)
    lat1, lon1 = radians(lat), radians(lon)
    lat2 = asin(sin(lat1) * cos(distancia_km / R_KM) + cos(lat1) *
sin(distancia_km / R_KM) * cos(rumbo))
    lon2 = lon1 + math_atan2(sin(rumbo) * sin(distancia_km / R_KM) * cos(lat1),
cos(distancia_km / R_KM) - sin(lat1) * sin(lat2))
    return degrees(lat2), degrees(lon2)

# ----------------------------
# Funciones de generación de rutas basadas en diagramas polares
# ----------------------------
def obtener_rutas(lat, lon, current_datetime_utc, polar_data):
    # 1. Obtener datos de viento del modelo (Merra)
    u_wind, v_wind = get_wind_at_node(lat, lon, current_datetime_utc)
    if u_wind is None or v_wind is None:
        return []

    # 2. Calcular velocidad y dirección del viento
    wind_speed_mps = sqrt(u_wind**2 + v_wind**2)
    wind_speed_knots = wind_speed_mps * 1.94384

    # La dirección DE LA QUE VIENE el viento, en radianes estándar (0=Este,
pi/2=Norte).
    wind_from_direction_rad = math_atan2(v_wind, u_wind) + pi

    # 3. Definir rumbos deseados por el usuario
    angulos_rumbo_usuario = [-30, -60, -90, -120, -150]
    rutas_generadas = []

    # 4. Para cada rumbo posible, calcular la velocidad y distancia
    for rumbo_usuario_deg in angulos_rumbo_usuario:

        # Lógica de ángulos
        rumbo_estandar_rad = radians(rumbo_usuario_deg + 90)
        diff = rumbo_estandar_rad - wind_from_direction_rad
        angulo_viento_barco_rad = abs(diff)
        if angulo_viento_barco_rad > pi:
            angulo_viento_barco_rad = 2 * pi - angulo_viento_barco_rad

        # Variables finales que se usarán para el cálculo y guardado
        angulo_final_para_calculo = angulo_viento_barco_rad
        rumbo_final_a_guardar_deg = rumbo_usuario_deg

        # --- LÓGICA DE CORRECCIÓN DE RUMBO (ANTI-APROADA) ---
        MIN_ANGLE_OFF_WIND_RAD = radians(32) # Nuevo ángulo mínimo: 32 grados
        NO_GO_ZONE_START_RAD = pi - MIN_ANGLE_OFF_WIND_RAD # Empieza a ~148
grados
```

```
        if angulo_viento_barco_rad > NO_GO_ZONE_START_RAD:
            # 1. El ángulo para calcular la velocidad será el mejor posible.
            angulo_final_para_calculo = NO_GO_ZONE_START_RAD

            # 2. Calcular el nuevo rumbo corregido.
            wind_to_direction_rad = wind_from_direction_rad - pi # Hacia dónde
sopla el viento

            # Determinar si el rumbo deseado era a babor o estribor del viento
            # para elegir el lado correcto de la virada.
            angular_diff_from_downwind = (rumbo_estandar_rad -
wind_to_direction_rad + pi) % (2*pi) - pi

            if angular_diff_from_downwind > 0: # Virada a babor (respecto al
viento)
                rumbo_corregido_estandar_rad = wind_to_direction_rad +
MIN_ANGLE_OFF_WIND_RAD
            else: # Virada a estribor
                rumbo_corregido_estandar_rad = wind_to_direction_rad -
MIN_ANGLE_OFF_WIND_RAD

            # 3. Convertir el rumbo corregido de vuelta al sistema del usuario.
            rumbo_final_a_guardar_deg = (degrees(rumbo_corregido_estandar_rad)
- 90 + 360) % 360

        # 5. Usar la función del diagrama polar
        velocidad_barco_mps = calcular_velocidad_barco(wind_speed_knots,
angulo_final_para_calculo, polar_data)

        # 6. Calcular la distancia
        distancia_km_por_hora = (velocidad_barco_mps * 3600) / 1000

        rutas_generadas.append((rumbo_final_a_guardar_deg,
distancia_km_por_hora))
    #print(u_wind, v_wind) Era un Prueba
    return rutas_generadas

def reconstruct_path(end_node_id, nodes_store): # Para las coordenadas del plot
    path_coords = []; current_id = end_node_id
    while current_id is not None:
        node_info = nodes_store[current_id]
        path_coords.append(node_info['pos'])
        current_id = node_info['parent_id']
    return path_coords[::-1]

# ---- Nueva función para reconstruir con detalles ----
def reconstruct_detailed_path(end_node_id, nodes_store):
    path_details = []
    current_id = end_node_id
```

```
    while current_id is not None:
        node_info = nodes_store[current_id]
        path_details.append({
            'pos': node_info['pos'],
            'steps': node_info['steps'],
            'costo_acumulado': node_info['costo_acumulado']
        })
        current_id = node_info['parent_id']
    return path_details[::-1] # De inicio a fin

# ----------------------------
# Algoritmo tipo BFS con Podas y Viento Real
# ----------------------------
def routing_bfs_real_wind(inicio_lat, inicio_lon, fecha_hora_inicio_utc,
profundidad_max_param, polar_data, ax=None):
     # --- MODIFICACIÓN: Calcular punto objetivo y distancia inicial ---
    lat_centro_llegada = (zona_llegada_rect['lat_min'] +
zona_llegada_rect['lat_max']) / 2
    lon_centro_llegada = (zona_llegada_rect['lon_min'] +
zona_llegada_rect['lon_max']) / 2
    distancia_inicial_a_destino = haversine_km(inicio_lat, inicio_lon,
lat_centro_llegada, lon_centro_llegada)
   # Calculamos también la distancia al checkpoint
    distancia_inicial_a_checkpoint = haversine_km(inicio_lat, inicio_lon,
CHECKPOINT_CANAL['lat'], CHECKPOINT_CANAL['lon'])

    print(f"Objetivo Final: Hampton Roads ({lat_centro_llegada:.2f},
{lon_centro_llegada:.2f})")
    print(f"Checkpoint Intermedio: Canal ({CHECKPOINT_CANAL['lat']:.2f},
{CHECKPOINT_CANAL['lon']:.2f})")

    queue = deque()
    nodes_store = {}; next_node_id = 0

    start_node_id = next_node_id; next_node_id += 1
    costo_inicio = 0.0
    nodes_store[start_node_id] = {'pos': (inicio_lat, inicio_lon), 'parent_id':
None,
                                  'costo_acumulado': costo_inicio, 'steps': 0}
    queue.append((costo_inicio, inicio_lat, inicio_lon, start_node_id, 0))

    mejor_tiempo_global = float('inf'); mejor_nodo_final_id = None
    min_steps_to_target_found = float('inf')

    nodos_expandidos_count = 0

    start_time_search = time.time()
    explorados_lons_para_scatter = []
    explorados_lats_para_scatter = []
```

```
    while queue:
        costo_actual, lat, lon, current_node_id, steps_actuales =
queue.popleft()
        nodos_expandidos_count += 1
        current_datetime = fecha_hora_inicio_utc +
timedelta(hours=steps_actuales)


        if steps_actuales >= mejor_tiempo_global:
            continue
        if steps_actuales > min_steps_to_target_found:
            continue
        if profundidad_max_param > 0 and steps_actuales >=
profundidad_max_param:
            continue


        for angulo, dist_segmento_efectiva in obtener_rutas(lat, lon,
current_datetime, polar_data):
            lat_nuevo, lon_nuevo = mover(lat, lon, angulo,
dist_segmento_efectiva)
            costo_nuevo = costo_actual + dist_segmento_efectiva
            nuevos_steps = steps_actuales + 1

            if nuevos_steps > min_steps_to_target_found:
                continue
            if profundidad_max_param > 0 and nuevos_steps >
profundidad_max_param:
                continue
            if not en_corredor_lineal(lat_nuevo, lon_nuevo,
puntos_limite_superior, puntos_limite_inferior):
                continue
            if en_zona_critica(lat_nuevo, lon_nuevo):
                 continue

            new_node_id = next_node_id; next_node_id += 1
            nodes_store[new_node_id] = {'pos': (lat_nuevo, lon_nuevo),
'parent_id': current_node_id,
                                        'costo_acumulado': costo_nuevo,
'steps': nuevos_steps}
            explorados_lons_para_scatter.append(lon_nuevo);
explorados_lats_para_scatter.append(lat_nuevo)

            if en_zona_llegada(lat_nuevo, lon_nuevo):
                if nuevos_steps < mejor_tiempo_global:
                    print(f"   ** Destino alcanzado en {nuevos_steps}h.
Optimizando costo en este nivel de horas. **")
```

```
                    mejor_tiempo_global = nuevos_steps; mejor_nodo_final_id =
new_node_id
                    print(f"     Nueva mejor ruta ({nuevos_steps}h). Costo:
{mejor_tiempo_global:.2f} MN, Nodo: {new_node_id}")
            else:
                if (profundidad_max_param == 0 or nuevos_steps <
profundidad_max_param) and \
                   nuevos_steps < mejor_tiempo_global:
                    queue.append((costo_nuevo, lat_nuevo, lon_nuevo,
new_node_id, nuevos_steps))

        # --- PODA BEAM SEARCH DIVERSIFICADA (Disparador por Nodos Expandidos)
---
        if nodos_expandidos_count > 0 and nodos_expandidos_count %
PODA_BEAM_CADA_N_EXPANDIDOS == 0:

            # --- Parámetros de la Poda Diversificada ---
            N_MEJORES = 1900      # Nº de rutas "élite" a mantener
(Explotación)
            N_ALEATORIOS = 100    # Nº de rutas "comodín" a mantener
(Exploración)
            BEAM_WIDTH = N_MEJORES + N_ALEATORIOS

            # Solo podar si la cola supera el tamaño total deseado
            if len(queue) > BEAM_WIDTH:
                def calcular_promesa(nodo_tupla):
                    lat_actual = nodo_tupla[1]
                    lon_actual = nodo_tupla[2]
                    horas_viaje = nodo_tupla[4]

                    if horas_viaje == 0:
                        return -float('inf')

                    # --- Lógica IF/ELSE para cambiar de objetivo ---
                    if lon_actual >= CHECKPOINT_CANAL['lon']:
                        # Si aún no hemos pasado la longitud del checkpoint,
nuestro objetivo es el checkpoint.
                        distancia_actual_al_objetivo = haversine_km(lat_actual,
lon_actual, CHECKPOINT_CANAL['lat'], CHECKPOINT_CANAL['lon'])
                        progreso_hacia_objetivo =
distancia_inicial_a_checkpoint - distancia_actual_al_objetivo
                    else:
                        # Si ya hemos pasado el checkpoint, nuestro objetivo es
Rotterdam o el siguiente chekpoint.
                        distancia_actual_al_objetivo = haversine_km(lat_actual,
lon_actual, lat_centro_llegada, lon_centro_llegada)
                        progreso_hacia_objetivo = distancia_inicial_a_destino -
distancia_actual_al_objetivo
```

```
                        # La puntuación es la velocidad de acercamiento al
objetivo actual
                    return progreso_hacia_objetivo / horas_viaje

                # Ordenamos toda la cola de mayor a menor puntuación
                sorted_queue_list = sorted(list(queue), key=calcular_promesa,
reverse=True)

                # 1. Nos quedamos con los N_MEJORES nodos (Explotación)
                supervivientes_mejores = sorted_queue_list[:N_MEJORES]

                # 2. Seleccionamos candidatos para la parte aleatoria
                pool_aleatorio = sorted_queue_list[N_MEJORES:N_MEJORES + 1000]

                supervivientes_aleatorios = []
                if pool_aleatorio:
                    num_a_escoger = min(N_ALEATORIOS, len(pool_aleatorio))
                    supervivientes_aleatorios = random.sample(pool_aleatorio,
num_a_escoger)

                # 3. La nueva cola es la combinación de los mejores y los
aleatorios
                queue = deque(supervivientes_mejores +
supervivientes_aleatorios)

                #La siguiente fila se utilizaba para comprobar que el código no
se paraba
                #if n_descartados_beam_actual > 0:
                    #print(f"   PODA BEAM DIVERSIFICADA (N_Exp
{nodos_expandidos_count}): Cola de {num_antes_beam} a {len(queue)}
({n_descartados_beam_actual} desc.)")


        if nodos_expandidos_count % 20000 == 0: # Aumentado para reducir output
            elapsed_search_time = time.time() - start_time_search
            print(f"Nodos exp: {nodos_expandidos_count}, Cola: {len(queue)},
Mejor Costo: {mejor_tiempo_global:.2f}, "
                  f"Min Horas Dest: {min_steps_to_target_found if
min_steps_to_target_found != float('inf') else 'N/A'}, "
                  f"Tiempo Sim: {elapsed_search_time:.2f}s, Fecha actual sim:
{current_datetime.strftime('%Y-%m-%d %H:%M')}")

    end_time_search = time.time()

    print(f"\n--- Estadísticas Búsqueda con Viento Real ---")
    print(f"Tiempo total de búsqueda: {end_time_search - start_time_search:.2f}
segundos")
    print(f"Total nodos expandidos (sacados de cola):
{nodos_expandidos_count}")
```

```
    print(f"Total nodos generados (almacenados en nodes_store):
{len(nodes_store)}")

    # -----Preparar y devolver datos detallados de la ruta ----
    detailed_path_result = None
    if mejor_nodo_final_id is not None:

        steps_de_la_mejor_ruta = nodes_store[mejor_nodo_final_id]['steps']
        detailed_path_result = reconstruct_detailed_path(mejor_nodo_final_id,
nodes_store) # Para la tabla
        return  steps_de_la_mejor_ruta, detailed_path_result
    else:
        global cached_wind_dataset
        if cached_wind_dataset:
            cached_wind_dataset.close(); cached_wind_dataset = None
        return None, None # Devolver None para detailed_path_result

# ----------------------------
# Cargamos diagramas polares
# ----------------------------
print("Iniciando la carga de datos polares...")
datos_polares_cargados = cargar_datos_polares(nombres_de_archivos_csv)
print("¡Carga de datos finalizada!")
# ----------------------------
# Bucle Principal de Simulación por Lotes
# ----------------------------


def ejecutar_simulaciones():

        fecha_inicio_lote = datetime.strptime(FECHA_INICIO_SIMS_STR, "%Y%m%d")
        fecha_fin_lote = datetime.strptime(FECHA_FIN_SIMS_STR,
"%Y%m%d").replace(hour=23)

    current_start_time = fecha_inicio_lote
    current_start_time += timedelta(hours=HORA_INICIO_VIAJE)
    file_exists = os.path.isfile(NOMBRE_ARCHIVO_RESULTADOS)

    with open(NOMBRE_ARCHIVO_RESULTADOS, 'a', newline='', encoding='utf-8') as
f:
        writer = csv.writer(f)
        if not file_exists:
            writer.writerow(["fecha_hora_salida", "tiempo_computacion_seg",
"tiempo_viaje_horas", "fecha_hora_llegada"])

        sim_count = 0
        while current_start_time <= fecha_fin_lote:
            sim_count += 1
```

```
            print(f"\n--- Sim. #{sim_count} | Saliendo:
{current_start_time.strftime('%Y-%m-%d %H:%M')} ---")

            tiempo_inicio_sim = time.time()

            mejor_tiempo, detailed_path = routing_bfs_real_wind(
                      inicio[0], inicio[1], current_start_time,profundidad_max,
datos_polares_cargados
            )

            tiempo_computacion = time.time() - tiempo_inicio_sim

            if mejor_tiempo is not None:
               fecha_llegada = current_start_time + timedelta(hours=mejor_tiempo)
                  resultado = [current_start_time.strftime('%Y-%m-%d %H:%M:%S'),
round(tiempo_computacion, 2), mejor_tiempo, fecha_llegada.strftime('%Y-%m-%d
%H:%M:%S')]
                writer.writerow(resultado)
                   print(f"-> ÉXITO. T. Viaje: {mejor_tiempo} h | T. Cómputo:
{tiempo_computacion:.1f} s")
            else:
                 resultado = [current_start_time.strftime('%Y-%m-%d %H:%M:%S'),
round(tiempo_computacion, 2), "N/A", "N/A", "N/A"]
                writer.writerow(resultado)
                print(f"-> FALLO. No se encontró ruta. ")

            f.flush()
            current_start_time += timedelta(hours=INTERVALO_HORAS_SIMS)

    print(f"\n--- Todas las {sim_count} simulaciones han terminado. Resultados
en '{NOMBRE_ARCHIVO_RESULTADOS}' ---")
    if cached_wind_dataset:
        cached_wind_dataset.close()

ejecutar_simulaciones()
```

Procesamiento de datos de tiempos de viaje

```
import pandas as pd
import matplotlib.pyplot as plt

# Cargar el archivo con nombres correctos
df = pd.read_csv('resultados_simulaciones.csv', parse_dates=['fecha_salida',
'fecha_llegada'])

# Ordenar por fecha de salida
df = df.sort_values('fecha_salida').reset_index(drop=True)

# Crear columna de duración corregida (copiamos la original al principio)
```

```
df['tiempo_viaje_corregido'] = df['horas_viaje']

# Aplicar corrección
for i in range(len(df)):
    salida_i = df.loc[i, 'fecha_salida']
    llegada_i = df.loc[i, 'fecha_llegada']

    posteriores = df[df['fecha_salida'] > salida_i]
    mejores = posteriores[posteriores['fecha_llegada'] < llegada_i]

    if not mejores.empty:
        mejor_llegada = mejores['fecha_llegada'].min()
        nueva_duracion = (mejor_llegada - salida_i).total_seconds() / 3600  # en
horas
        df.loc[i, 'tiempo_viaje_corregido'] = nueva_duracion

# Calcular la nueva fecha de llegada corregida
df['fecha_llegada_corregida']          =          df['fecha_salida']          +
pd.to_timedelta(df['tiempo_viaje_corregido'], unit='h')

# Mostrar gráfica comparativa
plt.figure(figsize=(12, 6))
plt.plot(df['fecha_salida'],    df['horas_viaje'],    'o',    label='Original',
alpha=0.6)
plt.plot(df['fecha_salida'],         df['tiempo_viaje_corregido'],         'o',
label='Corregido', alpha=0.6, color='red')
plt.xlabel('Fecha y hora de salida')
plt.ylabel('Duración del viaje (horas)')
plt.title('Comparación de duración de viajes: Original vs Corregido')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
df.to_csv('resultados_corregidos.csv', index=False)
```

Balance eléctrico buque convencional

			Navegación			Maniobra			C y D			Puerto			Emergencia		
Equipo	**Nº**	**Potencia (kW)**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**
Bomba trasiego HFO wartsila	2	0,5	0,5	0,3	0,15	0,5	0,3	0,15	0,5	0,3	0,15	0,5	0,3	0,15	0,5	0	0
Bomba trasiego MDO wartsila	8	0,18	0,5	0,3	0,216	1	0,3	0,432	1	0,3	0,432	1	0,3	0,432	1	0	0
Bomba de alimentación del separador HFO	1	0,3	1	0,5	0,15	1	0,5	0,15	1	0,5	0,15	1	0,5	0,15	1	0	0
Bomba de alimentación del separador MDO	2	0,3	0,5	0,5	0,15	0,5	0,5	0,15	0,5	0,5	0,15	0,5	0,5	0,15	0,5	0	0
Bomba de circulación de HFO	1	1,5	1	0,5	0,75	1	0,5	0,75	1	0,5	0,75	1	0,5	0,75	1	0	0
Bomba de circulación de MDO	2	1,5	0,5	0,5	0,75	0,5	0,5	0,75	0,5	0,5	0,75	0,5	0,5	0,75	0,5	0	0
Bomba de circulación de la unidad booster	1	1,7	1	0,8	1,36	1	0,8	1,36	1	0,8	1,36	1	0,8	1,36	1	0	0
Bomba de alimentación de la unidad booster	1	0,6	1	0,8	0,48	1	0,8	0,48	1	0,8	0,48	1	0,8	0,48	1	0	0
Precalentador del separador	1	15,3	1	0,8	12,24	1	0,8	12,24	1	0,8	12,24	1	0,8	12,24	1	0	0
Purificadora de HFO	2	3	0,5	0,5	1,5	0,5	0,5	1,5	0,5	0,5	1,5	0,5	0,5	1,5	0,5	0	0
Purificadora de MDO	2	3	0,5	0,5	1,5	0,5	0,5	1,5	0,5	0,5	1,5	0,5	0,5	1,5	0,5	0	
Separador de aceite	2	2	0,5	0,8	1,6	0,5	0,8	1,6	0,5	0	0	0,5	0	0	0,5	0	0
Bomba de alimentación del separador	1	0,2	1	0,8	0,16	1	0,8	0,16	1	0	0	1	0	0	1	0	0
Bomba de aceite lubricante	1	25,6	1	0,8	20,48	1	0,8	20,48	1	0	0	1	0	0	1	0	0
Bomba de aceite de prelubricacion	1	6,4	1	0,8	5,12	1	0,8	5,12	1	0	0	1	0	0	1	0	0
Bomba de trasiego de aceite	1	0,1	1	0,8	0,08	1	0,8	0,08	1	0		1	0	0	1	0	0
Compresor aire de arranque	2	31	0,5	0,3	9,3	0,5	0,5	15,5	0,5	0,3	9,3	0,5	0	0	0,5	0	0
Compresor aire de arranque emergencia grupos auxiliares	1	6,1	1	0,1	0,61	1	0,1	0,61	1	0,1	0,61	1	0	0	1	0	0
Compresor aire de arranque emergencia grupos emergencia	1	3,5	1	0	0	1	0	0	1	0	0	1	0	0	1	0,5	1,75
Compresor servicio y control	1	15	1	0,5	7,5	1	0,5	7,5	1	0,5	7,5	1	0,1	1,5	1	0	0
			Navegación			**Maniobra**			**C y D**			**Puerto**			**Emergencia**		
Equipo	**Nº**	**Potencia (kW)**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**
Bombas AS MP	2	8,1	0,5	1	8,1	0,5	1	8,1	0,5	0	0	0,5	0	0	0,5	0	0
Bombas AD MP	2	7	0,5	1	7	0,5	1	7	0,5	0	0	0,5	0	0	0,5	0	0
Generador de agua dulce	1	10	1	0,8	8	1	0,8	8	1	0	0	1	0	0	1	0	0
Bombas AS Sistema centralizado	2	4,5	0,5	0,8	3,6	0,5	0,8	3,6	0,5	0,8	3,6	0,5	0,8	3,6	0,5	0	0

Bombas AD Sistema centralizado	2	1,7	0,5	0,8	1,36	0,5	0,8	1,36	0,5	0,8	1,36	0,5	0,8	1,36	0,5	0	0
Bomba alimentación agua a la caldera	2	3,4	0,5	0,7	2,38	0,5	0,7	2,38	0,5	0,7	2,38	0,5	0,7	2,38	0,5	0	0
Ventilador CCMM	5	12	0,67	1	40,2	1	1	60	0,67	0,6	24,12	0,33	0,6	11,88	0,67	0	0
Ventilador sala depuradoras	2	2,2	0,5	1	2,2	0,5	1	2,2	0,5	0,6	1,32	0,5	0,6	1,32	0,5	0	0
Bombas principales de sentina	2	10,1	0,5	0,2	2,02	0,5	0,2	2,02	0,5	0,6	6,06	0,5	0,2	2,02	0,5	0,5	5,05
Bomba auxiliar de sentina	1	0,6	1	0,2	0,12	1	0,2	0,12	1	0,6	0,36	1	0,2	0,12	1	0,5	0,3
Separador de sentinas	1	10	1	0,7	7	1	0,7	7	1	0,7	7	1	0,5	5	1	0	0
Incinerador	1	12	1	0,5	6	1	0,5	6	1	0,5	6	1	0,5	6	1	0	0
Bomba de lodos	1	0,5	1	0	0	1	0	0	1	0,8	0,4	1	0,2	0,1	1	0	0
Bomba de trasiego AD	2	0,3	0,5	0,8	0,24	0,5	0,8	0,24	0,5	0,8	0,24	0,5	0,8	0,24	0,5	0	0
Calentador de agua sanitaria	1	12	1	0,8	9,6	1	0,8	9,6	1	0,8	9,6	1	0,8	9,6	1	0	0
Bomba recirculación agua caliente	2	0,1	0,5	0,8	0,08	0,5	0,8	0,08	0,5	0,8	0,08	0,5	0,8	0,08	0,5	0	0
Esterilizador UV	1	0,2	1	1	0,2	1	1	0,2	1	1	0,2	1	1	0,2	1	0	0
			Navegación			**Maniobra**			**C y D**			**Puerto**			**Emergencia**		
Equipo	**Nº**	**Potencia (kW)**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**
Planta tratamiento aguas negras	1	11,5	1	0,5	5,75	1	0,5	5,75	1	0,5	5,75	1	0,4	4,6	1	0	0
Unidad de vacío	1	7	1	0,8	5,6	1	0,8	5,6	1	0,8	5,6	1	0,7	4,9	1	0	0
Bombas de lastre	3	9,6	0,5	0,2	2,88	0,5	0,2	2,88	0,5	0,8	11,52	0,5	0,2	2,88	0,5	0	0
Planta tratamiento agua lastre	1	81	1	0,2	16,2	1	0,2	16,2	1	0,2	16,2	1	0,2	16,2	1	0	0
Sistema anti-incrustaciones	1	1,6	1	1	1,6	1	1	1,6	1	1	1,6	1	1	1,6	1	0	0
Bombas de contra incendios acomodación	2	11	0,5	0,1	1,1	0,5	0,1	1,1	0,5	0,1	1,1	0,5	0,1	1,1	0,5	1	11
Bombas de contra incendios	2	15,4	0,5	0,1	1,54	0,5	0,1	1,54	0,5	0,1	1,54	0,5	0,1	1,54	0,5	1	15,4
Bomba AS para rociadores CCMM	1	0,6	1	0,1	0,06	1	0,1	0,06	1	0,1	0,06	1	0,1	0,06	1	1	0,6
Bomba AS para rociadores acomodación	1	10,4	1	0,1	1,04	1	0,1	1,04	1	0,1	1,04	1	0,1	1,04	1	1	10,4
Ventilador cámara de carne	1	0,2	1	0,8	0,16	1	0,8	0,16	1	0,8	0,16	1	0,8	0,16	1	0	0
Ventilador cámara de pescado	2	0,2	1	0,8	0,32	1	0,8	0,32	1	0,8	0,32	1	0,8	0,32	1	0	0
Ventilador cámara verdura/fruta/lácteos	1	1	1	0,8	0,8	1	0,8	0,8	1	0,8	0,8	1	0,8	0,8	1	0	0
Compresor	2	6	0,5	0,5	3	0,5	0,5	3	0,5	0,5	3	0,5	0,5	3	0,5	0	0
Ventilador impulsión locales	2	3	0,5	0,85	2,55	0,5	0,85	2,55	0,5	0,85	2,55	0,5	0,8	2,4	0,5	0	0

Ventilador extracción locales	2	8	0,5	0,85	6,8	0,5	0,85	6,8	0,5	0,85	6,8	0,5	0,8	6,4	0,5	0	0
			Navegación			**Maniobra**			**C y D**			**Puerto**			**Emergencia**		
Equipo	**Nº**	**Potencia (kW)**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**
Chigres de amarre	2	63	1	0	0	1	0,2	25,2	1	0,2	25,2	1	0,2	25,2	1	0	0
Cabrestantes	9	4	1	0	0	1	0	0	1	0,2	7,2	1	0,2	7,2	1	0	0
Pescante bote salvavidas caída libre	1	11,6	1	0	0	1	0	0	1	0	0	1	0	0	1	1	11,6
Pescante bote rescate	1	11,6	1	0	0	1	0	0	1	0	0	1	0	0	1	1	11,6
Grúa balsas salvavidas y víveres	1	10,5	1	0	0	1	0	0	1	0	0	1	0	0	1	1	10,5
Grúas bodegas de carga	3	120	1	0	0	1	0	0	1	0,8	288	1	0	0	1	0	0
Sistema hidráulico tapas escotilla	2	15	0,5	0	0	0,5	0	0	0,5	0,5	7,5	0,5	0	0	0,5	0	0
Equipos limpieza bodegas	4	5	1	0	0	1	0	0	1	0,5	10	1	0	0	1	0	0
Servomotor	1	58	1	0,4	23,2	1	0,7	40,6	1	0	0	1	0	0	1	0	0
Equipos de cocina	1	50	1	0,3	15	1	0,3	15	1	0,3	15	1	0,3	15	1	0	0
Equipos de lavandería	1	20	1	0,4	8	1	0,4	8	1	0,2	4	1	0,2	4	1	0	0
Equipos de comunicación	1	6	1	0,8	4,8	1	0,8	4,8	1	0,5	3	1	0,2	1,2	1	1	6
Equipos de navegación	1	5	1	0,5	2,5	1	0,5	2,5	1	0	0	1	0	0	1	1	5
Equipos de control	1	5	1	0,5	2,5	1	0,5	2,5	1	0,5	2,5	1	0,2	1	1	1	5
Iluminación exterior	1	60	1	0,6	36	1	0,5	30	1	0,5	30	1	0,5	30	1	0	0
Iluminación interior	1	110	1	0,8	88	1	0,7	77	1	0,8	88	1	0,8	88	1	0	0
Luces de navegación	1	2	1	0,5	1	1	0,5	1	1	0	0	1	0	0	1	0	0
Equipos de taller y herramientas	1	30	1	0,2	6	1	0,2	6	1	0,2	6	1	0,2	6	1	0	0
Grúa desmontaje CCMM	1	10	1	0,2	2	1	0,2	2	1	0,2	2	1	0,2	2	1	0	0
TOTAL (kW)					**400,5**			**452,4**			**646,0**			**291,4**			**94,2**

Balance eléctrico buque a vela

			Navegación			Maniobra			C y D			Puerto			Emergencia		
Equipo	Nº	Potencia (kW)	Kn	Ksr	Pot,	Kn	Ksr	Pot,	Kn	Ksr	Pot,	Kn	Ksr	Pot,	Kn	Ksr	Pot,
Bomba trasiego MDO wartsila	8	0,18	0,5	0,3	0,216	1	0,3	0,432	1	0,3	0,432	1	0,3	0,432	1	0	0
Bomba de circulación de MDO	2	4	0,5	0,5	2	0,5	0,5	2	0,5	0,5	2	0,5	0,5	2	0,5	0	0
Purificadora de MDO	2	3	0,5	0,5	1,5	0,5	0,5	1,5	0,5	0,5	1,5	0,5	0,5	1,5	0,5	0	
Compresor aire de arranque	2	31	0,5	0,3	9,3	0,5	0,5	15,5	0,5	0,3	9,3	0,5	0	0	0,5	0	0
Compresor aire de arranque emergencia grupos auxiliares	1	6,1	1	0,1	0,61	1	0,1	0,61	1	0,1	0,61	1	0	0	1	0	0
Compresor aire de arranque emergencia grupos emergencia	1	3,5	1	0	0	1	0	0	1	0	0	1	0	0	1	0,5	1,75
Compresor servicio y control	1	15	1	0,5	7,5	1	0,5	7,5	1	0,5	7,5	1	0,1	1,5	1	0	0
			Navegación			**Maniobra**			**C y D**			**Puerto**			**Emergencia**		
Equipo	Nº	Potencia (kW)	Kn	Ksr	Pot,	Kn	Ksr	Pot,	Kn	Ksr	Pot,	Kn	Ksr	Pot,	Kn	Ksr	Pot,
Bombas AS Sistema centralizado	2	4,6	0,5	0,8	3,68	0,5	0,8	3,68	0,5	0,8	3,68	0,5	0,8	3,68	0,5	0	0
Bombas AD Sistema centralizado	2	1,8	0,5	0,8	1,44	0,5	0,8	1,44	0,5	0,8	1,44	0,5	0,8	1,44	0,5	0	0
Ventilador CCMM	3	11	0,67	1	22,11	1	1	33	0,67	0,6	13,266	0,33	0,6	6,534	0,67	0	0
Bombas principales de sentina	2	10,1	0,5	0,2	2,02	0,5	0,2	2,02	0,5	0,6	6,06	0,5	0,2	2,02	0,5	0,5	5,05
Bomba auxiliar de sentina	1	0,6	1	0,2	0,12	1	0,2	0,12	1	0,6	0,36	1	0,2	0,12	1	0,5	0,3
Separador de sentinas	1	10	1	0,7	7	1	0,7	7	1	0,7	7	1	0,5	5	1	0	0
Incinerador	1	12	1	0,5	6	1	0,5	6	1	0,5	6	1	0,5	6	1	0	0
Bomba de lodos	1	0,5	1	0	0	1	0	0	1	0,8	0,4	1	0,2	0,1	1	0	0
Bomba de trasiego AD	2	0,3	0,5	0,8	0,24	0,5	0,8	0,24	0,5	0,8	0,24	0,5	0,8	0,24	0,5	0	0
Calentador de agua sanitaria	1	12	1	0,8	9,6	1	0,8	9,6	1	0,8	9,6	1	0,8	9,6	1	0	0
Bomba recirculación agua caliente	2	0,1	0,5	0,8	0,08	0,5	0,8	0,08	0,5	0,8	0,08	0,5	0,8	0,08	0,5	0	0
Esterilizador UV	1	0,2	1	1	0,2	1	1	0,2	1	1	0,2	1	1	0,2	1	0	0
			Navegación			**Maniobra**			**C y D**			**Puerto**			**Emergencia**		
Equipo	**Nº**	**Potencia (kW)**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**
Planta tratamiento aguas negras	1	11,5	1	0,5	5,75	1	0,5	5,75	1	0,5	5,75	1	0,4	4,6	1	0	0
Unidad de vacío	1	7	1	0,8	5,6	1	0,8	5,6	1	0,8	5,6	1	0,7	4,9	1	0	0
Bombas de lastre	3	9,6	0,5	0,2	2,88	0,5	0,2	2,88	0,5	0,8	11,52	0,5	0,2	2,88	0,5	0	0

Planta tratamiento agua lastre	1	81	1	0,2	16,2	1	0,2	16,2	1	0,2	16,2	1	0,2	16,2	1	0	0
Sistema anti-incrustaciones	1	1,6	1	1	1,6	1	1	1,6	1	1	1,6	1	1	1,6	1	0	0
Bombas de contra incendios acomodación	2	11	0,5	0,1	1,1	0,5	0,1	1,1	0,5	0,1	1,1	0,5	0,1	1,1	0,5	1	11
Bombas de contra incendios	2	19,2	0,5	0,1	1,92	0,5	0,1	1,92	0,5	0,1	1,92	0,5	0,1	1,92	0,5	1	19,2
Bomba AS para rociadores CCMM	1	0,5	1	0,1	0,05	1	0,1	0,05	1	0,1	0,05	1	0,1	0,05	1	1	0,5
Bomba AS para rociadores acomodación	1	10,4	1	0,1	1,04	1	0,1	1,04	1	0,1	1,04	1	0,1	1,04	1	1	10,4
Ventilador cámara de carne	1	0,2	1	0,8	0,16	1	0,8	0,16	1	0,8	0,16	1	0,8	0,16	1	0	0
Ventilador cámara de pescado	2	0,2	1	0,8	0,32	1	0,8	0,32	1	0,8	0,32	1	0,8	0,32	1	0	0
Ventilador cámara verdura/fruta/lácteos	1	1	1	0,8	0,8	1	0,8	0,8	1	0,8	0,8	1	0,8	0,8	1	0	0
Compresor	2	6	0,5	0,5	3	0,5	0,5	3	0,5	0,5	3	0,5	0,5	3	0,5	0	0
Ventilador impulsión locales	2	3	0,5	0,85	2,55	0,5	0,85	2,55	0,5	0,85	2,55	0,5	0,8	2,4	0,5	0	0
Ventilador extracción locales	2	8	0,5	0,85	6,8	0,5	0,85	6,8	0,5	0,85	6,8	0,5	0,8	6,4	0,5	0	0
			Navegación			**Maniobra**			**C y D**			**Puerto**			**Emergencia**		
Equipo	**Nº**	**Potencia (kW)**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**	**Kn**	**Ksr**	**Pot,**
Chigres de amarre	2	63	1	0	0	1	0,2	25,2	1	0,2	25,2	1	0,2	25,2	1	0	0
Cabrestantes	9	4	1	0	0	1	0	0	1	0,2	7,2	1	0,2	7,2	1	0	0
Pescante bote salvavidas caída libre	1	11,6	1	0	0	1	0	0	1	0	0	1	0	0	1	1	11,6
Pescante bote rescate	1	11,6	1	0	0	1	0	0	1	0	0	1	0	0	1	1	11,6
Grúa balsas salvavidas y víveres	1	10,5	1	0	0	1	0	0	1	0	0	1	0	0	1	1	10,5
Sistema hidráulico tapas escotilla	2	15	0,5	0	0	0,5	0	0	0,5	0,5	7,5	0,5	0	0	0,5	0	0
Equipos limpieza bodegas	4	5	1	0	0	1	0	0	1	0,5	10	1	0	0	1	0	0
Servomotor	1	58	1	0,4	23,2	1	0,7	40,6	1	0	0	1	0	0	1	0	0
Equipos de cocina	1	50	1	0,3	15	1	0,3	15	1	0,3	15	1	0,3	15	1	0	0
Equipos de lavandería	1	20	1	0,4	8	1	0,4	8	1	0,2	4	1	0,2	4	1	0	0
Equipos de comunicación	1	6	1	0,8	4,8	1	0,8	4,8	1	0,5	3	1	0,2	1,2	1	1	6
Equipos de navegación	1	5	1	0,5	2,5	1	0,5	2,5	1	0	0	1	0	0	1	1	5
Equipos de control	1	5	1	0,5	2,5	1	0,5	2,5	1	0,5	2,5	1	0,2	1	1	1	5
Iluminación exterior	1	60	1	0,6	36	1	0,5	30	1	0,6	36	1	0,6	36	1	0	0
Iluminación interior	1	110	1	0,8	88	1	0,7	77	1	0,8	88	1	0,9	99	1	0	0

Luces de navegación	1	2	1	0,5	1	1	0,5	1	1	0	0	1	0	0	1	0	0
Equipos de taller y herramientas	1	30	1	0,2	6	1	0,2	6	1	0,2	6	1	0,2	6	1	0	0
Grúa desmontaje CCMM	1	10	1	0,2	2	1	0,2	2	1	0,2	2	1	0,2	2	1	0	0
TOTAL					**312,4**			**355,3**			**334,5**			**284,4**			**97,9**

Algoritmo de análisis económico

Algoritmo 1

El siguiente algoritmo es el que nos permite realizar el análisis económico y genera las gráficas de histogramas y análisis de sensibilidad

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import numpy_financial as npf

# --- 1. PARÁMETROS DE LA SIMULACIÓN ---
N_SIMULACIONES = 16000
VIDA_UTIL_PROYECTO = 35
N_BUQUES_CONVENCIONALES = 2


df_ida = pd.read_csv('resultados_corregidos_ida.csv')
tiempos_vela_ida_dias = pd.to_numeric(df_ida['tiempo_viaje_corregido'],
errors='coerce').dropna() / 24
df_vuelta = pd.read_csv('resultados_corregidos_vuelta.csv')
tiempos_vela_vuelta_dias = pd.to_numeric(df_vuelta['tiempo_viaje_corregido'],
errors='coerce').dropna() / 24


# --- Parámetros Económicos y Financieros ---
CAPEX_POR_BUQUE_CONVENCIONAL = 16313469
CAPEX_POR_BUQUE_VELA = 16394522
OPEX_ANUAL_POR_BUQUE_CONV = 3816
OPEX_ANUAL_POR_BUQUE_VELA = 16816
TASAS_POR_VIAJE_CONVENCIONAL = 49850
TASAS_POR_VIAJE_VELA = 49303
CAPACIDAD_CARGA_TONS = 5000
DIAS_OPERACION_ANUAL = 365 - 15
TIEMPO_NAVEGACION_CONV_DIAS = 218 / 24
TIEMPOS_POR_CICLO_DIAS = {'maniobra': 1.0, 'carga_descarga': 2.0, 'puerto':
4.0}
DENSIDAD_COMBUSTIBLE = 0.85
CONSUMOS_CONV = {'navegacion': 23.65*DENSIDAD_COMBUSTIBLE, 'maniobra':
2.88*DENSIDAD_COMBUSTIBLE, 'carga_descarga': 4.11*DENSIDAD_COMBUSTIBLE,
'puerto': 1.95*DENSIDAD_COMBUSTIBLE}
CONSUMOS_VELA = {'navegacion': 2.53*DENSIDAD_COMBUSTIBLE, 'maniobra':
2.88*DENSIDAD_COMBUSTIBLE, 'carga_descarga': 2.70*DENSIDAD_COMBUSTIBLE,
'puerto': 2.20*DENSIDAD_COMBUSTIBLE}
PRECIO_ETS_POR_TONELADA_CO2 = 70
FACTOR_EMISION_CO2 = 3.15
# Coste anual adicional por la gestión y administración de cada buque en la
flota
COSTE_GESTION_ANUAL_POR_BUQUE = 15000 # (€/año por buque)
```

```
# FUELEU Penalización ANUAL por TONELADA de combustible (€/tonelada)
penalizaciones_hfo = (
    [62.21] * 5 +       # 2025-2029
    [156.43] * 5 +      # 2030-2034
    [355.66] * 5 +      # 2035-2039
    [745.34] * 5 +      # 2040-2044
    [1475.59] * 5 +     # 2045-2049
    [1899.6] * 10       # 2050 en adelante (hasta completar 35 años)
)

penalizaciones_lfo = (
    [53.98] * 5 +       # 2025-2029
    [149.74] * 5 +      # 2030-2034
    [353.22] * 5 +      # 2035-2039
    [748.21] * 5 +      # 2040-2044
    [1490.32] * 5 +     # 2045-2049
    [1921.22] * 10      # 2050 en adelante (hasta completar 35 años)
)
PENALIZACION_FUELEU_HFO = np.array(penalizaciones_hfo)
PENALIZACION_FUELEU_LFO = np.array(penalizaciones_lfo)
PROPORCION_HFO = 107 / 218
PROPORCION_LFO = 111 / 218

# --- Variables libres ---
PRECIO_COMB_MIN = 350
PRECIO_COMB_PROBABLE = 500
PRECIO_COMB_MAX = 900
TASA_DESCUENTO_MIN = 0.06 # Tasa REAL
TASA_DESCUENTO_MAX = 0.13 # Tasa REAL

# --- Parámetros financieros ---
PORCENTAJE_FINANCIADO = 0.75
TASA_INTERES_PRESTAMO = 0.06
ANIOS_FINANCIACION = 10

# --- CÁLCULO DEL OBJETIVO ANUAL DE CARGA ---
tiempo_ciclo_conv = (TIEMPO_NAVEGACION_CONV_DIAS * 2) +
sum(TIEMPOS_POR_CICLO_DIAS.values())
viajes_por_barco_conv_anual = DIAS_OPERACION_ANUAL / tiempo_ciclo_conv
CARGA_ANUAL_OBJETIVO = N_BUQUES_CONVENCIONALES * viajes_por_barco_conv_anual *
CAPACIDAD_CARGA_TONS
print(f"La flota de {N_BUQUES_CONVENCIONALES} buques convencionales transporta
un objetivo de {CARGA_ANUAL_OBJETIVO:,.0f} toneladas al año.")

# --- FUNCIONES DEL MODELO ---
def calcular_flujos_financieros(coste_total_flota, vida_util):
    fondos_propios = coste_total_flota * (1 - PORCENTAJE_FINANCIADO)
    deuda_total = coste_total_flota * PORCENTAJE_FINANCIADO
```

```
    pago_principal_anual = deuda_total / ANIOS_FINANCIACION if
ANIOS_FINANCIACION > 0 else 0
    flujos_financieros = []
    deuda_pendiente = deuda_total
    for anio in range(1, vida_util + 1):
        pago_intereses = 0; pago_principal = 0
        if anio <= ANIOS_FINANCIACION:
            pago_intereses = deuda_pendiente * TASA_INTERES_PRESTAMO
            pago_principal = pago_principal_anual
            deuda_pendiente -= pago_principal
        flujos_financieros.append(pago_principal + pago_intereses)
    return fondos_propios, flujos_financieros

def simular_costes_y_carga_anual(buque, n_buques, precio_combustible,
tiempo_viaje_ida_vela=None, tiempo_viaje_vuelta_vela=None):
    if buque == 'vela':
        opex_anual_base = OPEX_ANUAL_POR_BUQUE_VELA
        consumos = CONSUMOS_VELA
        tasas_viaje = TASAS_POR_VIAJE_VELA
        tiempo_navegacion = tiempo_viaje_ida_vela + tiempo_viaje_vuelta_vela
    else:
        opex_anual_base = OPEX_ANUAL_POR_BUQUE_CONV
        consumos = CONSUMOS_CONV
        tasas_viaje = TASAS_POR_VIAJE_CONVENCIONAL
        tiempo_navegacion = TIEMPO_NAVEGACION_CONV_DIAS * 2

    consumo_comb_ciclo = (tiempo_navegacion * consumos['navegacion'] +
TIEMPOS_POR_CICLO_DIAS['maniobra'] * consumos['maniobra'] +
TIEMPOS_POR_CICLO_DIAS['carga_descarga'] * consumos['carga_descarga'] +
TIEMPOS_POR_CICLO_DIAS['puerto'] * consumos['puerto'])
    coste_variable_ciclo = (consumo_comb_ciclo * precio_combustible) +
tasas_viaje
    coste_gestion_anual = n_buques * COSTE_GESTION_ANUAL_POR_BUQUE
    tiempo_ciclo_completo = tiempo_navegacion +
sum(TIEMPOS_POR_CICLO_DIAS.values())
    num_ciclos_anual = DIAS_OPERACION_ANUAL / tiempo_ciclo_completo if
tiempo_ciclo_completo > 0 else 0

    coste_ets_anual = 0
    consumo_total_propulsion_flota = 0

    if buque == 'convencional':
        consumo_propulsion_anual_por_barco = (num_ciclos_anual *
tiempo_navegacion) * 21.26 #Consumo para la propulsión
        consumo_total_propulsion_flota =
n_buques*consumo_propulsion_anual_por_barco
        co2_emitido_anual_por_barco = consumo_propulsion_anual_por_barco *
FACTOR_EMISION_CO2
```

```
        coste_ets_anual = n_buques * co2_emitido_anual_por_barco *
PRECIO_ETS_POR_TONELADA_CO2

    carga_anual_flota = n_buques * num_ciclos_anual * CAPACIDAD_CARGA_TONS
    coste_operativo_anual_flota = (n_buques * (opex_anual_base +
(num_ciclos_anual * coste_variable_ciclo))) + coste_ets_anual +
coste_gestion_anual

    return coste_operativo_anual_flota, carga_anual_flota,
consumo_total_propulsion_flota

def calcular_FR(capex_inicial, costes_anuales, carga_anual, tasa_descuento):
    van_costes = capex_inicial + npf.npv(tasa_descuento, costes_anuales)
    van_carga = npf.npv(tasa_descuento, carga_anual)
    if van_carga <= 0: return float('inf')
    return van_costes / van_carga

# --- 4. BUCLE DE SIMULACIÓN DE MONTECARLO ---
print("Iniciando simulación de Montecarlo...")
resultados_simulacion = []
for _ in range(N_SIMULACIONES):
    precio_comb_actual = np.random.triangular(PRECIO_COMB_MIN,
PRECIO_COMB_PROBABLE, PRECIO_COMB_MAX)
    tasa_descuento_actual = np.random.uniform(TASA_DESCUENTO_MIN,
TASA_DESCUENTO_MAX)
    tiempo_ida_actual = np.random.choice(tiempos_vela_ida_dias)
    tiempo_vuelta_actual = np.random.choice(tiempos_vela_vuelta_dias)

    # --- Flota convencional ---
    costes_op_anuales_conv, carga_anual_conv, consumo_anual_propulsion_conv =
simular_costes_y_carga_anual('convencional', N_BUQUES_CONVENCIONALES,
precio_comb_actual)
    # --- FuelEU ---
    consumo_anual_hfo = consumo_anual_propulsion_conv * PROPORCION_HFO
    consumo_anual_lfo = consumo_anual_propulsion_conv * PROPORCION_LFO
    costes_fueleu_anual = (consumo_anual_hfo *
PENALIZACION_FUELEU_HFO[:VIDA_UTIL_PROYECTO] +
                           consumo_anual_lfo *
PENALIZACION_FUELEU_LFO[:VIDA_UTIL_PROYECTO]) * 0.5
    costes_op_totales_conv_anual = costes_op_anuales_conv + costes_fueleu_anual

    capex_inicial_conv, flujos_fin_conv =
calcular_flujos_financieros(CAPEX_POR_BUQUE_CONVENCIONAL *
N_BUQUES_CONVENCIONALES, VIDA_UTIL_PROYECTO)
    costes_totales_anuales_conv = costes_op_totales_conv_anual+ flujos_fin_conv
    FR_conv = calcular_FR(capex_inicial_conv, costes_totales_anuales_conv,
[carga_anual_conv] * VIDA_UTIL_PROYECTO, tasa_descuento_actual)

    # --- Flota a vela ---
```

```
    tiempo_ciclo_vela = tiempo_ida_actual + tiempo_vuelta_actual +
sum(TIEMPOS_POR_CICLO_DIAS.values())
    carga_por_barco_vela_anual = DIAS_OPERACION_ANUAL / tiempo_ciclo_vela *
CAPACIDAD_CARGA_TONS if tiempo_ciclo_vela > 0 else 0
    n_buques_vela_necesarios = np.ceil(CARGA_ANUAL_OBJETIVO /
carga_por_barco_vela_anual) if carga_por_barco_vela_anual > 0 else float('inf')
    costes_op_anuales_vela, carga_anual_vela,_ =
simular_costes_y_carga_anual('vela', n_buques_vela_necesarios,
precio_comb_actual, tiempo_ida_actual, tiempo_vuelta_actual)
    capex_inicial_vela, flujos_fin_vela =
calcular_flujos_financieros(CAPEX_POR_BUQUE_VELA * n_buques_vela_necesarios,
VIDA_UTIL_PROYECTO)
    costes_totales_anuales_vela = [costes_op_anuales_vela + fin for fin in
flujos_fin_vela]
    FR_vela = calcular_FR(capex_inicial_vela, costes_totales_anuales_vela,
[carga_anual_vela] * VIDA_UTIL_PROYECTO, tasa_descuento_actual)


    resultados_simulacion.append({
        'FR_conv': FR_conv,
        'FR_vela': FR_vela,
        'n_buques_vela': n_buques_vela_necesarios,
        'precio_combustible': precio_comb_actual,
        'tasa_descuento': tasa_descuento_actual,
        'tiempo_ciclo_vela': tiempo_ciclo_vela
    })
print("Simulación completada.")

# --- ANÁLISIS Y VISUALIZACIÓN DE RESULTADOS ---
df_resultados = pd.DataFrame(resultados_simulacion)
prob_vela_mas_barato = np.mean(df_resultados['FR_vela'] <
df_resultados['FR_conv']) * 100

print(f"Probabilidad de que la flota a vela sea más económica:
{prob_vela_mas_barato:.2f}%")

# Gráfico 1: Histograma de FR
plt.figure(figsize=(12, 7))
sns.histplot(df_resultados['FR_conv'], color="blue", label=f"Flota Convencional
(Media: {df_resultados['FR_conv'].mean():.2f}) €", kde=True, bins=50)
sns.histplot(df_resultados['FR_vela'], color="green", label=f"Flota a Vela
(Media: {df_resultados['FR_vela'].mean():.2f}) €", kde=True, bins=50,
alpha=0.7)
plt.title('Distribución del flete requerido (FR) por flota', fontsize=16)
plt.xlabel('FR (€ por tonelada)', fontsize=12)
plt.ylabel('Frecuencia', fontsize=12)
plt.legend()
plt.grid(True)
plt.show()
```

```
# Gráfico 2: Histograma del Número de Buques a Vela Necesarios
plt.figure(figsize=(12, 7))
sns.histplot(df_resultados['n_buques_vela'], discrete=True, kde=False,
stat="probability")
plt.title(f'Distribución del tamaño de la flota a vela necesaria', fontsize=16)
plt.xlabel('Número de buques a vela', fontsize=12)
plt.ylabel('Probabilidad', fontsize=12)
plt.grid(True)
plt.xticks(range(int(df_resultados['n_buques_vela'].min()),
int(df_resultados['n_buques_vela'].max()) + 1))
plt.show()

# Gráfico 3: Análisis de Sensibilidad, precio de combustible y tasa de
descuento
df_resultados['vela_es_mejor'] = df_resultados['FR_vela'] <
df_resultados['FR_conv']

plt.figure(figsize=(12, 8))
sns.scatterplot(
    data=df_resultados,
    x='precio_combustible',
    y='tasa_descuento',
    hue='vela_es_mejor',
    palette={True: 'green', False: 'red'},
    alpha=0.5,
    s=20 # Tamaño de los puntos
)

plt.title('Mapa de rentabilidad: precio del combustible vs. tasa de descuento',
fontsize=16)
plt.xlabel('Precio del Combustible ($/tonelada)', fontsize=12)
plt.ylabel('Tasa de Descuento', fontsize=12)
plt.gca().yaxis.set_major_formatter(plt.FuncFormatter('{:.1%}'.format)) #
plt.grid(True)
plt.show()

# Gráfica 4: Análisis de sensibilidad tipo tornado
df_resultados['FR_diff'] = df_resultados['FR_vela'] - df_resultados['FR_conv']

# Seleccionamos las variables de entrada y el resultado
df_sensibilidad = df_resultados[['precio_combustible', 'tasa_descuento',
'tiempo_ciclo_vela', 'FR_diff']].copy()
df_sensibilidad.rename(columns={
    'precio_combustible': 'Precio Combustible',
    'tasa_descuento': 'Tasa de Descuento',
    'tiempo_ciclo_vela': 'Tiempo de Viaje (Vela)'
}, inplace=True)
```

```
# Calcular la correlación
corr = df_sensibilidad.corr(method='spearman')['FR_diff'].dropna()
# Quitar la correlación consigo misma y ordenar por valor absoluto
corr = corr.drop('FR_diff').sort_values(key=abs, ascending=False)
print(corr)

# Gráfico de Tornado (Barra horizontal)
plt.figure(figsize=(10, 6))
corr.plot(kind='barh', color=(corr > 0).map({True: 'coral', False: 'skyblue'}))
plt.title('Análisis de Sensibilidad (Gráfico de Tornado)', fontsize=16)
plt.xlabel('Coeficiente de Correlación de Spearman con la Rentabilidad de la
Flota a Vela', fontsize=12)
plt.grid(axis='x', linestyle='--')
plt.show()
```

Algoritmo 2

El siguiente algoritmo permite representar la rentabilidad de diferentes flotas a vela a lo largo de los años.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import numpy_financial as npf

# --- 1. PARÁMETROS DE LA SIMULACIÓN ---
N_SIMULACIONES = 16000
VIDA_UTIL_ANIOS_LISTA = [5, 10, 15, 20, 25, 30, 35]
N_BUQUES_CONVENCIONALES_LISTA = [1, 2, 4, 6, 8]
# --- Cargar datos de tiempos de viaje ---

df_ida = pd.read_csv('resultados_corregidos_ida.csv')
tiempos_vela_ida_dias = pd.to_numeric(df_ida['tiempo_viaje_corregido'],
errors='coerce').dropna() / 24
df_vuelta = pd.read_csv('resultados_corregidos_vuelta.csv')
tiempos_vela_vuelta_dias = pd.to_numeric(df_vuelta['tiempo_viaje_corregido'],
errors='coerce').dropna() / 24


# --- Parámetros económicos y financieros ---
CAPEX_POR_BUQUE_CONVENCIONAL = 16313469
CAPEX_POR_BUQUE_VELA = 16394522
OPEX_ANUAL_POR_BUQUE_CONV = 3816
OPEX_ANUAL_POR_BUQUE_VELA = 16816
TASAS_POR_VIAJE_CONVENCIONAL = 49850
TASAS_POR_VIAJE_VELA = 49303
CAPACIDAD_CARGA_TONS = 5000
DIAS_OPERACION_ANUAL = 365 - 15
TIEMPO_NAVEGACION_CONV_DIAS = 218 / 24
TIEMPOS_POR_CICLO_DIAS = {'maniobra': 1.0, 'carga_descarga': 2.0, 'puerto':
4.0}
DENSIDAD_COMBUSTIBLE = 0.85
CONSUMOS_CONV = {'navegacion': 23.65*DENSIDAD_COMBUSTIBLE, 'maniobra':
2.88*DENSIDAD_COMBUSTIBLE, 'carga_descarga': 4.11*DENSIDAD_COMBUSTIBLE,
'puerto': 1.95*DENSIDAD_COMBUSTIBLE}
CONSUMOS_VELA = {'navegacion': 2.53*DENSIDAD_COMBUSTIBLE, 'maniobra':
2.88*DENSIDAD_COMBUSTIBLE, 'carga_descarga': 2.70*DENSIDAD_COMBUSTIBLE,
'puerto': 2.20*DENSIDAD_COMBUSTIBLE}
PRECIO_ETS_POR_TONELADA_CO2 = 70
FACTOR_EMISION_CO2 = 3.15
# Coste anual adicional por la gestión y administración de cada buque en la
flota
COSTE_GESTION_ANUAL_POR_BUQUE = 15000 # (€/año por buque)
# FUELEU Penalización ANUAL por TONELADA de combustible (€/tonelada)
penalizaciones_hfo = (
```

```
    [62.21] * 5 +       # 2025-2029
    [156.43] * 5 +      # 2030-2034
    [355.66] * 5 +      # 2035-2039
    [745.34] * 5 +      # 2040-2044
    [1475.59] * 5 +     # 2045-2049
    [1899.6] * 10       # 2050 en adelante (hasta completar 35 años)
)

penalizaciones_lfo = (
    [53.98] * 5 +       # 2025-2029
    [149.74] * 5 +      # 2030-2034
    [353.22] * 5 +      # 2035-2039
    [748.21] * 5 +      # 2040-2044
    [1490.32] * 5 +     # 2045-2049
    [1921.22] * 10      # 2050 en adelante (hasta completar 35 años)
)
PENALIZACION_FUELEU_HFO = np.array(penalizaciones_hfo)
PENALIZACION_FUELEU_LFO = np.array(penalizaciones_lfo)
PROPORCION_HFO = 107 / 218
PROPORCION_LFO = 111 / 218

# --- Variables libres ---
PRECIO_COMB_MIN = 350
PRECIO_COMB_PROBABLE = 500
PRECIO_COMB_MAX = 900
TASA_DESCUENTO_MIN = 0.06 # Tasa REAL
TASA_DESCUENTO_MAX = 0.13 # Tasa REAL

# --- Parámetros financieros ---
PORCENTAJE_FINANCIADO = 0.75
TASA_INTERES_PRESTAMO = 0.06
ANIOS_FINANCIACION = 10



# --- FUNCIONES DEL MODELO ---
def calcular_flujos_financieros(coste_total_flota, vida_util):
    fondos_propios = coste_total_flota * (1 - PORCENTAJE_FINANCIADO)
    deuda_total = coste_total_flota * PORCENTAJE_FINANCIADO
    pago_principal_anual = deuda_total / ANIOS_FINANCIACION if
ANIOS_FINANCIACION > 0 else 0
    flujos_financieros = []
    deuda_pendiente = deuda_total
    for anio in range(1, vida_util + 1):
        pago_intereses = 0; pago_principal = 0
        if anio <= ANIOS_FINANCIACION:
            pago_intereses = deuda_pendiente * TASA_INTERES_PRESTAMO
            pago_principal = pago_principal_anual
            deuda_pendiente -= pago_principal
```

```
        flujos_financieros.append(pago_principal + pago_intereses)
    return fondos_propios, flujos_financieros

def simular_costes_y_carga_anual(buque, n_buques, precio_combustible,
tiempo_viaje_ida_vela=None, tiempo_viaje_vuelta_vela=None):
    if buque == 'vela':
        opex_anual_base = OPEX_ANUAL_POR_BUQUE_VELA
        consumos = CONSUMOS_VELA
        tasas_viaje = TASAS_POR_VIAJE_VELA
        tiempo_navegacion = tiempo_viaje_ida_vela + tiempo_viaje_vuelta_vela
    else:
        opex_anual_base = OPEX_ANUAL_POR_BUQUE_CONV
        consumos = CONSUMOS_CONV
        tasas_viaje = TASAS_POR_VIAJE_CONVENCIONAL
        tiempo_navegacion = TIEMPO_NAVEGACION_CONV_DIAS * 2

    consumo_comb_ciclo = (tiempo_navegacion * consumos['navegacion'] +
TIEMPOS_POR_CICLO_DIAS['maniobra'] * consumos['maniobra'] +
TIEMPOS_POR_CICLO_DIAS['carga_descarga'] * consumos['carga_descarga'] +
TIEMPOS_POR_CICLO_DIAS['puerto'] * consumos['puerto'])
    coste_variable_ciclo = (consumo_comb_ciclo * precio_combustible) +
tasas_viaje
    coste_gestion_anual = n_buques * COSTE_GESTION_ANUAL_POR_BUQUE
    tiempo_ciclo_completo = tiempo_navegacion +
sum(TIEMPOS_POR_CICLO_DIAS.values())
    num_ciclos_anual = DIAS_OPERACION_ANUAL / tiempo_ciclo_completo if
tiempo_ciclo_completo > 0 else 0

    coste_ets_anual = 0
    consumo_total_propulsion_flota = 0

    if buque == 'convencional':
        consumo_propulsion_anual_por_barco = (num_ciclos_anual *
tiempo_navegacion) * 21.26 #Consumo para la propulsión
        consumo_total_propulsion_flota =
n_buques*consumo_propulsion_anual_por_barco
        co2_emitido_anual_por_barco = consumo_propulsion_anual_por_barco *
FACTOR_EMISION_CO2
        coste_ets_anual = n_buques * co2_emitido_anual_por_barco *
PRECIO_ETS_POR_TONELADA_CO2

    carga_anual_flota = n_buques * num_ciclos_anual * CAPACIDAD_CARGA_TONS
    coste_operativo_anual_flota = (n_buques * (opex_anual_base +
(num_ciclos_anual * coste_variable_ciclo))) + coste_ets_anual +
coste_gestion_anual

    return coste_operativo_anual_flota, carga_anual_flota,
consumo_total_propulsion_flota
```

```
def calcular_FR(capex_inicial, costes_anuales, carga_anual, tasa_descuento):
    van_costes = capex_inicial + npf.npv(tasa_descuento, costes_anuales)
    van_carga = npf.npv(tasa_descuento, carga_anual)
    if van_carga <= 0: return float('inf')
    return van_costes / van_carga

# --- 4. BUCLE DE SIMULACIÓN DE MONTECARLO ---
print("Iniciando análisis de sensibilidad completo...")
resultados_finales = []

for n_buques_conv in N_BUQUES_CONVENCIONALES_LISTA:
    for vida_util_proyecto in VIDA_UTIL_ANIOS_LISTA:
        tiempo_ciclo_conv = (TIEMPO_NAVEGACION_CONV_DIAS * 2) +
sum(TIEMPOS_POR_CICLO_DIAS.values())
        viajes_por_barco_conv_anual = DIAS_OPERACION_ANUAL / tiempo_ciclo_conv
        CARGA_ANUAL_OBJETIVO = n_buques_conv * viajes_por_barco_conv_anual *
CAPACIDAD_CARGA_TONS

        FRs_conv_escenario = []
        FRs_vela_escenario = []

        for _ in range(N_SIMULACIONES):
            precio_comb_actual = np.random.triangular(PRECIO_COMB_MIN,
PRECIO_COMB_PROBABLE, PRECIO_COMB_MAX)
            tasa_descuento_actual = np.random.uniform(TASA_DESCUENTO_MIN,
TASA_DESCUENTO_MAX)
            tiempo_ida_actual = np.random.choice(tiempos_vela_ida_dias)
            tiempo_vuelta_actual = np.random.choice(tiempos_vela_vuelta_dias)

            # --- Flota Convencional ---
            costes_op_anuales_conv, carga_anual_conv,
consumo_anual_propulsion_conv = simular_costes_y_carga_anual('convencional',
n_buques_conv, precio_comb_actual)
            # --- FuelEU ---
            consumo_anual_hfo = consumo_anual_propulsion_conv * PROPORCION_HFO
            consumo_anual_lfo = consumo_anual_propulsion_conv * PROPORCION_LFO
            costes_fueleu_anual = (consumo_anual_hfo *
PENALIZACION_FUELEU_HFO[:vida_util_proyecto] +
                                   consumo_anual_lfo *
PENALIZACION_FUELEU_LFO[:vida_util_proyecto]) * 0.5
            costes_op_totales_conv_anual = costes_op_anuales_conv +
costes_fueleu_anual

            capex_inicial_conv, flujos_fin_conv =
calcular_flujos_financieros(CAPEX_POR_BUQUE_CONVENCIONAL * n_buques_conv,
vida_util_proyecto)
            costes_totales_anuales_conv = costes_op_totales_conv_anual+
flujos_fin_conv
```

```
            FR_conv = calcular_FR(capex_inicial_conv,
costes_totales_anuales_conv, [carga_anual_conv] * vida_util_proyecto,
tasa_descuento_actual)

            # --- Flota a vela ---
            tiempo_ciclo_vela = tiempo_ida_actual + tiempo_vuelta_actual +
sum(TIEMPOS_POR_CICLO_DIAS.values())
            carga_por_barco_vela_anual = DIAS_OPERACION_ANUAL /
tiempo_ciclo_vela * CAPACIDAD_CARGA_TONS if tiempo_ciclo_vela > 0 else 0
            n_buques_vela_necesarios = np.ceil(CARGA_ANUAL_OBJETIVO /
carga_por_barco_vela_anual) if carga_por_barco_vela_anual > 0 else float('inf')
            costes_op_anuales_vela, carga_anual_vela,_ =
simular_costes_y_carga_anual('vela', n_buques_vela_necesarios,
precio_comb_actual, tiempo_ida_actual, tiempo_vuelta_actual)
            capex_inicial_vela, flujos_fin_vela =
calcular_flujos_financieros(CAPEX_POR_BUQUE_VELA * n_buques_vela_necesarios,
vida_util_proyecto)
            costes_totales_anuales_vela = [costes_op_anuales_vela + fin for fin
in flujos_fin_vela]
            FR_vela = calcular_FR(capex_inicial_vela,
costes_totales_anuales_vela, [carga_anual_vela] * vida_util_proyecto,
tasa_descuento_actual)


            FRs_conv_escenario.append(FR_conv)
            FRs_vela_escenario.append(FR_vela)
        # Calcular la probabilidad para este escenario
        probabilidad = np.mean(np.array(FRs_vela_escenario) <
np.array(FRs_conv_escenario)) * 100
        resultados_finales.append({'tamano_flota_conv':
n_buques_conv,'vida_util': vida_util_proyecto,'probabilidad_rentabilidad_vela':
probabilidad})


# --- ANÁLISIS Y VISUALIZACIÓN DE RESULTADOS ---
df_final = pd.DataFrame(resultados_finales)

plt.figure(figsize=(12, 8))
sns.lineplot(data=df_final, x='vida_util', y='probabilidad_rentabilidad_vela',
hue='tamano_flota_conv',
             marker='o', markersize=8, palette='viridis', linewidth=2.5)

plt.title('Sensibilidad de la Rentabilidad de la Flota a Vela', fontsize=18,
fontweight='bold')
plt.xlabel('Vida Útil del Proyecto (Años)', fontsize=14)
plt.ylabel('Probabilidad de que la Flota a Vela sea más Rentable (%)',
fontsize=14)
plt.legend(title='Tamaño Flota Convencional de Ref.')
plt.grid(True, which='both', linestyle='--')
```

```
plt.ylim(0, 100)
plt.show()
```

Algoritmo 3

El siguiente código implementado en el cálculo de los anteriores permite calcular la variabilidad asumida por el fletador.

```
PRECIO_CARBON_POR_TONELADA = 110  # (€/tonelada)
TASA_COSTE_CAPITAL_CLIENTE = 0.08 # (8%) Tasa de oportunidad

tiempo_fijo_ciclo_dias = sum(TIEMPOS_POR_CICLO_DIAS.values())
consumo_diario_fabrica = CARGA_ANUAL_OBJETIVO / 365

# Tiempos de ciclo a vela
tiempo_ciclo_minimo = tiempos_vela_ida_dias.min() +
tiempos_vela_vuelta_dias.min() + tiempo_fijo_ciclo_dias
tiempo_ciclo_promedio = tiempos_vela_ida_dias.mean() +
tiempos_vela_vuelta_dias.mean() + tiempo_fijo_ciclo_dias
tiempo_ciclo_maximo = tiempos_vela_ida_dias.max() +
tiempos_vela_vuelta_dias.max() + tiempo_fijo_ciclo_dias

# Enfoque prudente (máximo vs. promedio)
retraso_prudente_dias = tiempo_ciclo_maximo - tiempo_ciclo_promedio
stock_seguridad_prudente_tons = consumo_diario_fabrica * retraso_prudente_dias
coste_anual_prudente = (stock_seguridad_prudente_tons *
PRECIO_CARBON_POR_TONELADA) * TASA_COSTE_CAPITAL_CLIENTE
sobrecoste_por_ton_prudente = coste_anual_prudente / CARGA_ANUAL_OBJETIVO

#Enfoque conservador (máximo vs. mínimo)
retraso_conservador_dias = tiempo_ciclo_maximo - tiempo_ciclo_minimo
stock_seguridad_conservador_tons = consumo_diario_fabrica *
retraso_conservador_dias
coste_anual_conservador = (stock_seguridad_conservador_tons *
PRECIO_CARBON_POR_TONELADA) * TASA_COSTE_CAPITAL_CLIENTE
sobrecoste_por_ton_conservador = coste_anual_conservador / CARGA_ANUAL_OBJETIVO
```

Plano de disposición general