



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Trayectorias de Consumo Eficiente: Un Modelo de Optimización con Variables Binarias

Grado en Ingeniería Mecánica

ALUMNO: Adriá Gavaldá González

DIRECTORES: Débora Cores Carrera

CURSO ACADÉMICO: 2024-2025

Universida_{de}Vigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Trayectorias de Consumo Eficiente: Un Modelo de Optimización
con Variables Binarias*

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

UniversidadeVigo

RESUMEN

A lo largo de la historia, la navegación ha sido un elemento clave en el desarrollo económico y estratégico de las naciones. Con el avance de la tecnología y la creciente preocupación por la eficiencia y el consumo energético en el transporte marítimo, se hace necesario el uso de herramientas que permitan optimizar trayectorias en función de múltiples restricciones.

En este contexto, el presente Trabajo Fin de Grado propone el desarrollo de un modelo de optimización de rutas marítimas mediante programación binaria en MATLAB, capaz de determinar la trayectoria mínima entre dos puntos sobre una malla espacial discretizada. Además, se contempla la posibilidad de introducir obstáculos dinámicos, simulando condiciones reales como zonas restringidas o áreas no navegables.

El algoritmo implementado, basado en la función `intlinprog`, ha sido validado a través de diferentes casos de estudio, entre ellos la conexión entre los puertos de Cádiz y Recife o entre A Coruña y Nueva York, mostrando cómo el sistema adapta la ruta en función del entorno. El resultado es una herramienta versátil que combina conceptos de optimización matemática, navegación marítima y simulación, con aplicaciones en logística, planificación táctica y eficiencia energética naval.

PALABRAS CLAVE

Programación Binaria, Rutas marítimas, Optimización, Obstáculo, Ortodrómica.

AGRADECIMIENTOS

En primer lugar, quiero agradecer a mi tutora, la profesora Débora Cores, por su inestimable ayuda en la realización de este trabajo. Su tiempo, paciencia y dedicación en cada una de las tutorías han sido fundamentales para poder llevarlo a cabo.

Agradezco a todos mis amigos y en especial la camareta 202 por tantas risas compartidas y el apoyo mutuo que ha hecho mucho más amena esta última singladura. Estoy seguro de que seguirán aportando el buen ambiente y compañerismo necesario tanto en lo profesional como en lo personal.

Finalmente, reservo mi último y más importante agradecimiento para mi familia, quienes han sido un pilar fundamental en el que siempre he podido apoyarme. Más aún, han sido el espejo y la fuente constante de inspiración que me impulsa a ser cada día un mejor profesional, y sobre todo, una mejor persona. Sin ellos, nada de lo que he logrado hasta ahora habría sido posible.

CONTENIDO

Contenido	1
Índice de Figuras	3
Índice de Tablas.....	4
1 Introducción y objetivos	5
1.1 Motivación	5
1.2 Antecedentes	6
1.3 Objetivo del trabajo.....	6
1.4 Estructura del trabajo	6
2 Estado del arte	8
2.1 Optimización lineal	8
2.1.1 Definición	8
2.1.2 Método Simplex.....	9
2.2 Optimización no lineal	10
2.2.1 Optimización no lineal sin restricciones (ONLSR)	10
2.2.2 Optimización no lineal con restricciones (ONLCR)	11
2.3 Optimización Binaria	13
2.3.1 Branch and Bound	14
2.4 MATLAB.....	14
2.4.1 Funciones para optimización lineal	15
2.4.2 Funciones para optimización no lineal	15
2.4.3 Funciones para optimización binaria	16
2.5 Aplicaciones en el transporte marítimo	16
2.5.1 Aplicaciones de Optimización lineal en el transporte marítimo.....	16
2.5.2 Aplicaciones de Optimización no lineal en el transporte marítimo.....	17
3 Desarrollo del TFG.....	18
3.1 Planteamiento del problema de forma matemática	18
3.1.1 Discretización gráfica del problema	18
3.1.2 Definición de nodos en la malla	19
3.1.3 Funcionamiento de los arcos en el nodo	19
3.1.4 Vector distancia	21
3.1.5 Modelo del problema.....	22
3.2 Funcionamiento del código.....	23
3.2.1 Argumentos de entrada	23

3.2.2 Argumentos de salida	23
3.2.3 Introducción de obstáculos	23
3.3 Descripción de los enunciados a resolver	24
3.3.1 Introducción a la navegación por círculo máximo	24
3.3.2 Ejemplo 1: Tránsito Almería-Melilla	25
3.3.3 Ejemplo 2: Tránsito Coruña – Nueva York	26
3.3.4 Ejemplo 3: Tránsito Cádiz – Recife.....	27
4 Resultados Obtenidos	30
4.1 Ejemplo 1: Tránsito Almería-Melilla.....	30
4.1.1 Sin obstáculos	30
4.1.2 Con obstáculos.....	32
4.2 Ejemplo 2: Tránsito Coruña – Nueva York	35
4.3 Ejemplo 3: Tránsito Cádiz-Recife.....	37
5 Conclusiones y líneas futuras	39
5.1 Conclusiones	39
5.2 Líneas Futuras	39
6 Bibliografía.....	41
Anexo I: Implicaciones Sociales, y/o Económicas, y/o Ambientales	44
Anexo II: Reflexiones Éticas y Sociales	46
Anexo III: Código del Programa	47

ÍNDICE DE FIGURAS

Figura 2-1: George Dantzig. Figura extraída de [12]	9
Figura 2-2: Método Simplex. Figura extraída de [13]	10
Figura 2-3: Ramificación Binomial. Figura extraída de [30]	14
Figura 2-4: Logo de MATLAB. Figura extraída de [33]	14
Figura 3-1: Discretización con nodos y arcos del espacio marítimo.(Fuente: Autoría Propia)	18
Figura 3-2: Numeración de los nodos para una discretización de $n = 7$ y $m=8$. (Fuente: Autoría Propia)	19
Figura 3-3: Cualquier nodo interno con sus 8 vecinos correspondientes. (Fuente: Autoría Propia)	20
Figura 3-4: Nodo ubicado en la esquina superior izquierda de la malla con sus 3 vecinos correspondientes. (Fuente: Autoría Propia)	20
Figura 3-5: Nodo lateral izquierdo de la malla que no esquina con sus 5 vecinos correspondientes. (Fuente: Autoría Propia)	21
Figura 3-6: Representación de ruta ortodrómica sobre el ecuador. Figura extraída de [46]	24
Figura 3-7: Representación de ruta ortodrómica sobre un meridiano. Figura extraída de [46]	25
Figura 3-8: Ruta entre los puertos de Almería y Melilla. Figura extraída de [47]	25
Figura 3-9: Ruta ortodrómica Coruña-Nueva York. (Fuente: Autoría Propia)	26
Figura 3-10: Ruta ortodrómica Cádiz-Recife. (Fuente: Autoría Propia)	27
Figura 3-11: Islas de Madeira y Portosanto. Figura extraída de [47]	27
Figura 3-12: Archipiélago de las canarias. Figura extraída de [47]	28
Figura 3-13: Archipiélago de Cabo Verde. Figura extraída de [47]	28
Figura 3-14: Isla de Villa dos remedios. Figura extraída de [47]	29
Figura 4-1: Discretización de la zona Almería-Melilla. (Fuente: Autoría Propia)	30
Figura 4-2: Ruta óptima entre puertos.(Fuente : Autoría Propia)	31
Figura 4-3: Obstáculos entre los puntos A-B. (Fuente: Autoría Propia)	32
Figura 4-4: Ruta optimizada evitando los obstáculos.(Fuente: Autoría Propia)	33
Figura 4-5: Malla 6 x 6. (Fuente: Autoría Propia)	34
Figura 4-6: Ruta ortodrómica Coruña-Nueva York. (Fuente: Autoría Propia)	35
Figura 4-7: Tránsito entre Cádiz-Recife. (Fuente: Autoría Propia)	37

ÍNDICE DE TABLAS

Tabla 2-1: Diferencia entre optimizaciones. (tomada de [21]).....	12
Tabla 2-2: Función y tipo de problema. (tomada de [34])	15
Tabla 2-3: Funciones de MATLAB y sus aplicaciones. (tomada de [36]).....	15
Tabla 2-4: Funciones binarias y su aplicación. (Fuente: Autoría Propia)	16
Tabla 4-1: Datos tránsito Almería-Melilla. (Fuente: Autoría Propia).....	31
Tabla 4-2: Valores optimización ruta Almería-Melilla. (Fuente: Autoría Propia).....	33
Tabla 4-3: Ruta ajustada a la malla 6 x 6. (Fuente: Autoría Propia)	34
Tabla 4-4: Valores optimización ruta Coruña-Nueva York. (Fuente: Autoría Propia).....	36
Tabla 4-5: Datos tránsito Cádiz-Recife. (Fuente: Autoría Propia).....	38

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Motivación

Desde tiempos muy antiguos, la navegación ha tenido un papel fundamental en la historia de la humanidad. Ya desde la antigua civilización egipcia con sus primeras barcas, pasando por las carabelas europeas hasta llegar a los barcos modernos como cargueros propulsados por combustibles fósiles [1].

Debido a los avances tecnológicos que han sufrido los buques a lo largo de la historia, se aprecia una notable diferencia en la dependencia de factores climatológicos como son el viento o las mareas en la propulsión de las naves por una dependencia en los combustibles fósiles gracias a la implementación de los motores de explosión.

Con esta dependencia a los combustibles fósiles surge la tesitura de cuanta dependencia tienen países de la Unión Europea como Alemania o España de los mayores países exportadores de dichos combustibles como Argelia, Rusia o Estados Unidos. Conflictos como la guerra entre Ucrania y Rusia pueden provocar aumentos de hasta un 60 % en el precio del barril de crudo.[2]. Además, a lo largo de la historia, numerosos conflictos han generado aumentos sostenidos en los precios, con solo reducciones temporales [3].

La Organización Marítima Internacional (OMI) estimó que en 2012 el transporte marítimo internacional generó unos 796 millones de toneladas de CO₂, lo que equivalía al 2,2 % de las emisiones globales de ese año. Posteriormente, el Cuarto Estudio de Gases de Efecto Invernadero (GEI) publicado en 2020 indica que en 2018 las emisiones alcanzaron los 1.056 millones de toneladas de CO₂, representando aproximadamente el 2,89 % de las emisiones antropogénicas mundiales [4].

Este trabajo se enmarca en esa línea de mejora, buscando optimizar trayectorias marítimas mediante técnicas matemáticas y computacionales, permitiendo simular diferentes condiciones y restricciones como zonas prohibidas o áreas de riesgo. La idea de poder calcular una ruta alternativa que esquive obstáculos y, aun así, represente la mejor opción posible en cuanto a distancia o coste, es una capacidad muy valiosa para planificadores logísticos, operadores portuarios o buques de guerra que requieren flexibilidad táctica.

La motivación de este trabajo, por tanto, no es únicamente técnica, sino también estratégica: aportar una herramienta que, basada en optimización entera y binaria, contribuya a mejorar la eficiencia operativa y energética del transporte marítimo, alineándose con los objetivos de sostenibilidad, seguridad y autonomía energética de Europa.

1.2 Antecedentes

La optimización de rutas de transporte constituye desde hace décadas un área de interés estratégico en sectores como la logística industrial y la planificación operativa militar. A lo largo del tiempo, el desarrollo de modelos matemáticos orientados a encontrar trayectorias eficientes ha avanzado de forma paralela al crecimiento de las capacidades computacionales, especialmente con el auge de herramientas como MATLAB y la consolidación de técnicas de la programación binaria.

En el ámbito civil, un ejemplo destacado de aplicación de la programación binaria en MATLAB es el desarrollado por Joel Val García en su Trabajo Fin de Grado titulado “*Desarrollo de una aplicación de logística en MATLAB para la optimización de rutas de transporte*” (2021) [5]. En este trabajo, el autor diseña un algoritmo orientado a obtener la ruta óptima de distribución entre múltiples localizaciones logísticas, tomando como caso real una cadena de supermercados con centros logísticos y puntos de venta distribuidos geográficamente. La solución propuesta se basa en la formulación del problema del viajante (TSP) y se implementa mediante programación lineal entera con variables binarias, utilizando la función *intlinprog*.

En el ámbito militar, un caso representativo de aplicación de la programación binaria es el estudio realizado por Liu et al. (2019) [6], titulado “*Optimization of Base Location and Patrol Routes for Unmanned Aerial Vehicles in Border Intelligence, Surveillance, and Reconnaissance*”, publicado en el Journal of Advanced Transportation. Este trabajo plantea un modelo matemático para optimizar simultáneamente la ubicación de bases operativas y las rutas de patrullaje de vehículos aéreos no tripulados (UAVs) en misiones de inteligencia, vigilancia y reconocimiento en zonas fronterizas. El modelo utiliza variables binarias para representar si una conexión entre dos nodos es utilizada ($x_{ij} = 1$) o no ($x_{ij} = 0$), integrando así decisiones discretas dentro de una red compleja de posibles rutas.

1.3 Objetivo del trabajo

Este trabajo tiene como objetivo desarrollar un modelo que optimice el cálculo de rutas marítimas óptimas sobre una malla discretizada, utilizando programación lineal entera con variables binarias, con el fin de minimizar la distancia recorrida entre dos puntos geográficos. El modelo debe permitir además la introducción dinámica de obstáculos en la red, simulando condiciones reales como zonas restringidas o no navegables, y demostrar su funcionalidad.

1.4 Estructura del trabajo

El presente documento se estructura en cinco capítulos diferenciados, con el objetivo de organizar adecuadamente el contenido desarrollado, así como los resultados obtenidos. A continuación, se describe brevemente el contenido de cada capítulo:

- **Capítulo 1:** Introducción y objetivos. Se presenta el contexto del trabajo, abordando la importancia de la optimización de rutas marítimas en entornos discretizados, su motivación, los antecedentes relevantes y los objetivos que se persiguen con este estudio.
- **Capítulo 2:** Estado del arte. Se realiza un análisis de los modelos de optimización utilizados en el ámbito marítimo, tanto lineales como no lineales, con especial atención a las técnicas de optimización binaria. Asimismo, se revisan las funciones de MATLAB aplicables a estos problemas y se destacan casos reales de aplicación en el transporte marítimo.
- **Capítulo 3:** Desarrollo del TFG. Se expone la formulación matemática del problema, detallando la discretización del espacio marítimo, la definición de nodos y arcos, y el funcionamiento del código desarrollado. También se incluye cómo se introducen obstáculos en la red y los diferentes casos de estudio simulados.
- **Capítulo 4:** Resultados obtenidos. Se analizan los resultados derivados de la simulación de diferentes rutas de navegación, tanto sin obstáculos como en presencia de restricciones. Se detallan las trayectorias óptimas, las distancias calculadas y las decisiones del algoritmo.

- **Capítulo 5:** Conclusiones y líneas futuras. Se evalúa el cumplimiento de los objetivos del trabajo y se reflexiona sobre las mejoras alcanzadas. Finalmente, se proponen posibles líneas de trabajo que pueden ampliar y enriquecer el modelo planteado.
- **Bibliografía y Anexos.** Se incluyen todas las referencias bibliográficas utilizadas, así como o extractos de código fuente relevante.

2 ESTADO DEL ARTE

El transporte marítimo desempeña un papel crucial en el comercio global, siendo responsable del traslado de una gran parte de las mercancías a nivel mundial. La eficiencia en la planificación y ejecución de rutas marítimas es fundamental para reducir costos operativos, minimizar el impacto ambiental y mejorar la seguridad en la navegación. En este contexto, la optimización de rutas marítimas se ha convertido en un área de estudio clave, en la que se aplican diversas técnicas matemáticas y computacionales para mejorar la eficiencia del transporte marítimo.

Este capítulo presenta un análisis detallado de los enfoques de optimización lineal y no lineal aplicados a la mejora de rutas marítimas. Se inicia con la definición y características de los problemas de optimización lineal y no lineal, destacando sus diferencias. Posteriormente, se exploran los principales métodos de resolución incluyendo enfoques analíticos y computacionales, como la programación lineal, los métodos de gradiente, los algoritmos de punto interior.

El objetivo de esta revisión es proporcionar una visión integral de los enfoques utilizados en la optimización de rutas marítimas, resaltando cómo la aplicación de estos métodos ha permitido mejorar la eficiencia operativa, reducir tiempos de navegación y optimizar el consumo de combustible en diferentes escenarios marítimos. Con ello, se busca establecer una base sólida para el desarrollo de futuras investigaciones y soluciones innovadoras en el ámbito del transporte marítimo.

2.1 Optimización lineal

2.1.1 Definición

La optimización lineal, también conocida como programación lineal (PL), es una técnica matemática utilizada para optimizar (maximizar o minimizar) una función objetivo lineal, sujeta a un conjunto de restricciones también lineales. Esta metodología es fundamental en campos como la economía, la ingeniería y la gestión empresarial para la toma de decisiones óptimas [7].

La programación lineal surgió como un modelo matemático durante la Segunda Guerra Mundial, diseñado para optimizar la planificación de gastos y retornos, con el objetivo de reducir costos en el ejército y maximizar el impacto en el enemigo. Debido a su importancia estratégica, su desarrollo se mantuvo en secreto hasta 1947.

Tras la guerra, la técnica se extendió a diversas industrias, que comenzaron a utilizarla en su planificación y toma de decisiones diarias. Entre sus principales impulsores se encuentran George Dantzig, quien en 1947 introdujo el algoritmo Simplex, John von Neumann, quien en el mismo año formuló la teoría de la dualidad, y Leonid Kantoróvich, un matemático ruso que ya había aplicado métodos similares en economía antes de los trabajos de Dantzig [8].

Un problema típico de optimización lineal ha de cumplir los siguientes requisitos:

1. La función objetivo debe ser lineal, lo que significa que todas las variables de decisión toman como valores número reales representados por letras con subíndices ($x_1, x_2, \dots$) o también como vector de un conjunto indexado de variables $x = (x_1, x_2, \dots)$.
2. El objetivo del problema debe ser maximizar o minimizar una función lineal, representando la meta del tomador de decisiones dentro del modelo de optimización [9].
3. Las restricciones representan las limitaciones prácticas impuestas por la disponibilidad de recursos o las condiciones físicas del entorno. Se formulan mediante ecuaciones e inecuaciones lineales en función de las variables de decisión. Matemáticamente, se expresa como se indica en (2-1) [10].

$$\begin{aligned} g_i(x) &\geq b_i; \quad g_i(x) \leq b_i; \quad g_i(x) = b_i \\ i &= 1, \dots, m; \text{ con } g_i \text{ una función lineal en } x \end{aligned} \quad (2-1)$$

De forma genérica, un modelo de optimización lineal se escribe como se indica en (2-2):

$$\left\{ \begin{array}{ll} \text{Max/Min} & f(x) \\ \text{Sujeto a} & Ax \leq b \\ & A_{eqx} = b_{eq} \\ & l_b \leq x \leq u_b \end{array} \right. \quad (2-2)$$

donde se tiene que la función lineal a optimizar es $f(x) = \sum_{i=1}^n a_i x_i$ sujeta a las siguientes restricciones [11]:

- Restricción lineal de desigualdad: $Ax \leq b$
- Restricción lineal de igualdad: $A_{eqx} = b_{eq}$
- Restricción de límites inferior y superior: $l_b \leq x \leq u_b$

2.1.2 Método Simplex

El Método Simplex es el enfoque más comúnmente utilizado para resolver problemas de optimización lineal. Se trata de un algoritmo versátil capaz de abordar cualquier problema de esta naturaleza, independientemente del número de variables involucradas. Fue desarrollado por el matemático George Dantzig, quien, durante su trabajo en la Rama de Análisis de Combate de los Cuarteles Centrales Estadísticos de la Fuerza Aérea de los Estados Unidos, propuso este método para resolver un problema de programación lineal. Su objetivo era determinar la duración óptima de las diferentes etapas de un programa de despliegue, entrenamiento y suministro logístico, con el fin de mejorar la eficiencia operativa y optimizar los recursos disponibles [12].

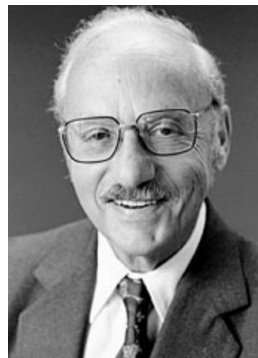


Figura 2-1: George Dantzig. Figura extraída de [12]

Se trata de un algoritmo iterativo que mejora progresivamente la solución en cada paso. Su fundamento matemático radica en que el método se desplaza de un vértice a otro dentro de un poliedro, seleccionando siempre un vértice adyacente que optimice el valor de la función objetivo, ya sea aumentando o disminuyendo su valor según corresponda (maximización o minimización). Dado que el número de vértices en un poliedro solución es finito, y siempre que se cumpla el conjunto de restricciones, el método garantiza la obtención de al menos una solución óptima [13].

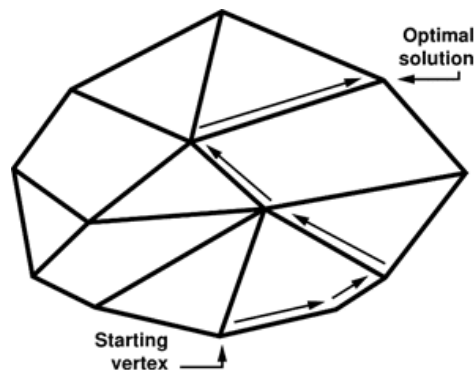


Figura 2-2: Método Simplex. Figura extraída de [13]

El Método Simplex siempre considera como posible solución un punto ubicado en uno de los vértices de la región factible. Inicialmente, la primera aproximación suele ser el origen. A partir de ahí, en cada iteración, el algoritmo avanza hacia un vértice adyacente, buscando mejorar el valor de la función objetivo. Este proceso continúa hasta alcanzar un vértice en el que el valor de la función objetivo no pueda mejorarse más respecto a los vértices vecinos. En ese momento, se ha encontrado la solución óptima del problema de programación lineal [14].

Hay 3 etapas para la resolución de problemas de optimización lineal con este método:

1. **Etapla inicial:** Se establece la primera solución factible, ubicándose en el vértice correspondiente al origen.
2. **Etapla iterativa:** El método avanza hacia otro vértice en busca de una solución mejor que la anterior.
3. **Etapla de prueba de optimalidad:** Se alcanza cuando la solución en un vértice resulta superior a la de sus vértices adyacentes, determinando así la solución óptima [14].

Existen cuatro posibles resultados en la resolución del problema:

- **Óptimo único:** Se encuentra una única solución óptima.
- **Óptimo no único:** Existen múltiples soluciones óptimas, lo que implica infinitas alternativas viables.
- **Problema no factible:** No existe ninguna solución que satisfaga simultáneamente todas las restricciones del problema.
- **Problema no acotado:** La solución puede crecer indefinidamente en un problema de maximización o disminuir sin límite en un problema de minimización, lo que implica que el problema no tiene una solución finita [15].

2.2 Optimización no lineal

2.2.1 Optimización no lineal sin restricciones (ONLSR)

En muchas situaciones, en la vida real nos podemos encontrar problemas no lineales sin restricciones. Es importante señalar que, si el propósito es la maximización, basta con minimizar la función con el signo cambiado. Por esta razón, en adelante se considerará la minimización sin pérdida de generalidad.

Los métodos de ONLSR siguen un enfoque iterativo. Se inicia con una estimación inicial x_0 y, tras determinar la dirección de búsqueda d_1 , se procede a encontrar el punto x_1 mediante una búsqueda monodimensional en esa dirección. La forma en que se define la dirección de búsqueda varía según el método empleado. De manera general, en la iteración k-ésima, el proceso se expresa como:

$$x_{k+1} = x_k + s_k \cdot d_k \quad (2-3)$$

Tenemos que s_k es el parámetro obtenido a través de la búsqueda monodimensional, como por ejemplo la condición de Armijo.

Como refleja el siguiente documento [16], hay diversos métodos numéricos que reducen este tipo de problema a un problema unidimensional. A continuación, se explicarán los más comunes:

1. **Método de Gradiente o de Cauchy:** Es una técnica de optimización iterativa utilizada para encontrar mínimos de funciones diferenciables. En cada iteración, se calcula la dirección de descenso, que corresponde al negativo del gradiente de la función $\left(d_k = -\frac{\nabla f(x_k)}{\|\nabla f(x_k)\|}\right)$ en el punto actual x_k . Con dicha dirección, se vuelve a minimizar la función con $s_k \in [s_{k1}, s_{k2}]$. El proceso se repite hasta que el gradiente en la iteración k-ésima sea nulo, lo que indica que se ha alcanzado un punto óptimo [17].
2. **Método de Newton:** Es un algoritmo general utilizado en optimización matemática para encontrar los puntos óptimos (mínimos o máximos) de una función $f(x)$. Se basa en la expansión en serie de Taylor (2-4), y la matriz Hessiana: $(\nabla^2 f(x_k))$, para mejorar la convergencia. Si la matriz Hessiana es definida positiva, d_k representa una dirección de descenso y minimiza el modelo [18].

$$f(x_k + d) \approx f(x_k) + d^t \nabla f(x_k) + \frac{1}{2} d^t \nabla^2 f(x_k) \cdot d \quad (2-4)$$

3. **Método de Cuasi-Newton:** Es un algoritmo basado en el método de Newton, diseñado para encontrar ceros, máximos o mínimos de funciones en ecuaciones no lineales o en problemas de optimización continua. Su principal ventaja radica en que permite aproximar la matriz jacobiana o hessiana cuando su cálculo directo resulta complicado o inviable [19].

2.2.2 Optimización no lineal con restricciones (ONLCR)

La optimización no lineal es una rama de la optimización matemática que se centra en maximizar o minimizar una función objetivo que no es lineal, considerando posibles restricciones que también pueden ser no lineales. Este tipo de optimización es esencial en situaciones donde las relaciones entre variables son complejas y no pueden ser representadas mediante ecuaciones lineales [20].

Esta optimización se formula de forma genérica como:

$$\left\{ \begin{array}{ll} \text{Min} & f(x) \text{ no lineal} \\ \text{Sujeto a} & Ax \leq b \\ & A_{eq}x = b_{eq} \\ & C(x) \leq 0 \\ & C_{eq}(x) = 0 \\ & l_b \leq x \leq u_b \end{array} \right. \quad (2-5)$$

A diferencia de la optimización lineal, este tipo de problemas puede incluir restricciones no lineales de igualdad y desigualdad:

- Restricción lineal de desigualdad: $Ax \leq b$
- Restricción lineal de igualdad: $A_{eq}x = b_{eq}$
- Restricción no lineal de desigualdad: $C(x) \leq 0$
- Restricción no lineal de igualdad: $C_{eq}(x) = 0$
- Restricción de límites inferior y superior: $l_b \leq x \leq u_b$

Como se refleja en la siguiente Tabla 2-1, observamos las diferencias entre el modelo de optimización lineal y el modelo de optimización no lineal:

Aspecto	Programación lineal	Programación no lineal
Naturaleza del problema	Optimización con restricciones y funciones lineales	Optimización con restricciones y funciones no lineales
Relaciones en el modelo	Todas las restricciones y la función objetivo son lineales	Las restricciones y la función objetivo pueden ser no lineales
Solución optimizada	Se puede encontrar la solución en un número finito de iteraciones	Puede no ser posible encontrar la solución en un número finito de iteraciones
Determinación de la solución	Se determina al localizar los puntos extremos de la región factible	Se pueden usar puntos extremos, interiores y factibles para encontrar la solución
Aplicabilidad de métodos tradicionales	Los métodos de cálculo tradicionales siempre son aplicables	En algunos casos, los métodos de cálculo tradicionales pueden no ser aplicables
Eficacia de métodos iterativos	Menos efectivos en comparación con métodos iterativos de la programación no lineal	Más efectivos los métodos iterativos para encontrar la solución

Tabla 2-1: Diferencia entre optimizaciones. (tomada de [21])

Como se pueden ver en documentos como [22] o [23] estos son algunos de los métodos más comunes para la resolución de este tipo de problemas:

1. **Método de puntos interiores:** Los algoritmos de punto interior exploran el interior del conjunto factible mediante un proceso iterativo, acercándose progresivamente a la solución óptima.
Con cada iteración, se aproxima progresivamente a la solución hasta cumplir con la condición de convergencia.
Cabe destacar que existen diversas variantes y optimizaciones de estos algoritmos, cuya implementación puede diferir según el enfoque utilizado. Gracias a su alta eficiencia, los métodos de punto interior son especialmente adecuados para resolver problemas de optimización lineal a gran escala, representando una alternativa robusta al método símplex de optimización lineal [24].
2. **Método de programación cuadrática sucesiva:** Este algoritmo aborda problemas de optimización con función objetivo y restricciones no lineales de manera iterativa, empleando

aproximaciones cuadráticas para la función objetivo y aproximaciones lineales para las restricciones. Además, implementa una estrategia basada en el conjunto activo de restricciones para guiar el proceso de optimización [25].

La principal ventaja de los métodos de programación cuadrática sucesiva es su rápida convergencia, requiriendo un número reducido de evaluaciones de las funciones. No obstante, en problemas con un alto número de grados de libertad, los métodos de gradiente reducido pueden ser una opción más eficiente, ya que permiten disminuir la dimensionalidad del problema a resolver [26].

3. **Método de región de confianza:** Estos métodos se fundamentan en un modelo que aproxima la función objetivo, generalmente utilizando una función cuadrática. En cada iteración, los métodos de región de confianza establecen un área en la que se espera que el modelo represente con precisión la función objetivo, conocida como región de confianza. Si el modelo de la función objetivo dentro de la región de confianza es preciso, la región se amplía; en caso contrario, si la aproximación resulta inexacta, la región se reduce progresivamente hasta encontrar un modelo adecuado de la función [27].

2.3 Optimización Binaria

La programación lineal (PL) es una de las herramientas fundamentales en optimización matemática, utilizada para maximizar o minimizar una función objetivo sujeta o no a restricciones lineales.

La programación lineal entera (PLE) es una extensión de la programación lineal en la que algunas o todas las variables de decisión están restringidas a ser valores enteros. Mientras que en PL las soluciones pueden ser números continuos, en PLE las decisiones deben ser discretas, lo que la hace ideal para modelar situaciones donde las fracciones no tienen sentido, como la asignación de trabajadores, la planificación de producción o la optimización de rutas.

Según el tipo de variables, los problemas de programación entera se clasifican en:

1. **Enteros puros:** Todas las variables son enteras. Si además los coeficientes del problema también lo son, se considera totalmente entero.
2. **Mixtos:** Combinan variables continuas y enteras.
3. **Binarios:** Las variables solo pueden tomar valores 0 o 1 [28].

En este apartado nos centraremos en la programación lineal entera binaria, la cual al representar decisiones discretas: sí o no, activo o inactivo, seleccionado o no seleccionado. En estos casos, la programación binaria juega un papel clave, ya que restringe las variables a solo dos valores posibles:

$$x_i \in \{0,1\} \quad (2-6)$$

Según lo indicado en [29], un problema de optimización binaria se expresa como (2-7):

$$\left\{ \begin{array}{ll} \text{Min} & f(x) = c^T x \\ \text{Sujeto a} & Ax \leq b \\ & x_i \in \{0,1\}, \forall i \end{array} \right. \quad (2-7)$$

Donde:

- x : es un vector de variables binarias
- c : es un vector de coeficientes en la función objetivo.
- A : es una matriz de restricciones.
- b : es un vector de restricciones.

En este tipo de problemas, la solución se obtiene mediante iteraciones, eligiendo un camino u otro según si la tarea ha sido completada ($x = 1$) o, por el contrario, no se ha llevado a cabo ($x = 0$) [30]. Gráficamente se encuentra explicado en Figura 2-3, donde se puede observar todos los caminos posibles.

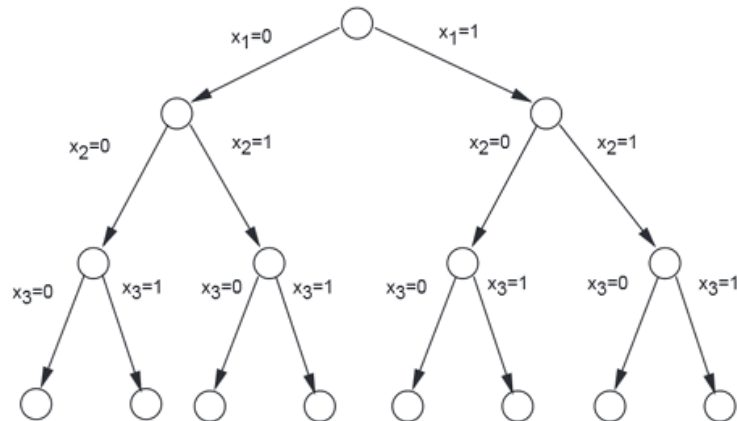


Figura 2-3: Ramificación Binomial. Figura extraída de [30]

En este contexto, los algoritmos utilizados para resolver modelos de Programación Entera difieren de aquellos empleados en Programación Lineal. Entre los más destacados se encuentran el Algoritmo de Ramificación y Acotamiento (Branch & Bound), Branch & Cut, Planos Cortantes y Relajación Lagrangeana, entre otros [31].

2.3.1 Branch and Bound

El método Branch and Bound (Ramificación y Acotamiento) es un algoritmo diseñado para resolver modelos de Programación Entera. Su enfoque consiste en transformar el problema en un modelo de Programación Lineal, resolviéndolo como tal y generando cotas cuando alguna variable entera toma un valor fraccionario.

El algoritmo aplica de manera recursiva restricciones adicionales para acercar las soluciones a valores enteros. Este proceso, conocido como relajación continua, permite resolver la versión lineal del modelo entero y refinar la solución hasta cumplir con la condición de enteridad en todas las variables de decisión [32].

2.4 MATLAB

MATLAB es un entorno de programación y software de cálculo numérico utilizado para realizar análisis de datos, modelado, simulación y visualización de resultados, facilitando la resolución de problemas matemáticos complejos [33].



Figura 2-4: Logo de MATLAB. Figura extraída de [33]

Mencionado el software que se va a usar para el trabajo, a continuación, se muestran varias funciones englobadas para diferentes optimizaciones.

2.4.1 Funciones para optimización lineal

MATLAB ofrece potentes herramientas para resolver problemas de optimización lineal a través de funciones que se encuentran en toolboxes especializados. La toolbox especializada de optimización incluye solvers para programación lineal (LP), programación lineal de enteros mixtos (MILP), programación cuadrática (QP), programación no lineal (NLP), mínimos cuadrados lineales restringidos, mínimos cuadrados no lineales y ecuaciones no lineales, incluyendo la programación lineal [34]. A continuación, se detallan las principales funciones:

Tipo de problema	Función
Programación Lineal	linprog
Programación Cuadrática Sucesiva	quadprog
Mínimos Cuadrados Lineales	lsqlin

Tabla 2-2: Función y tipo de problema. (tomada de [34])

Hay que destacar la función *linprog*;, como la función principal para abordar problemas de programación lineal de la forma (2-2). Esta función permite minimizar una función objetivo lineal sujeta a restricciones lineales de igualdad y desigualdad, así como a límites en las variables [35].

2.4.2 Funciones para optimización no lineal

En la Tabla 2-3 se muestran las funciones más comunes para resolver problemas de optimización no lineal

Función	Aplicación
fminbnd	Encontrar el mínimo de una función de una única variable en un intervalo fijo
fmincond	Encontrar el mínimo de una función multivariable no lineal restringida
fminsearch	Encontrar el mínimo de una función multivariable no restringida utilizando el método sin derivadas
fminunc	Encontrar el mínimo de una función multivariable no restringida
fseminf	Encuentra el mínimo de una función no lineal multivariable con restricciones semi-infinitas

Tabla 2-3: Funciones de MATLAB y sus aplicaciones. (tomada de [36])

2.4.3 Funciones para optimización binaria

En la Tabla 2-4 se muestran las diferentes funciones para la resolución de problemas de optimización binaria.[37], [38], [39]:

Aplicación	Función
Resuelve problemas de optimización binaria y entera mixta.	intlinprog
Utilizado para optimización binaria en problemas no lineales.	ga
Resuelve problemas cuando la función es no diferenciable, discontinua o multimodal	particleswarm

Tabla 2-4: Funciones binarias y su aplicación. (Fuente: Autoría Propia)

2.5 Aplicaciones en el transporte marítimo

La optimización de rutas marítimas es esencial para mejorar la eficiencia operativa, reducir costos y minimizar el impacto ambiental en el transporte marítimo. Tanto los métodos de optimización lineal como los no lineal se han aplicado con éxito en este ámbito.

2.5.1 Aplicaciones de Optimización lineal en el transporte marítimo

1. **Optimización de Flotas de Portacontenedores:** En el estudio "Optimización de flotas de portacontenedores para rutas en la costa atlántica europea", se desarrolla un modelo de programación lineal para maximizar el beneficio de una naviera. El modelo optimiza una flota de buques portacontenedores de pequeño tamaño en el transporte multimodal, que integra tramos terrestres y marítimos, es decir, tráfico marítimo de corta distancia. Para lograr esta optimización, se analizan diversos factores, incluyendo los flujos de mercancía contenerizada entre regiones, las rutas más probables para dichos flujos, las frecuencias necesarias para atender la demanda y la interacción entre el buque y el puerto.. Este enfoque permite una gestión más eficiente de la flota y una reducción en los costos operativos.[40]
2. **Programación lineal aplicada a la optimización de rutas de transporte en SXR-POLYMERS:** La optimización de rutas de transporte en la empresa SXR-POLYMERS se llevó a cabo mediante un modelo de programación lineal de transporte, cuyo objetivo principal fue minimizar los costos logísticos y mejorar la eficiencia en la distribución de productos. Para ello, se analizaron múltiples variables, como la ubicación de los clientes, los costos de transporte, la oferta y la demanda, además de evaluar diferentes grupos de transporte según criterios de capacidad, calidad, seguridad y rentabilidad. En este modelo, las variables de decisión (X_{ij}) representaban la cantidad de producto transportado desde cada origen (i) hasta cada destino (j), mientras que la función objetivo buscaba minimizar los costos totales mediante (2-8) :

$$MinZ = \sum_{i=1}^4 \sum_{j=1}^4 X_{ij}C_{ij} \quad (2-8)$$

Donde C_{ij} representa el costo unitario de transporte entre cada punto de origen y destino. Además, se establecieron restricciones de oferta, que limitaban la cantidad de producto enviado según la disponibilidad en cada origen, y restricciones de demanda, asegurando que cada destino recibiera la cantidad necesaria. Finalmente, la restricción de no negatividad garantizó que todas las variables fueran mayores o iguales a cero, evitando asignaciones irreales. Gracias a este planteamiento, se logró una reducción significativa de costos,

alcanzando un total de \$118,250, con la solución validada mediante el software QM for Windows [41].

2.5.2 Aplicaciones de Optimización no lineal en el transporte marítimo.

1. **SMART SHIPPING: Optimización de rutas marítimas:** Presenta un innovador sistema de weather routing que utiliza datos meteorológicos y oceanográficos en tiempo real para optimizar las rutas marítimas entre puertos. Este enfoque tiene como principales objetivos minimizar el consumo de combustible, reducir las emisiones de gases de efecto invernadero y mejorar la seguridad en la navegación al evitar condiciones climáticas adversas. Para lograr esta optimización, se emplean diversos métodos matemáticos, incluyendo técnicas de optimización variacional, que formulan el problema como una optimización continua en el espacio mediante gradientes; modelos de grafos, donde el océano se discretiza en una malla de nodos para aplicar algoritmos de búsqueda óptima como A y Dijkstra; y programación dinámica, que ajusta la trayectoria del buque en función de las condiciones cambiantes. Además, se implementan algoritmos genéticos, que iterativamente evolucionan soluciones mediante procesos de selección y mutación, permitiendo encontrar rutas cada vez más eficientes [42].
2. **Simultaneous Planning of Liner Ship Speed Optimization, Fleet Deployment, Scheduling and Cargo Allocation with Container Transshipment:** El Proyecto un modelo MINLP que optimiza simultáneamente la velocidad de los buques, el despliegue de la flota, la programación de servicios y la asignación de carga con transbordo de contenedores, buscando equilibrar costos operativos y tiempos de transporte, considerando el consumo de combustible y las emisiones de CO₂. Para resolver esta compleja optimización, se emplean algoritmos evolutivos multiobjetivo (MOEAs), como NSGA-II y OCEA, que permiten obtener soluciones óptimas en distintos escenarios operativos. El modelo incorpora un enfoque avanzado al considerar que el consumo de combustible depende tanto de la velocidad como de la carga del buque, diferenciándose de modelos más simplificados. Su función objetivo busca minimizar dos criterios contradictorios: el costo total del transporte, incluyendo operación, combustible, tasas portuarias y emisiones de CO₂, y el tiempo total de transporte [43].

3 DESARROLLO DEL TFG

3.1 Planteamiento del problema de forma matemática

El presente trabajo tiene como objetivo la optimización de rutas entre dos puntos. El problema o modelo empleado permitirá que, mediante una optimización binaria, el programa sea capaz de seleccionar aquella ruta que minimice la distancia teniendo en cuenta posibles obstáculos que puedan aparecer.

3.1.1 Discretización gráfica del problema

El problema será modelado en el plano 2D gracias a la discretización del espacio marítimo en el eje x (longitud) y en el eje y (latitud). La división en cuadrículas de la malla generada vendrá dada en función del número de columnas (parámetro m) y filas que se introduzcan (parámetro n), donde el punto inicial (“A”) y final (“B”) se encontrarán en el interior de la malla; tal y como se muestra en la Figura 3-1:

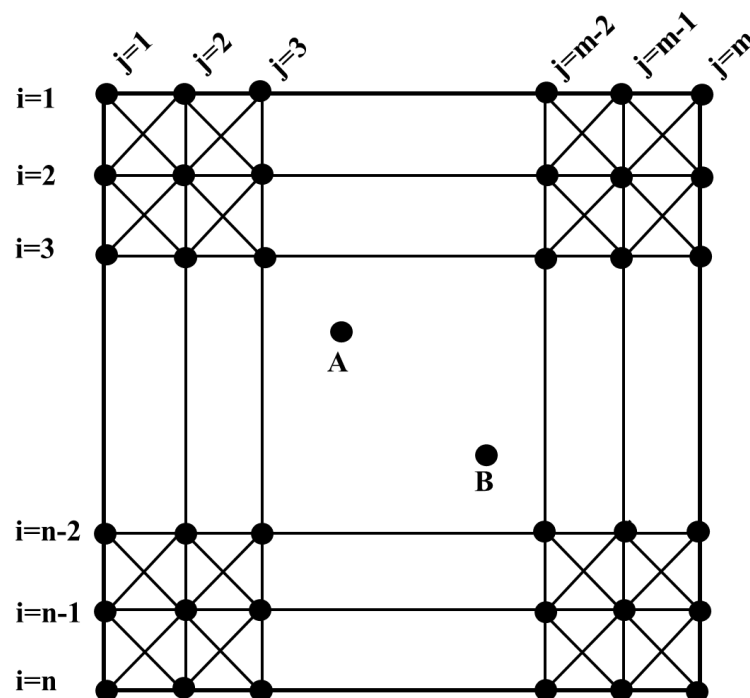


Figura 3-1: Discretización con nodos y arcos del espacio marítimo.(Fuente: Autoría Propia)

El objetivo del trabajo es, una vez generado los arcos de unión entre todos los nodos, encontrar la ruta que une los puntos A y B con mínima distancia. El resultado obtenido será todos los puntos GPS por donde deberá pasar el barco.

3.1.2 Definición de nodos en la malla

Como se menciona en el apartado 3.1.1, los nodos son creados debido al cruce de intersecciones entre latitudes y longitudes, como se muestra en la Figura 3-2. Además, el número total de estos nodos sigue la fórmula (3-1) :

$$nodos = nn \times mm \quad (3-1)$$

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$
$i=1$	● 1	● 8	● 15	● 22	● 29	● 36	● 43	● 50
$i=2$	● 2	● 9	● 16	● 23	● 30	● 37	● 44	● 51
$i=3$	● 3	● 10	● 17	● 24	● 31	● 38	● 45	● 52
$i=4$	● 4	● 11	● 18	● 25	● 32	● 39	● 46	● 53
$i=5$	● 5	● 12	● 19	● 26	● 33	● 40	● 47	● 54
$i=6$	● 6	● 13	● 20	● 27	● 34	● 41	● 48	● 55
$i=7$	● 7	● 14	● 21	● 28	● 35	● 42	● 49	● 56

Figura 3-2: Numeración de los nodos para una discretización de $n = 7$ y $m=8$. (Fuente: Autoría Propia)

Tal y como se puede observar en la Figura 3-2, el número total de nodos para $n=7$, $m=8$ es $7 \times 8 = 56$.

Además, para cada nodo k , le corresponde una fila i y una columna j , que determina la numeración del nodo mediante la fórmula (3-2), como se puede observar en la Figura 3-2.

$$k = (j - 1) \cdot n + i \quad (3-2)$$

Por ejemplo, el nodo en la fila $i = 3$, columna $j=4$, en la Figura 3-2 es $k = (4 - 1) \cdot 7 + 3 = 24$.

3.1.3 Funcionamiento de los arcos en el nodo

Para cada par de nodos i, j , se define un arco (i,j) , si existe un arco factible del nodo i al nodo j .

Los arcos factibles se definen para cada nodo $i = 1, 2, \dots, n$, con los nodos vecinos j inmediatos. Denotamos el arco factible que va del nodo i al nodo j como s_{ij} . Entonces $s_{ij} \neq s_{ji}$, ya que el arco s_{ij} representa aquel que sale del nodo i y llega al nodo j , mientras que el arco s_{ji} representa lo contrario.

Por otra parte, hay arcos que no existen debido a la situación del nodo dentro de la malla. Por ejemplo, el nodo 24 de la Figura 3-2 solo puede ir a los nodos vecinos 16,17,18,23,25,30,31 y 32. A continuación, la Figura 3-3, Figura 3-4, Figura 3-5 muestran algunas de las posibilidades de nodos vecinos dentro de la malla:

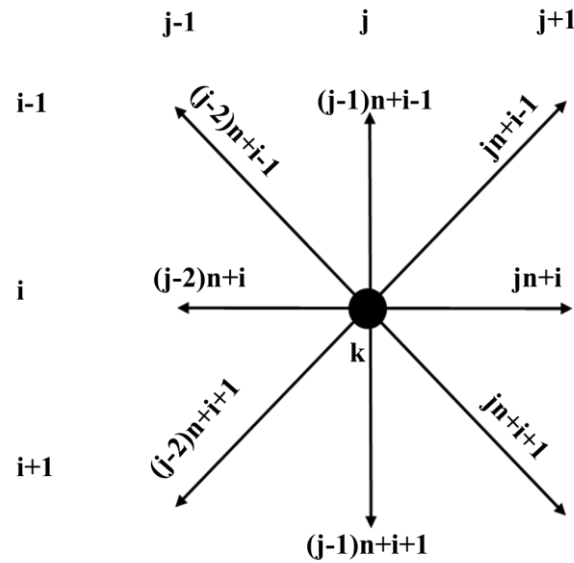


Figura 3-3: Cualquier nodo interno con sus 8 vecinos correspondientes. (Fuente: Autoría Propia)

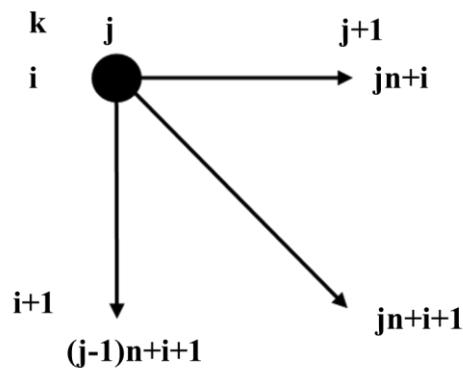


Figura 3-4: Nodo ubicado en la esquina superior izquierda de la malla con sus 3 vecinos correspondientes. (Fuente: Autoría Propia)

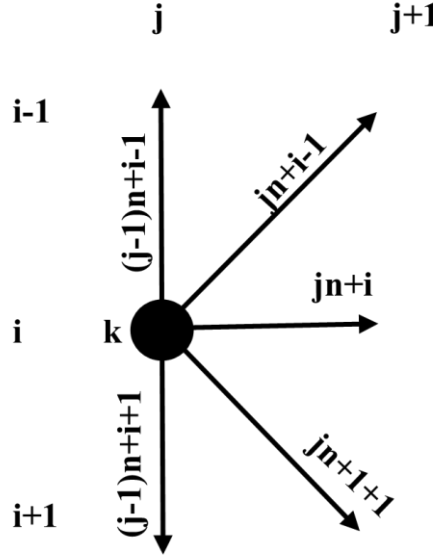


Figura 3-5: Nodo lateral izquierdo de la malla que no esquina con sus 5 vecinos correspondientes. (Fuente: Autoría Propia)

Una vez definidos los nodos de la discretización y los arcos factibles, denotamos por S_k el vector que contiene todos los arcos factibles que salen de cada nodo k , $k=1,2,\dots,n\cdot m$, a los nodos vecinos, entonces:

- $S_k \in \mathbb{R}^8$ si k es nodo interno de la malla.
- $S_k \in \mathbb{R}^5$ si k es nodo lateral de la malla sin estar en las esquinas de la malla.
- $S_k \in \mathbb{R}^3$ si k es nodo en alguna de las cuatro esquinas de la malla.

Sea $S = (S_1, S_2, \dots, S_{n\cdot m})$ el vector que contiene todos los arcos factibles para los $n \cdot m$ nodos de la malla, el número total de arcos factibles viene dado por :

$$\text{arcos totales} = 2 \cdot (4nm - 3m - 3n + 2) = 2 \cdot \text{dimen} \quad (3-3)$$

Por lo tanto, el vector S conteniendo todos los arcos factibles es:

$$S = (S_1, S_2, \dots, S_{n\cdot m}) = (s_1, s_2, \dots, s_{2\cdot \text{dimen}}) \in \mathbb{R}^{2\cdot \text{dimen}} \quad (3-4)$$

3.1.4 Vector distancia

Sea d el vector que contiene las distancias asociadas a todos los arcos factibles. A cada arco factible s_k le corresponde una distancia d_k , entonces el vector vendrá dado por:

$$d = (d_1, d_2, \dots, d_{2\cdot \text{dimen}}) \in \mathbb{R}^{2\cdot \text{dimen}} \quad (3-5)$$

La distancia del arco que une el nodo k de coordenadas en grados $(l_{i,k}, L_{i,k})$, donde $l_{i,k}$ es la latitud y $L_{i,k}$ la longitud, con el nodo vecino r de coordenadas en grados $(l_{i,r}, L_{i,r})$, viene dada por la fórmula de Harversine:

$$\begin{aligned}
d(k, r) &= d((l_{i,k}, L_{i,k}) (l_{i,r}, L_{i,r})) \\
&= 2 \cdot R \cdot \arcsen \left(\sqrt{\sin^2 \left(\frac{l_{i,r} - l_{i,k}}{2} \right) + \cos(l_{i,k}) \cdot \cos(l_{i,r}) \cdot \sin^2 \left(\frac{L_{i,r} - L_{i,k}}{2} \right)} \right)
\end{aligned} \tag{3-6}$$

El uso de la fórmula (3-6) es debido a que determina la distancia del círculo máximo entre dos puntos en una esfera a partir de sus coordenadas de longitud y latitud.[44]

3.1.5 Modelo del problema

Una vez definida la malla de la discretización del espacio marítimo y los vectores S de incógnitas del problema y el vector d que contiene las distancias de los arcos factibles, el problema propuesto se escribe como:

$$\left\{ \begin{array}{ll}
\text{Min}_{S \in \mathbb{R}^{2 \cdot \text{dimen}}} \sum_{k=1}^{2 \cdot \text{dimen}} s_k \cdot d_l & (3-7) \\
\sum_{\bar{K}=\{\text{índices de los arcos que salen de } k\}} s_{\bar{K}} - \sum_{\bar{K}=\{\text{índices de los arcos que entran en } k\}} s_{\bar{K}} = 0 & (3-8) \\
\text{para } k = 1, 2, \dots, 2 \cdot \text{dimen} & \\
\sum_{\bar{K}=\{\text{índices de los arcos que salen del nodo } A\}} s_{\bar{K}} - \sum_{\bar{K}=\{\text{índices de los arcos que entran en el nodo } A\}} s_{\bar{K}} = 1 & (3-9) \\
\sum_{\bar{K}=\{\text{índices de los arcos que salen del nodo } B\}} s_{\bar{K}} - \sum_{\bar{K}=\{\text{índices de los arcos que entran en el nodo } B\}} s_{\bar{K}} = -1 & (3-10) \\
s_{\bar{K}} = 0 & (3-11) \\
\bar{K} = \{\text{índices de los arcos no transitables}\} & \\
s_k \in \{0, 1\} & (3-12) \\
k = 1, 2, \dots, 2 \cdot \text{dimen} &
\end{array} \right. \text{ Sujeto a }$$

Donde:

- s_l es el arco factible en la posición l del vector S
- d_l es la distancia asociada al arco s_l .
- El problema se encuentra sujeto a cuatro restricciones que indican que:
 - La restricción (3-8) fuerza que la trayectoria sea un camino continuo tal, el número de arcos que entran en un nodo intermedio es igual al número de arcos que salen de dicho nodo. A excepción de los nodos A y B. El número de ecuaciones que se generan con dicha restricción sigue la fórmula:

$$n^{\circ} \text{ ecuaciones} = 2 * \text{dimen} \quad (3-13)$$

- La restricción (3-9) y fuerza que el número de arcos que sale de A sea 1, teniendo una sola ecuación.
- La restricción (3-10) fuerza que el número de arcos que entren en B sea 1, teniendo una sola ecuación.
- La restricción (3-11) fija los arcos no transitables con un valor de 0. El número de ecuaciones generadas depende de la decisión del usuario sobre cuantos obstáculos quiere añadir y que arcos restringir su paso.
- La restricción (3-12) restringe los valores de los arcos a 0 y 1.

3.2 Funcionamiento del código

Como se ha mencionado previamente, existen diversos programas y software con la capacidad de resolver modelos de optimización binaria. En este trabajo, se empleará MATLAB, utilizando específicamente la función "intlinprog", como se observa en el apartado 2.4.3 ,para el desarrollo del trabajo.

3.2.1 Argumentos de entrada

La función "intlinprog" de MATLAB requiere que el usuario proporcione una serie de parámetros que definen el problema de optimización. Estos parámetros deben especificarse dentro de los paréntesis en la llamada a la función, siguiendo la estructura:

$$[f] = \text{intlinprog}(f, \text{intcon}, A, b, Aeq, beq, lb, ub) "$$

- *f*: vector que contiene las distancias entre nodos. Representa el vector de coeficientes de la función objetivo a minimizar.
- *intcon*: vector de índices que indica qué variables deben ser enteras. En este caso, se define como $1:2 * \text{dimen}$, donde *dimen* es calculado mediante la correspondiente ecuación (3-3)
- *A*, *b*: se tratan de la matriz y el vector que corresponden con la desigualdad lineal (2-5).
- *Aeq*, *beq*: se tratan de la matriz y el vector que corresponden con la restricción de igualdad.(2-5). La matriz *Aeq* está asociada a las restricciones (3-8), (3-9), (3-10), (3-11) asegurando un camino continuo entre los puntos A y B. El vector *beq* contiene valores 0, -1, 1, según la restricción.
- *lb,ub*: vectores que definen los límites superior e inferior: $s_k \in \{0,1\}$. En el trabajo, el arco valdrá 0 si se encuentra dentro de la ruta óptima y 1 en el caso contrario.

Esta función se usaría para buscar la ruta óptima. Sin embargo, en el supuesto que hubiese algún obstáculo entre ambos puertos a mayores se usaría la función:

3.2.2 Argumentos de salida

En esta sección se explicará que datos obtendrá el usuario a modo de resultado de la optimización de la ruta:

$$[s, fval, \text{exitflag}, \text{output}] = \text{intlinprog}()$$

- *s*: vector con ceros y unos. Las posiciones con valores 1 corresponden a los arcos de la malla marítima que se han seleccionado en la trayectoria óptima.
- *fval*: Representa la distancia mínima recorrida entre el puerto inicial y el final.

3.2.3 Introducción de obstáculos

En la navegación real, pueden existir áreas restringidas o inaccesibles que deben ser evitadas en la planificación de rutas.

Para introducir obstáculos en la simulación de la ruta óptima entre los puertos, el programa emplea la matriz de restricciones Aeq y el vector beq . Las restricciones con los obstáculos (3-11) corresponde a cada fila de la matriz Aeq , que tiene un valor uno en la posición del arco no transitable y ceros en el resto de las posiciones. Por lo que la matriz Aeq tienen tantas filas como arcos no transitables. El vector beq es un vector de ceros de tamaño igual al número de arcos no transitable.

Al incorporar estos obstáculos, el programa ajusta automáticamente la trayectoria óptima, recalculando las posibles rutas sin violar las restricciones impuestas. Gracias a la construcción de Aeq y beq , se pueden modificar dinámicamente las condiciones del entorno sin necesidad de reestructurar el modelo completo. Como resultado, la función de optimización “intlinprog” genera una nueva solución viable que minimiza la distancia recorrida evitando los obstáculos, lo que permite una simulación realista de condiciones de navegación en presencia de restricciones operativas.

3.3 Descripción de los enunciados a resolver

En este apartado se presentan las coordenadas geográficas de los puntos de partida y destino correspondientes a diversas localizaciones, con el objetivo de determinar las rutas óptimas de navegación entre ellos.

Asimismo, se analiza cómo el programa responde a la introducción de obstáculos en la trayectoria, evaluando su capacidad para recalcular y ajustar la ruta en función de las restricciones impuestas.

3.3.1 Introducción a la navegación por círculo máximo

La distancia más corta entre dos puntos en la superficie terrestre está determinada por un arco de círculo máximo que conecta ambas ubicaciones. En navegación, esta trayectoria se conoce como derrota ortodrómica. Por lo tanto, para recorrer la mínima distancia posible entre dos puntos, es necesario seguir esta derrota.

Sin embargo, en una carta de navegación en proyección Mercator, la derrota ortodrómica no se representa como una línea recta, sino como una línea curva, debido a la distorsión de la proyección. Las únicas excepciones a esta representación ocurren cuando la navegación se realiza a lo largo de un meridiano o del ecuador, donde la trayectoria se muestra como una línea recta.[45]

Esto se debe a que las cartas Mercator representan el ecuador como una línea horizontal recta y los meridianos como líneas verticales rectas. Sin embargo, dado que los meridianos convergen en los polos en la realidad, esta proyección genera una distorsión progresiva de las áreas, aumentando significativamente a medida que se aleja del ecuador.[46]

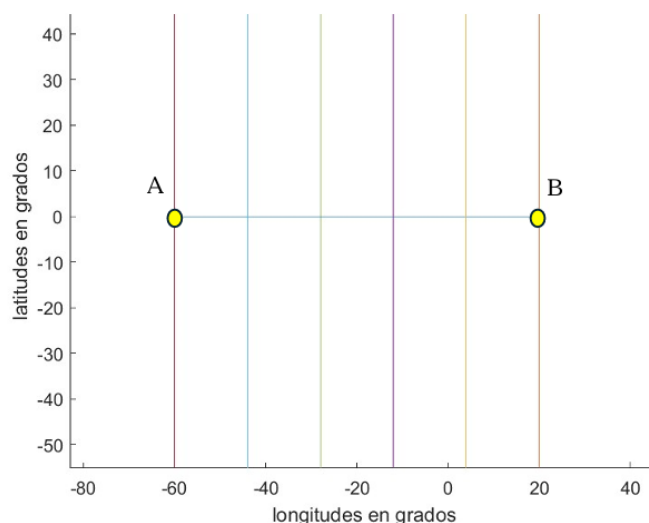


Figura 3-6: Representación de ruta ortodrómica sobre el ecuador. Figura extraída de [46]

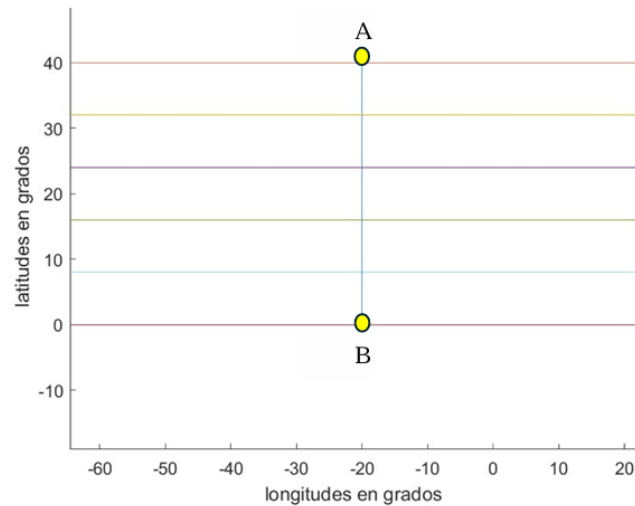


Figura 3-7: Representación de ruta ortodrómica sobre un meridiano. Figura extraída de [46]

3.3.2 Ejemplo 1: Tránsito Almería-Melilla

En la primera navegación, se procede a calcular con precisión la distancia mínima entre el puerto de Almería, de coordenadas $35^{\circ} 16' 48''\text{N}$ y $2^{\circ} 27' \text{W}$, y el puerto de Melilla, de coordenadas $35^{\circ} 16' 48'' \text{N}$ y $2^{\circ} 56' 24'' \text{W}$ tal como se ilustra en la Figura 3-8. Además, se analiza el proceso mediante el cual el programa selecciona los nodos que establecen la conexión entre ambos puertos.



Figura 3-8: Ruta entre los puertos de Almería y Melilla. Figura extraída de [47]

3.3.3 Ejemplo 2: Tránsito Coruña – Nueva York

En la segunda navegación se calcula la distancia mínima entre el puerto de Nueva York de coordenadas $40^{\circ} 37' 12''$ N y $74^{\circ} 4' 12''$ W y el puerto de Lisboa de coordenadas $43^{\circ} 21' 36''$ N $8^{\circ} 24' 36''$ W, puertos que no tienen ningún obstáculo entre ambos, demostrando como el programa optimiza la ruta respetando la navegación por círculo máximo mencionado en la sección 3.3.1.

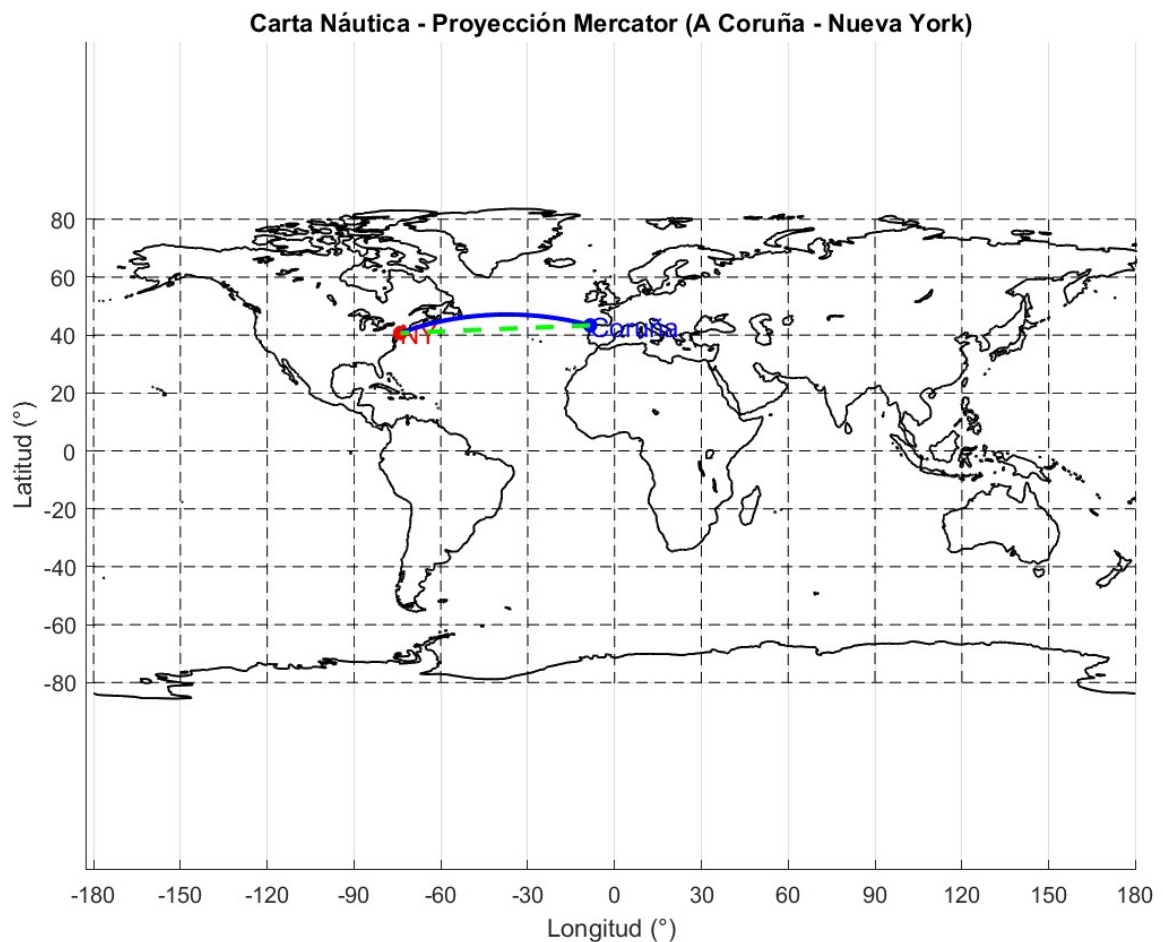


Figura 3-9: Ruta ortodrómica Coruña-Nueva York. (Fuente: Autoría Propia)

3.3.4 Ejemplo 3: Tránsito Cádiz – Recife

En la tercera navegación, simulamos el tránsito realizado entre el puerto de Cádiz de coordenadas $36^{\circ} 32' 24''$ N y $6^{\circ} 16' 48''$ W y el puerto de Recife de coordenadas $8^{\circ} 2' 24''$ S y $34^{\circ} 51' 36''$ W que ha realizado EL Juan Sebastián Elcano este año en el XCVII crucero de Guardiamarinas

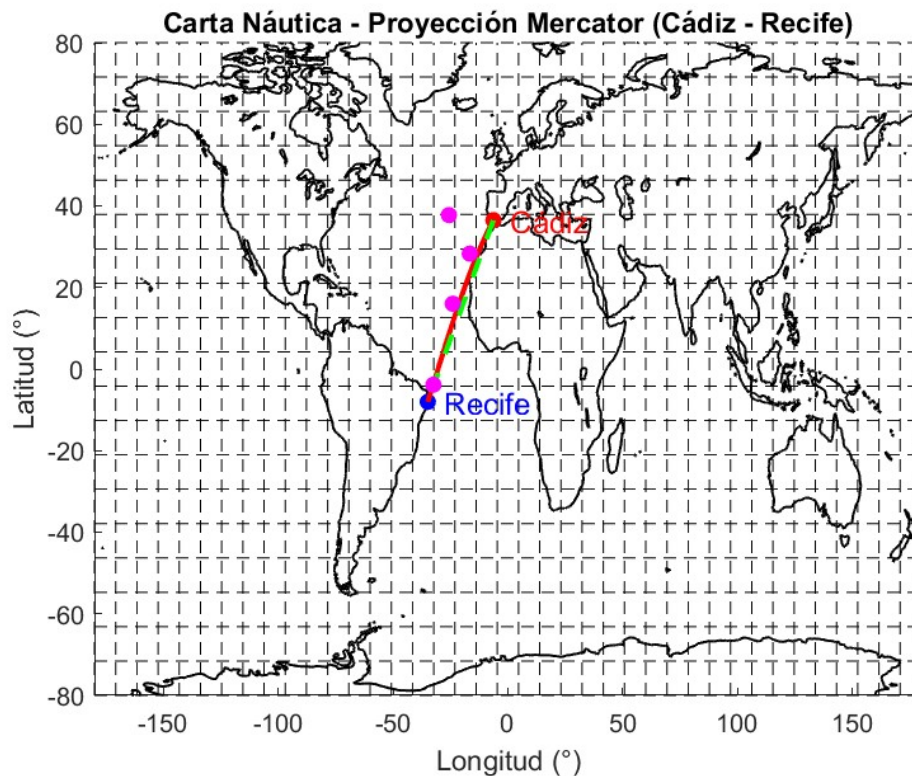


Figura 3-10: Ruta ortodrómica Cádiz-Recife. (Fuente: Autoría Propia)

En la Figura 3-10 se observa la ruta que une ambos puertos con los obstáculos indicados con puntos violetas que el buque se ha encontrado a lo largo de la travesía. Dichos obstáculos son:

- **Islas de Madeira y Portosanto:** Se incluyen tres coordenadas para acotar dichas islas.
 - **Madeira1:** $32^{\circ}39'30.69''$ N, $16^{\circ}53'34.27''$ W.
 - **Madeira2:** $32^{\circ}51'33.04''$ N, $17^{\circ}11'46.50''$ W.
 - **Madeira3:** $33^{\circ}03'17.63''$ N, $16^{\circ}19'03.35''$ W.



Figura 3-11: Islas de Madeira y Portosanto. Figura extraída de [47]

- **Archipiélago de las canarias:** Se incluyen las 8 coordenadas para acotar dichas islas:
 - **Canarias1:** 28°42'10.55"N, 17°46'05.79"W.
 - **Canarias2:** 29°07'27.59"N, 13°29'56.90"W.
 - **Canarias3:** 28°02'49.89"N, 14°19'53.19"W.
 - **Canarias4:** 27°46'01.01"N, 15°36'55.41"W.
 - **Canarias5:** 28°00'36.81"N, 16°42'55.96"W.
 - **Canarias6:** 28°05'51.62"N, 17°14'04.73"W.
 - **Canarias7:** 27°41'02.35"N, 17°59'09.50"W.



Figura 3-12: Archipiélago de las canarias. Figura extraída de [47]

- **Cabo verde:** Se incluyen las 7 coordenadas para acotar dichas islas:
 - **CaboVerde1:** 17°02'39.79"N, 25°07'29.36"W.
 - **CaboVerde2:** 16°35'59.98"N, 22°54'36.10"W.
 - **CaboVerde3:** 16°00'16.98"N, 22°44'50.84"W.
 - **CaboVerde4:** 15°09'13.76"N, 23°09'17.30"W.
 - **CaboVerde5:** 14°53'48.87"N, 23°31'24.46"W.
 - **CaboVerde6:** 16°37'03.06"N, 24°11'47.71"W.
 - **CaboVerde7:** 14°53'45.52"N, 24°27'54.25"W.



Figura 3-13: Archipiélago de Cabo Verde. Figura extraída de [47]

- **Villa dos remedios:** 3°51'06"S 32°24'49"W.



Figura 3-14: Isla de Villa dos remedios. Figura extraída de [47]

Estos puntos geográficos permiten una correcta identificación de los obstáculos naturales que podrían interferir en la planificación de la ruta marítima.

4 RESULTADOS OBTENIDOS

4.1 Ejemplo 1: Tránsito Almería-Melilla

4.1.1 Sin obstáculos

En la Figura 4-1 se presenta la discretización del área de estudio, considerando tanto paralelos como meridianos. En esta representación, el punto A corresponde al puerto de Almería, mientras que el punto B representa el puerto de Melilla. Asimismo, se muestran todos los arcos posibles que conectan los nodos dentro de la malla generada por el programa.

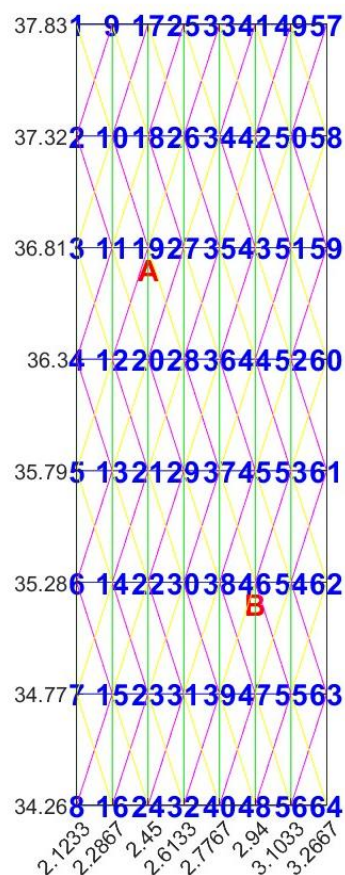


Figura 4-1: Discretización de la zona Almería-Melilla. (Fuente: Autoría Propia)

A continuación, la Figura 4-2 muestra la trayectoria óptima generada por el programa para conectar ambos puertos.

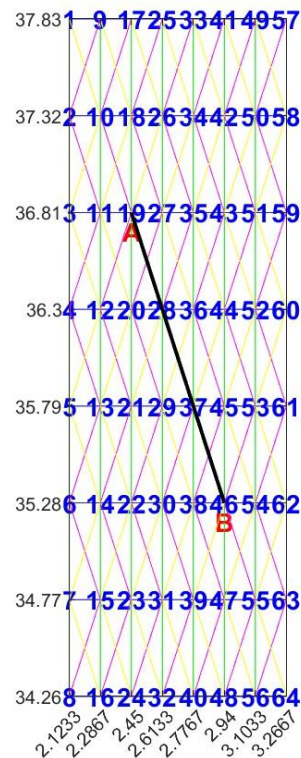


Figura 4-2: Ruta óptima entre puertos.(Fuente : Autoría Propia)

Como se mencionó previamente en este apartado, el programa no solo determina los nodos por los que transcurre la ruta óptima, sino que también proporciona las coordenadas GPS de cada nodo y calcula la distancia total a recorrer, representada por el parámetro *fval* (Apartado 3.2.2). Esta información se recoge en la Tabla 4-1:

Nodo	Latitud	Longitud	Distancia (mn)
19	36° 48' 36" N	2° 27' 0 " W	0.00
28	36° 18' 00"N	2° 36 '48" W	31.62
37	35° 47' 24" N	2° 46' 36" W	31.63
46	35° 16' 48"N	2° 56' 24" W	31.64
Total:			94.89

Tabla 4-1: Datos tránsito Almería-Melilla. (Fuente: Autoría Propia)

4.1.2 Con obstáculos

Se introduce un nuevo escenario en el que se incorporan diversos obstáculos en la ruta entre ambos puertos. Esto permitirá analizar cómo el programa ajusta la trayectoria para seleccionar la mejor alternativa que minimice la distancia total de navegación.

Para el nuevo escenario introducimos obstáculos en las coordenadas $36^{\circ}17'24''\text{N}$, $2^{\circ}36'36''\text{W}$ y $35^{\circ}54'\text{N}$, $2^{\circ}46'12''\text{W}$ dentro del modelo de optimización, se deberán identificar los nodos más cercanos a estos puntos en la malla generada y restringir el tránsito a través de ellos.

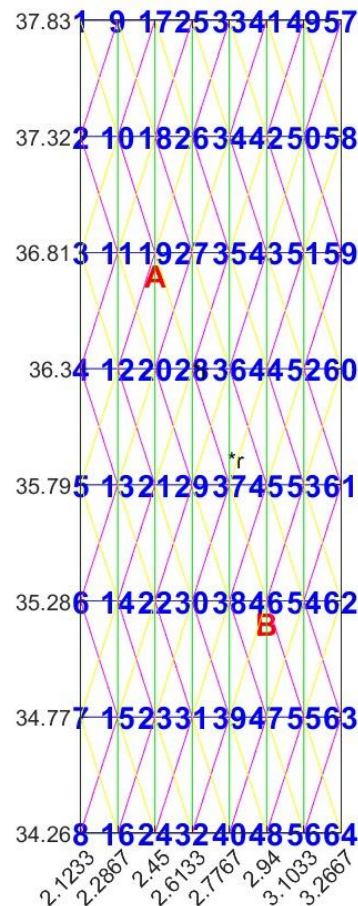


Figura 4-3: Obstáculos entre los puntos A-B. (Fuente: Autoría Propia)

En la Figura 4-3 los obstáculos están representados mediante el carácter 'r'. Se observa que uno de ellos se encuentra ubicado exactamente en el nodo 28, mientras que el otro se sitúa entre los nodos 29 y 37.

En esta fase del programa, el usuario debe ingresar manualmente los arcos de salida y entrada en el nodo, los cuales definirán las restricciones de tránsito en la malla, asegurando que los nodos afectados sean correctamente excluidos de la trayectoria óptima.

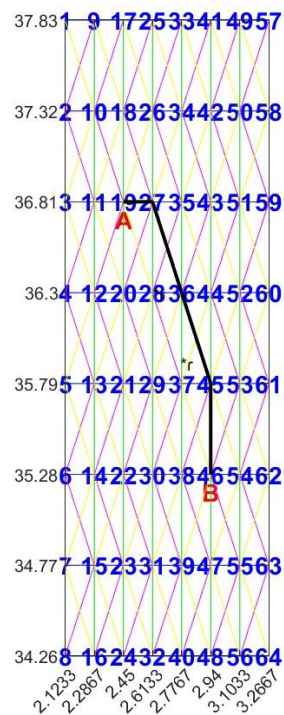


Figura 4-4: Ruta optimizada evitando los obstáculos.(Fuente: Autoría Propia)

A continuación, la Tabla 4-2 muestra los datos de la ruta optimizada Figura 4-4, evitando los obstáculos:

Nodo	Latitud	Longitud	Distancia (mn)
19	36° 48' 36" N	2° 27' 0 " W	0.00
27	36° 48' 36 "N	2° 36 '36" W	30.62
36	35° 18' 00" N	2° 46' 12" W	7.85
45	35° 47' 24" N	2° 56' 24" W	31.61
46	35° 16' 48" N	2° 56' 24" W	31.63
Total:			101.71

Tabla 4-2: Valores optimización ruta Almería-Melilla. (Fuente: Autoría Propia)

Estos valores se podrían ajustar aún más si conseguimos discretizar con valores mayores de los parámetros m y n , ya que el usuario será capaz de aproximar lo máximo posible el paso de la ruta lo más cerca posible del obstáculo sin que este interfiera.

Realizamos un ejemplo para $m = n = 6$:

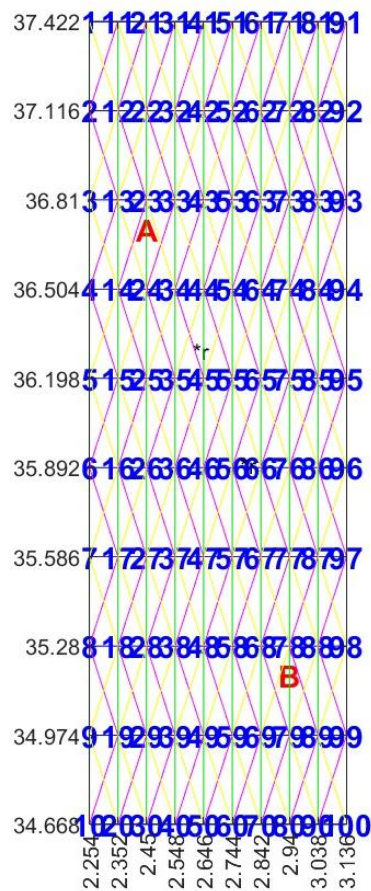


Figura 4-5: Malla 6 x 6. (Fuente: Autoría Propia)

A continuación, la Tabla 4-3 muestra los resultados aún más ajustados de la distancia:

Nodo	Latitud	Longitud	Distancia (mn)
23	36° 48' 36" N	2° 27' 00" W	0.00
33	36° 48' 36" N	2° 32' 53" W	18.37
44	36° 30' 14" N	2° 38' 46" W	4.71
55	36° 11' 53" N	2° 44' 38" W	18.97
66	35° 53' 31" N	2° 50' 31" W	18.97
77	35° 35' 10" N	2° 56' 24" W	18.98
78	35° 16' 48" N	2° 59' 24" W	18.98
Total:			98.98

Tabla 4-3: Ruta ajustada a la malla 6 x 6. (Fuente: Autoría Propia)

A partir de este punto, se prescindirá de la implementación de imágenes con nodos, como en la Figura 4-3, con el objetivo de mejorar la claridad y comprensión del contenido para el lector. Sin embargo, es importante destacar que el funcionamiento del sistema y la lógica aplicada en los ejemplos seguirán siendo exactamente los mismos.

4.2 Ejemplo 2: Tránsito Coruña – Nueva York

En este tránsito sin la presencia de obstáculos, se representa la ruta entre ambos puertos y se evidencia cómo MATLAB es capaz de trazar la trayectoria siguiendo, en la medida de lo posible, el círculo máximo, tal como se describe en la sección 3.3.1. Esto permite que la navegación siga un trayecto ortodrómico, el cual minimiza la distancia recorrida al aproximarse a la ruta más corta entre dos puntos sobre la superficie terrestre. Dicha trayectoria ortodrómica se compone de múltiples segmentos de loxodrómicas, garantizando una aproximación eficiente.

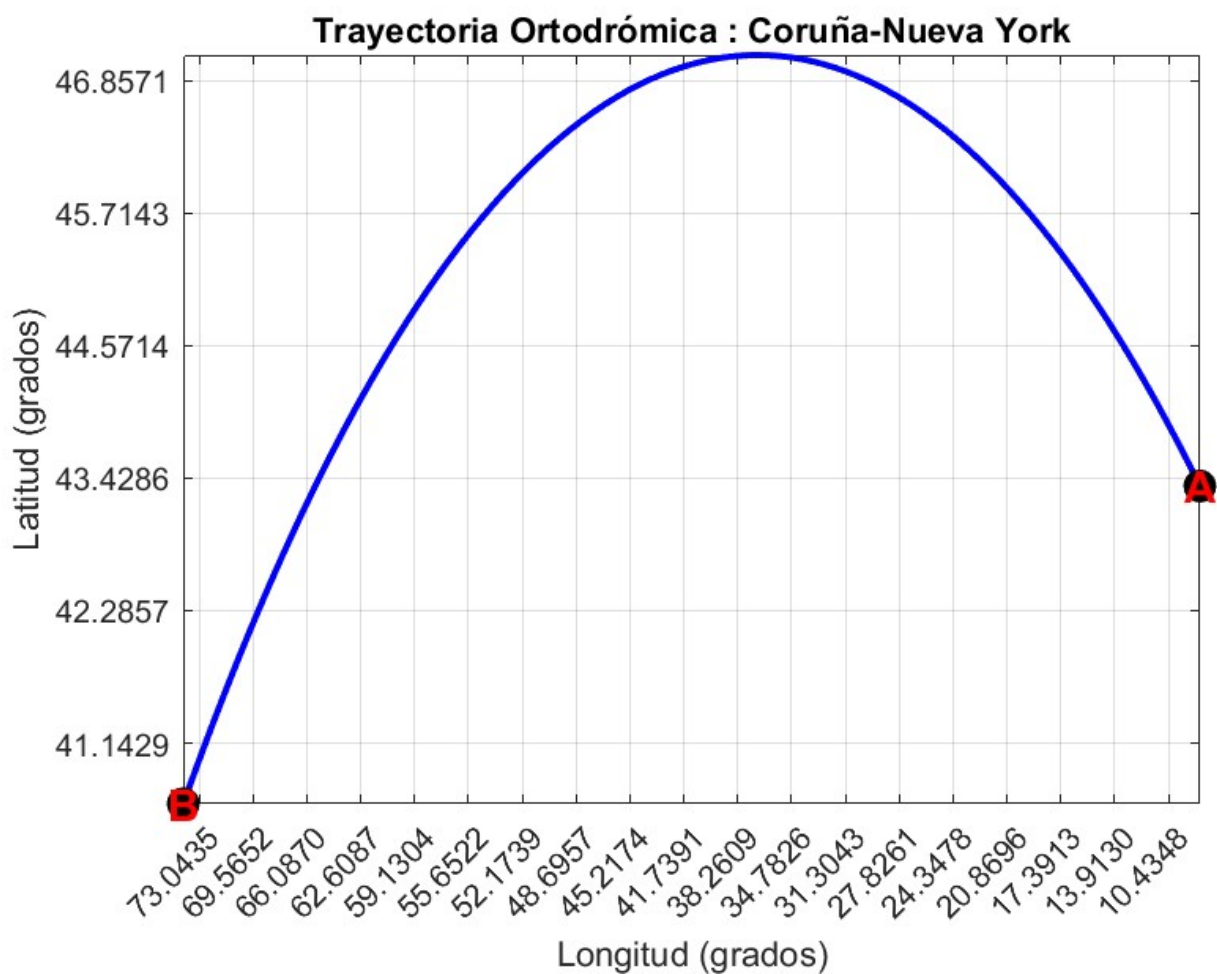


Figura 4-6: Ruta ortodrómica Coruña-Nueva York. (Fuente: Autoría Propia)

A continuación, se presenta la Tabla 4-4 con las coordenadas de los puntos por los que debe transitar la ruta, además de la distancia a navegar entre nodos:

Nodo	Latitud	Longitud	Distancia (mn)
19	43° 21' 36.0" N	8° 24' 36.0" W	0.0
26	44° 16' 23.88" N	11° 51' 56.88" W	159.79
33	45° 11' 12.12" N	15° 19' 17.76" W	157.63
41	45° 11' 12.12" N	18° 46' 38.64" W	146.64
49	45° 11' 12.12" N	22° 13' 59.52" W	146.64
57	45° 11' 12.12" N	25° 41' 20.04" W	146.63
65	45° 11' 12.12" N	29° 8' 40.92" W	146.64
73	45° 11' 12.12" N	32° 36' 1.8" W	146.64
81	45° 11' 12.12" N	36° 3' 22.68" W	146.64
89	45° 11' 12.12" N	39° 30' 43.56" W	146.64
97	45° 11' 12.12" N	42° 58' 4.44" W	146.64
105	45° 11' 12.12" N	46° 25' 25.32" W	146.64
113	45° 11' 12.12" N	49° 52' 46.2" W	146.64
121	45° 11' 12.12" N	53° 20' 7.08" W	146.64
129	45° 11' 12.12" N	56° 47' 27.96" W	146.64
138	44° 16' 23.88" N	60° 14' 48.48" W	157.63
147	43° 21' 36.0" N	63° 42' 9.36" W	159.79
156	42° 26' 48.12" N	67° 9' 30.24" W	161.92
165	41° 31' 59.88" N	70° 36' 51.12" W	164.02
174	40° 37' 12.0" N	74° 4' 12.0" W	166.09
Total:			2886.54

Tabla 4-4: Valores optimización ruta Coruña-Nueva York. (Fuente: Autoría Propia)

4.3 Ejemplo 3: Tránsito Cádiz-Recife

En este apartado se analiza la ruta transoceánica con el objetivo de evaluar el comportamiento del programa ante la presencia de obstáculos en el trayecto. A través de este caso, se pretende mostrar cómo el algoritmo de optimización identifica y traza la ruta más eficiente en un ejemplo a gran escala como este, adaptando la trayectoria para evitar zonas no transitables o restringidas, previamente definidas en el modelo.

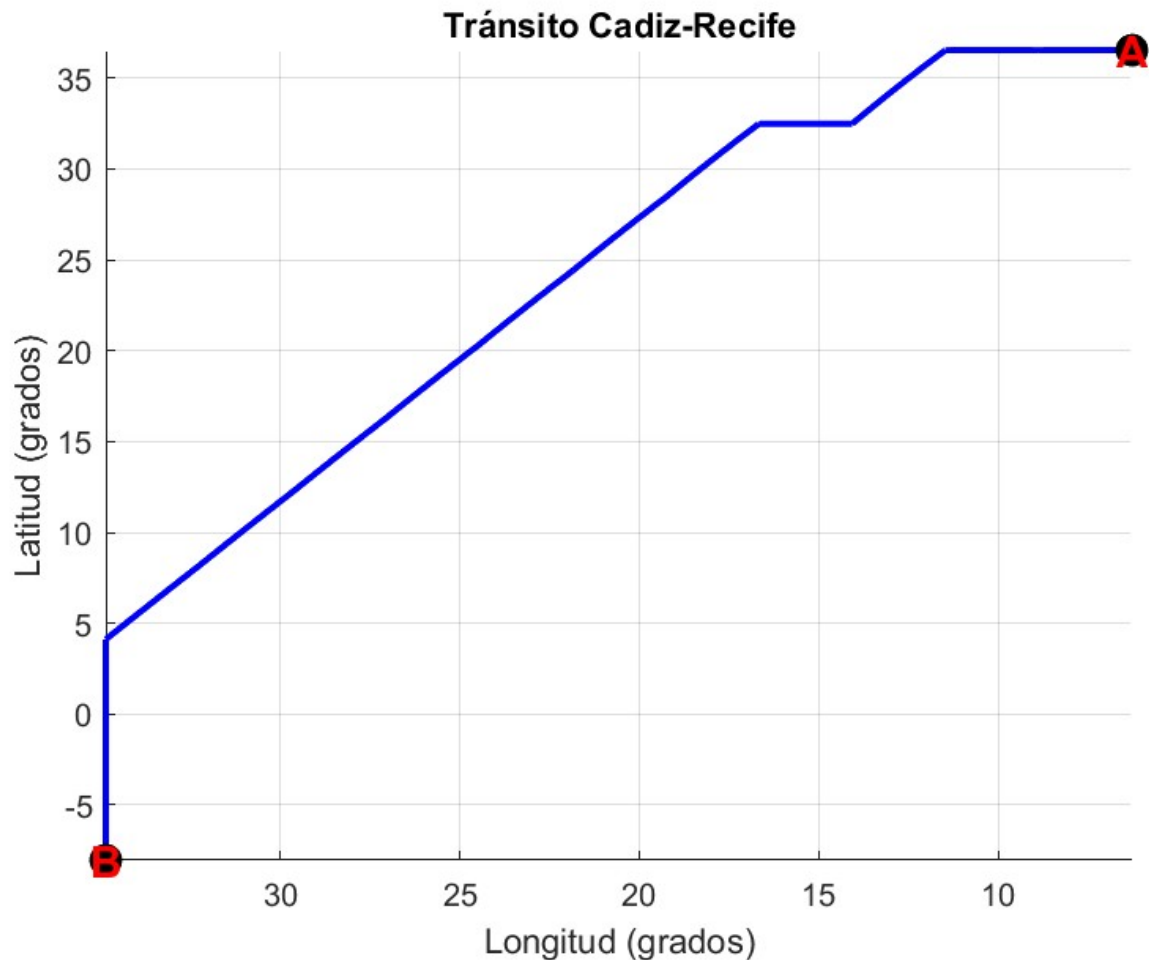


Figura 4-7: Tránsito entre Cádiz-Recife. (Fuente: Autoría Propia)

A diferencia de la Figura 4-6, donde la ruta entre Coruña y Nueva York sigue la derrota ortodrómica en la Figura 4-7 se aprecia que la trayectoria entre Cádiz y Recife no sigue el círculo máximo. Esta desviación se debe principalmente a la presencia de obstáculos definidos en el modelo. Además, se puede observar que muchos tramos de la ruta comparten latitudes o longitudes constantes, lo que implica que el programa ha optado por segmentos rectos en los ejes de coordenadas. Tal como se explica en el apartado 3.3.1.

A continuación, se presenta la Tabla 4-5 con las coordenadas de los puntos por los que debe transitar la ruta, además de la distancia a navegar entre nodos:

Nodo	Latitud	Longitud	Distancia (mn)
35	36° 32' 24" N	6° 16' 48" W	0.00
51	36° 32' 24" N	8° 52' 42" W	243.33
67	36° 32' 24" N	11° 28' 35" W	243.33
84	32° 29' 14" N	14° 4' 28" W	243.33
100	32° 29' 14" N	16° 40' 22" W	125.33
117	28° 26' 4" N	19° 16' 15" W	125.33
134	24° 22' 54" N	21° 52' 9" W	131.58
151	20° 19' 45" N	24° 28' 2" W	275.16
168	16° 16' 35" N	27° 3' 56" W	277.98
185	12° 13' 25" N	29° 39' 49" W	280.56
202	8° 10' 15" N	32° 15' 42" W	282.86
219	4° 7' 6" N	34° 51' 36" W	284.83
220	0° 3' 56" N	34° 51' 36" W	286.46
221	3° 59' 14" S	34° 51' 36" W	287.7
222	8° 2' 24" S	34° 51' 36" W	288.54
Total:			3376.32

Tabla 4-5: Datos tránsito Cádiz-Recife. (Fuente: Autoría Propia)

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

Este Trabajo Fin de Grado ha cumplido su objetivo principal: desarrollar un programa en MATLAB capaz de calcular rutas marítimas óptimas sobre una malla de nodos, utilizando programación binaria para seleccionar automáticamente la mejor trayectoria posible entre dos puntos geográficos.

El programa ha demostrado ser preciso en el cálculo de distancias, tanto en trayectos cortos como en largas travesías. En el caso de Almería–Melilla, se obtuvo correctamente la ruta más corta sin obstáculos. Por otro lado, en la ruta entre A Coruña y Nueva York, también sin obstáculos, se aplicó navegación ortodrómica (por círculo máximo), lo que permitió encontrar la distancia mínima real en un cruce transatlántico.

Una de las principales ventajas del modelo es su capacidad para adaptarse a obstáculos, como se muestra en la simulación entre Almería y Melilla con nodos bloqueados. El programa ajusta la trayectoria automáticamente y sigue ofreciendo una solución válida y eficiente. Esta flexibilidad permite simular condiciones reales como zonas restringidas o áreas a evitar.

Además, el programa ha demostrado ser escalable, funcionando correctamente en rutas de mayor alcance, como en el caso entre Cádiz y Recife, donde se resolvió una malla de mayor tamaño y se obtuvieron resultados coherentes.

En resumen, el programa ofrece una solución robusta y versátil para la planificación de rutas marítimas, tanto en entornos simples como complejos, adaptándose a obstáculos y proporcionando siempre trayectorias óptimas en distancia.

5.2 Líneas Futuras

A partir del desarrollo realizado, se identifican tres líneas principales para continuar y ampliar este trabajo:

1. Optimización energética y temporal de trayectorias

Una posible evolución del modelo es incorporar variables adicionales como la velocidad de navegación, el consumo estimado de combustible y el tiempo total del trayecto. Esto permitiría transformar la optimización actual, centrada únicamente en la distancia, en una optimización multiobjetivo, orientada a mejorar la eficiencia operativa y reducir el impacto medioambiental. Para ello, la función objetivo sería:

$$F = \lambda \cdot v^{\Omega} \left(\frac{\text{tons}}{\text{día}} \right) \quad (5-1)$$

donde v es la velocidad del buque y λ y Ω son parámetros que dependen del tipo de barco.

Dicha función (5-1) estaría supeditada a una optimización no lineal con incógnitas de variables continuas con las restricciones binarias de este trabajo (los arcos que sí o no escoge para trazar la ruta) [36].

También se podrían considerar zonas con condiciones variables (meteorología, corrientes) que influyan directamente en el coste de cada ruta.

2. Mejora del realismo y de la experiencia de uso

El modelo puede adaptarse para resolver rutas con escalas intermedias o múltiples destinos, lo que ampliaría su aplicabilidad en escenarios logísticos más complejos. Asimismo, se propone el desarrollo de una interfaz gráfica interactiva (GUI), que permita al usuario marcar puertos, definir obstáculos y visualizar en tiempo real las rutas óptimas sobre una carta náutica.

3. Ampliación del modelo y validación práctica

Se sugiere incrementar la resolución de la malla para lograr mayor precisión, especialmente en zonas costeras. Por último, sería útil comparar los resultados del programa con rutas reales obtenidas mediante sistemas AIS, con el objetivo de validar la utilidad del algoritmo en contextos operativos reales.

6 BIBLIOGRAFÍA

- [1] nautadmin, «La historia de la navegación: cómo ha evolucionado», Nautilus. Accedido: 21 de enero de 2025. [En línea]. Disponible en: <https://nautilus.es/la-historia-la-navegacion-ha-evolucionado/>
- [2] «La guerra en Ucrania dispara el precio del gas y el petróleo y agrava la crisis energética», www.elsaltodiario.com. Accedido: 21 de enero de 2025. [En línea]. Disponible en: <https://www.elsaltodiario.com/guerra-en-ucrania/dispara-precio-gas-petroleo-agrava-crisis-energetica>
- [3] E. Stratta, «El precio del crudo y su historia», 2016.
- [4] «Gases de efecto invernadero». Accedido: 22 de marzo de 2025. [En línea]. Disponible en: <https://www.imo.org/es/OurWork/Environment/Paginas/GHG-Emissions.aspx?>
- [5] «TFG_JOEL_VAL_GARCIA.pdf». Accedido: 23 de marzo de 2025. [En línea]. Disponible en: https://oa.upm.es/67562/1/TFG_JOEL_VAL_GARCIA.pdf?utm_source=chatgpt.com
- [6] Y. Liu, Z. Liu, J. Shi, G. Wu, y C. Chen, «Optimization of Base Location and Patrol Routes for Unmanned Aerial Vehicles in Border Intelligence, Surveillance, and Reconnaissance», *J. Adv. Transp.*, vol. 2019, pp. 1-13, ene. 2019, doi: 10.1155/2019/9063232.
- [7] C. Ortega, «Programación lineal: Qué es, usos y pasos para realizarla», QuestionPro. Accedido: 9 de febrero de 2025. [En línea]. Disponible en: <https://www.questionpro.com/blog/es/programacion-lineal/>
- [8] «Historia de la programación lineal», Optimus. Accedido: 9 de febrero de 2025. [En línea]. Disponible en: <https://inveoperaciones.wordpress.com/2012/05/07/historia-de-la-programacion-lineal/>
- [9] Dr. H. Arsham, *Modelos Deterministas: Optimización Lineal*. 2020.
- [10] «Tema 1.pdf». Accedido: 21 de febrero de 2025. [En línea]. Disponible en: <https://www.fdi.ucm.es/profesor/jjruiz/MasterUned/Documentos%20en%20aLF/Tema%201.pdf>
- [11] «Programación lineal». Accedido: 9 de febrero de 2025. [En línea]. Disponible en: <https://es.mathworks.com/discovery/linear-programming.html>
- [12] L. Montoya, «George Dantzig», Historia y biografía de. Accedido: 11 de febrero de 2025. [En línea]. Disponible en: <https://historia-biografia.com/george-bernard-dantzig/>

- [13] B. S. López, «Método Simplex » Programación lineal » Investigación de operaciones», Ingeniería Industrial Online. Accedido: 11 de febrero de 2025. [En línea]. Disponible en: <https://ingenieriaindustrialonline.com/investigacion-de-operaciones/metodo-simplex/>
- [14] «Un momento...». Accedido: 12 de febrero de 2025. [En línea]. Disponible en: https://www.researchgate.net/profile/Izar-Juan-Manuel/publication/329931840_SimplexProgLineal/links/655ce0beb86a1d521bfcea79/SimplexProgLineal.pdf?__cf_chl_tk=APnV2WiYAIuLwQRm1isZ1ps15UIFJXbFIfwH793GsRw-1739375838-1.0.1.1-cJcE4MCwpHoNmTlZxoJl7NHQsXDWzXqwRbUCNK10ADY
- [15] «66_13_0804200912835.pdf». Accedido: 12 de febrero de 2025. [En línea]. Disponible en: https://www.mate.unlp.edu.ar/practicas/66_13_0804200912835.pdf
- [16] D. G. H. Oliva, «Métodos Clásicos de Optimización para Problemas No-Lineales sin Restricciones».
- [17] «Vol+1%2F3.+Optimización+no+lineal.pdf». Accedido: 13 de febrero de 2025. [En línea]. Disponible en: <https://biblus.us.es/bibing/proyectos/abreproy/3880/fichero/Vol%2B1%252F3.%2BOptimizaci%C3%B3n%2Bno%2Blineal.pdf?>
- [18] «MNOmaster1.pdf». Accedido: 17 de febrero de 2025. [En línea]. Disponible en: <https://www.dma.uvigo.es/~aurea/MNOmaster1.pdf>
- [19] «Método cuasi-Newton (BFGS, DFP) para resolver problemas de optimización sin restricciones - programador clic». Accedido: 17 de febrero de 2025. [En línea]. Disponible en: <https://programmerclick.com/article/4634692937/>
- [20] «Programación no lineal: qué es y para qué sirve», VIU Colombia. Accedido: 12 de febrero de 2025. [En línea]. Disponible en: <https://www.universidadviu.com/co/actualidad/nuestros-expertos/programacion-no-lineal-que-es-y-para-que-sirve>
- [21] «Programación no lineal: qué es y para qué sirve», VIU Colombia. Accedido: 17 de febrero de 2025. [En línea]. Disponible en: <https://www.universidadviu.com/co/actualidad/nuestros-expertos/programacion-no-lineal-que-es-y-para-que-sirve>
- [22] C. de Prada, «Optimización con restricciones».
- [23] «Programacion no lineal como resolver problemas de optimizacion complejos y no lineales», FasterCapital. Accedido: 22 de febrero de 2025. [En línea]. Disponible en: <https://fastercapital.com/es/contenido/Programacion-no-lineal--como-resolver-problemas-de-optimizacion-complejos-y-no-lineales.html>
- [24] C. F. Ruiz Paredes, J. Morales Paredes, y L. E. Ruiz Paredes, «Métodos de punto interior – un nuevo método de punto interior», *Sci. Tech.*, vol. 28, n.º 3, pp. 157-164, 2023.
- [25] «content.pdf». Accedido: 23 de febrero de 2025. [En línea]. Disponible en: <https://repositorio.upct.es/server/api/core/bitstreams/81d472a6-3671-463b-aa91-9706e67f5883/content>
- [26] «Introduccion_a_la_programacion_no_lineal.pdf». Accedido: 23 de febrero de 2025. [En línea]. Disponible en: https://rua.ua.es/dspace/bitstream/10045/19734/5/Introduccion_a_la_programacion_no_lineal.pdf
- [27] A. Reyes-Figueroa, «Métodos de Región de Confianza».
- [28] «Clase14.pdf». Accedido: 10 de marzo de 2025. [En línea]. Disponible en: <https://www.uv.es/~sala/Clase14.pdf>
- [29] «intlinprog». Accedido: 10 de marzo de 2025. [En línea]. Disponible en: <https://es.mathworks.com/help/optim/ug/intlinprog.html>

- [30] I. P. Lefler y D. C. Carrera, «Modelo lineal para la minimización de costos de adquisición de materiales para organizaciones».
- [31] G. E. O. Tutoriales, «Qué es la Programación Entera», Gestión de Operaciones. Accedido: 10 de marzo de 2025. [En línea]. Disponible en: <https://www.gestiondeoperaciones.net/programacion-entera/que-es-la-programacion-entera/>
- [32] G. E. O. Tutoriales, «Ejemplo del Algoritmo de Branch and Bound (Ramificación y Acotamiento)», Gestión de Operaciones. Accedido: 10 de marzo de 2025. [En línea]. Disponible en: <https://www.gestiondeoperaciones.net/programacion-entera/ejemplo-del-algoritmo-de-branch-and-bound-ramificacion-y-acotamiento/>
- [33] «>Qué es Matlab y para qué sirve | Características, aplicaciones y funciones». Accedido: 10 de marzo de 2025. [En línea]. Disponible en: <https://deingenierias.com/software/matlab/>
- [34] «Optimization Toolbox». Accedido: 10 de febrero de 2025. [En línea]. Disponible en: <https://es.mathworks.com/help/optim/index.html>
- [35] «linprog». Accedido: 10 de febrero de 2025. [En línea]. Disponible en: <https://es.mathworks.com/help/optim/ug/linprog.html>
- [36] J. I. R. Montañés y D. C. Carrera, «Modelo de optimización no lineal para la obtención de rutas de navegación óptimas con mínimo consumo de combustible.».
- [37] «intlinprog». Accedido: 10 de marzo de 2025. [En línea]. Disponible en: <https://es.mathworks.com/help/optim/ug/intlinprog.html>
- [38] «shortestpath». Accedido: 10 de marzo de 2025. [En línea]. Disponible en: <https://es.mathworks.com/help/matlab/ref/graph.shortestpath.html>
- [39] «particleswarm». Accedido: 10 de marzo de 2025. [En línea]. Disponible en: <https://es.mathworks.com/help/gads/particleswarm.html>
- [40] A. Munín-Doce, «Optimización de flotas de portacontenedores para rutas multimodales en la costa atlántica europea», 2022, Accedido: 23 de febrero de 2025. [En línea]. Disponible en: <https://ruc.udc.es/dspace/handle/2183/33307>
- [41] K. Y. T. Huerta *et al.*, «Programación lineal aplicada a la optimización de rutas de transporte en SXR-POLYMERS», *Interconectando Saberes*, n.º 18, Art. n.º 18, doi: 10.25009/is.v0i18.2889.
- [42] «Smart Shipping: Optimización de Rutas Marítimas | PDF», Scribd. Accedido: 24 de febrero de 2025. [En línea]. Disponible en: <https://es.scribd.com/document/658501470/SMART-SHIPPING-OPTIMIZACION-DE-RUTAS-MARITIMAS>
- [43] J. Mandal, A. Goswami, L. Thakur, y M. K. Tiwari, «Simultaneous Planning of Liner Ship Speed Optimization, Fleet Deployment, Scheduling and Cargo Allocation with Container Transshipment», 21 de julio de 2023, *arXiv*: arXiv:2307.11583. doi: 10.48550/arXiv.2307.11583.
- [44] «Fórmula de Haversina _ AcademiaLab». Accedido: 12 de marzo de 2025. [En línea]. Disponible en: <https://academia-lab.com/enciclopedia/formula-de-haversina/>
- [45] M. Pastor, «Náutica», Náutica. Accedido: 16 de marzo de 2025. [En línea]. Disponible en: <https://fomalhaut-mp.blogspot.com/2013/09/calculo-ortodromica.html>
- [46] J. I. R. Montañés y D. C. Carrera, «Modelo de optimización no lineal para la obtención de rutas de navegación óptimas con mínimo consumo de combustible.».
- [47] «Introducción – Google Earth», Google Earth. Accedido: 24 de marzo de 2025. [En línea]. Disponible en: <https://www.google.com/intl/es/earth/>

ANEXO I: IMPLICACIONES SOCIALES, Y/O ECONÓMICAS, Y/O AMBIENTALES

La optimización de rutas marítimas con MATLAB emplea modelos avanzados para trazar trayectos eficientes, considerando factores como clima, corrientes y tráfico. Esto reduce distancia, tiempo y consumo de combustible, mejorando la eficiencia operativa. Además, tiene impactos sociales, económicos y ambientales, beneficiando el comercio global y la sostenibilidad marina.

- **Implicaciones sociales**

Las comunidades costeras se benefician de rutas marítimas optimizadas, ya que la reducción del consumo de combustible disminuye las emisiones contaminantes en puertos y costas, mejorando la calidad del aire y la salud pública. Además, al evitar zonas peligrosas y condiciones extremas, se reducen los riesgos de accidentes marítimos que podrían afectar la pesca, el turismo y la vida cotidiana en estas áreas. Para los trabajadores marítimos, estas mejoras significan una mayor seguridad y menor estrés, ya que las rutas optimizadas esquivan temporales y zonas de alto tráfico, permitiendo travesías más predecibles y seguras.

Por otro lado, la digitalización y optimización de rutas están transformando la industria naviera, creando demanda de nuevos perfiles profesionales con competencias tecnológicas. Ahora, puertos y navieras requieren analistas de datos y especialistas en logística que interpreten información en tiempo real para ajustar rutas con mayor eficiencia. Además, crecen los empleos enfocados en sostenibilidad, como consultores ambientales que garantizan el cumplimiento de regulaciones y la reducción del impacto ecológico. A largo plazo, estos avances pueden mejorar las condiciones laborales generales, fortaleciendo la seguridad y calidad de vida de quienes trabajan en el sector marítimo

- **Implicaciones económicas**

La optimización de rutas reduce significativamente los costos operativos de las navieras al disminuir el consumo de combustible, minimizar escalas innecesarias y evitar retrasos. Algoritmos avanzados, como los implementados en MATLAB, permiten ajustar la velocidad y trayectoria de los buques en tiempo real, aprovechando corrientes favorables y evitando condiciones adversas. Esto no solo ahorra millones de dólares anuales, sino que también optimiza el uso de recursos y reduce costos logísticos por tonelada transportada. Además, la mejora en la planificación disminuye gastos adicionales como tarifas portuarias y sobrecostos por demoras, haciendo más eficiente toda la cadena de suministro.

A nivel global, la optimización de rutas fortalece el comercio internacional al agilizar el flujo de mercancías y mejorar la competitividad de las navieras. Al reducir costos y garantizar mayor puntualidad, las empresas pueden ofrecer tarifas más competitivas y mejorar la fiabilidad de las entregas, beneficiando a exportadores e importadores. Esto impacta en los precios finales de los bienes transportados, ya que menores costos logísticos pueden traducirse en productos más accesibles para los consumidores. Además, al abaratar el transporte marítimo, se contribuye a contener la inflación y fortalecer la economía, favoreciendo tanto a empresas como a consumidores en un mercado cada vez más globalizado

- **Implicaciones ambientales:**

La optimización de rutas marítimas reduce el consumo de combustible y las emisiones de CO₂, disminuyendo el impacto climático del transporte y ayudando a cumplir regulaciones ambientales. Al encontrar trayectos más cortos y condiciones favorables, los buques consumen menos carburante y emiten menos contaminantes atmosféricos como SO₂ y NO_x, mejorando la calidad del aire en zonas costeras. Además, la planificación inteligente contribuye a reducir la contaminación acústica submarina, protegiendo la fauna marina sensible. También ayuda a evitar ecosistemas frágiles y minimiza riesgos

de accidentes ecológicos, como derrames de petróleo. Así, la optimización de rutas equilibra eficiencia económica y sostenibilidad ambiental, favoreciendo un transporte marítimo más responsable.

ANEXO II: REFLEXIONES ÉTICAS Y SOCIALES

En primer lugar, el uso de modelos de optimización aplicados a la navegación marítima contribuye directamente a una mejor planificación del transporte, permitiendo reducir trayectorias innecesarias, tiempos de navegación y, en consecuencia, el consumo de combustible. Esta mejora en la eficiencia está alineada con los objetivos de sostenibilidad promovidos a nivel internacional, como los recogidos en la Agenda 2030 de la ONU. Minimizar distancias también implica reducir la huella de carbono del transporte marítimo, un sector responsable de cerca del 3 % de las emisiones globales de gases de efecto invernadero.

Desde una perspectiva social, el programa desarrollado podría servir como base para sistemas de ayuda a la toma de decisiones tanto en entornos comerciales como en aplicaciones estratégicas, como la planificación de rutas navales seguras en contextos de riesgo. El hecho de poder esquivar zonas peligrosas, de exclusión o con conflictos activos tiene una dimensión ética importante, al priorizar la seguridad de las tripulaciones y la protección del medio marino.

ANEXO III: CÓDIGO DEL PROGRAMA

```
% Driver para correr el problema de trayectoria de mínima distancia
prompt= " valor de latitud 1 en grados: "
lat1=input(prompt);
prompt= " valor de longitud 1 en grados: "
long1=input(prompt);
prompt= " valor de latitud 2 en grados: "
lat2=input(prompt);
prompt= " valor de longitud 2 en grados: "
long2=input(prompt);
prompt= "Número de filas (n) de la discretizacion entre A y B: "
n = input(prompt);
prompt= "Número de columnas (m) de la discretizacion entre A y B: "
m = input(prompt);
%obs=1 tiene obstaculos, obs=0 no tiene obstaculos
obs=1;
% distancia entre los puntos A y B
R = 6371; % Radio de la Tierra en km
R=R*0.539957; %radio en millas nauticas
deltateta = lat2 - lat1;
deltalambda = long2 - long1;
a = (sind(deltateta/2))^2 + cosd(lat1) * cosd(lat2) * (sind(deltalambda/2))^2;
distAB = 2 * R * asin(sqrt(a))
%Creando la malla de puntos (nodos) del espacio marítimo (x(i),y(j)), done i=1,...,n y
%j=1,...,m
y(1)=lat1;
if n == 1
    vdis=(lat2+lat1)/2;
else
    vdis = (lat2-lat1)/(n-1)
    for j=1:n-2
        y(j+1) = y(j)+vdis;
    end
    y(n)=lat2;
end
x(1)=long1;
if m ==1
    hdis=(long2+long1)/2;
else
    hdis=(long2-long1)/(m-1)
    for i=1:m-1
        x(i+1)= x(i)+hdis;
    end
    x(m)=long2;
end
x;
y;
%creando la malla exterior con cuatro filas extras y 4 columnas extras
nn=n+4;
mm=m+4;
longq1=long1-2*hdis;
latq1=lat1-2*vdis;
longqf=long2+2*hdis;
latqf=lat2+2*vdis;
hhdis=(longqf-longq1)/(mm-1)
vvdis=(latqf-latq1)/(nn-1)
ymod(1)=latq1;
for j=1:nn-2
```

```
    ymod(j+1) = ymod(j)+vvdis;
end
ymod(nn)=latqf;
xmod(1)=longq1;
for i=1:mm-1
    xmod(i+1)= xmod(i)+hhdis;
end
xmod(mm)=longqf;
% xmod
% ymod
% mm
%nn
for j=1:mm
    for i=1:nn
        xx((j-1)*nn+i)=xmod(j);
        yy((j-1)*nn+i)=ymod(i);
    end
end
xx;
yy;
if long1 < long2
    if lat1 < lat2
        inicial=[nn-2;3];
        final=[3;mm-2];
    end
    if lat1 > lat2
        inicial=[3;3];
        final=[nn-2;mm-2];
    end
    if lat1 == lat2
        inicial=[3;3];
        final=[3;mm-2];
    end
end
if long1 > long2
    if lat1 < lat2
        inicial=[nn-2;mm-2];
        final=[3;3];
    end
    if lat1 > lat2
        inicial = [3;mm-2];
        final=[nn-2;3];
    end
    if lat1 == lat2
        inicial=[3;mm-2];
        final=[3;3];
    end
end
if (long1 == long2)
    if (lat1 > lat2)
        inicial=[3;3];
        final=[nn-2;3];
    end
    if lat1< lat2
        inicial = [nn-2;3];
        final=[3;3];
    end
end
[x1,y1,AA]=grafica_malla(xx,yy,long1,lat1,long2,lat2,nn,mm,hdis,vdis)
pause
dimen=4*nn*mm-3*mm-3*nn+2;
```

```
%INTRODUCCIÓN DE ARCOS NO TRANSITABLES:
if obs == 1
    nodsalnotran=[];
    nodentnotran=[];
    nume=length(nodsalnotran);
    [dd,Aeq,beq,M]=inicializador(nn,mm,xx,yy,dimen,R,nodsalnotran,nodentnotran,nume);
else
    [dd,Aeq,beq,M]=inicializador2(nn,mm,xx,yy,dimen,R);
end

%restricciones correspondientes a cotas en las variables
lb=zeros(2*dimen,1);
ub=ones(2*dimen,1);
A=[];
b=[];
intcon=1:2*dimen;
f=dd;
display("la longitud de dd=f es")
length(dd)
length(f)
display("tamaño de la matriz Aeq")
size(Aeq)

% -----
% llamada al optimizador si solo se verifican las restricciones básicas
% del problema
%[z,fval,exitflag,output]=intlinprog(f,intcon,A,b,Aeq,beq,lb,ub,z0)
[z,fval,exitflag,output]=intlinprog(f,intcon,A,b,Aeq,beq,lb,ub);

% Imprimiendo los resultados obtenidos
vector = find(z == 1);

display("índices de la posiciones igual 1 en S")
vector
display('la suma de los elementos de la solución es')
sum(z)
nu=length(vector); %número de nodos en la trayectoria óptima
for k=1:nu
    [nodo_sal(k), nodo_ent(k)] = find(M == vector(k));
end

for k=1:nu
    display("arcos en la solución óptima")
    display("nodo")
    k
    nodo_sal(k)
    nodo_ent(k)
end
display('vector con nodos de salida')
nodo_sal
display('vector con nodos de entrada')
nodo_ent
display('nodo puerto inicial')
nodo_inicial= (inicial(2)-1)*nn+inicial(1)
display('nodo puerto final')
nodo_final=(final(2)-1)*nn+final(1)

pos = find(nodo_sal == nodo_inicial);
% Inicializar el nuevo vector con el primer valor igual al nodo del puerto
```

```
% inicial
nuevo_vector = [nodo_inicial];

%while ~isempty(pos) % Continuar mientras haya una posición válida
for i=1:nu
    valor_siguiente = nodo_ent(pos); % Obtener el valor en nodo_sal en la posición
    encontrada
    nuevo_vector = [nuevo_vector, valor_siguiente]; % Agregar al vector

    % Buscar el índice de este valor en nodo_ent
    pos = find(nodo_sal == valor_siguiente);
end
%C = union(nodo_sal, nodo_ent)
nC=length(nuevo_vector);

for k=1:nC
[filas(k), columnas(k)] = find(AA == nuevo_vector(k));
end
for k=1:nC
    y2(k)=y1(filas(k));
    x2(k)=x1(columnas(k));
end
display('fila, columna del puerto inicial')
inicial
display('fila, columna del puerto final')
final

display('posiciones x de los arcos solución')
x2
display('posiciones y de los arcos solución')
y2

plot(x2,y2,'k','LineWidth', 2)
set(gca, 'YDir', 'normal')
display("el valor de la función en el óptimo")
fval
pause

%activar si se quiere esa grafica para círculo máximo:
%Graficar la trayectoria ortodrómica pasando por todos los puntos de la ruta
figure;
hold on
title('Trayectoria Ortodrómica - Círculo Máximo por nodos óptimos')
xlabel('Longitud (grados)')
ylabel('Latitud (grados)')

% Interpolación entre cada par de nodos consecutivos
num_puntos = 100; % Número de puntos entre cada par para suavidad
for i = 1:length(x2)-1
    % Convertir a radianes
    lat1_rad = deg2rad(y2(i));
    long1_rad = deg2rad(x2(i));
    lat2_rad = deg2rad(y2(i+1));
    long2_rad = deg2rad(x2(i+1));

    % Interpolación sobre la esfera
    t = linspace(0, 1, num_puntos);
    Bx = cos(lat1_rad) * cos(long1_rad) * (1 - t) + cos(lat2_rad) * cos(long2_rad) * t;
    By = cos(lat1_rad) * sin(long1_rad) * (1 - t) + cos(lat2_rad) * sin(long2_rad) * t;
    Bz = sin(lat1_rad) * (1 - t) + sin(lat2_rad) * t;
```

```

% Normalizar
norma = sqrt(Bx.^2 + By.^2 + Bz.^2);
Bx = Bx ./ norma;
By = By ./ norma;
Bz = Bz ./ norma;

% Convertir de vuelta
lat = rad2deg(asin(Bz));
long = rad2deg(atan2(By, Bx));
long = mod(long + 180, 360) - 180;

% Dibujar segmento ortodrómico
plot(long, lat, 'b', 'LineWidth', 2)
end

% Dibujar puntos de inicio y fin
plot(x2(1), y2(1), 'ko', 'MarkerFaceColor', 'k', 'MarkerSize', 10)
plot(x2(end), y2(end), 'ko', 'MarkerFaceColor', 'k', 'MarkerSize', 10)
text(x2(1), y2(1), 'A', 'FontSize', 14, 'Color', 'r', 'FontWeight', 'bold',
'HorizontalAlignment', 'center')
text(x2(end), y2(end), 'B', 'FontSize', 14, 'Color', 'r', 'FontWeight', 'bold',
'HorizontalAlignment', 'center')

% Ajuste de ejes
set(gca, 'YDir', 'normal')
set(gca, 'XDir', 'reverse')

axis tight
grid on

%DRIVER PARA GENERAR EL MALLADO:
function[x1,y1,AA] = grafica_malla(xx,yy,long1,lat1,long2,lat2,n,m,hdis,vdis)
% n = 6; % Tamaño de la matriz
% m=7;
[I, J] = meshgrid(1:n, 1:m);
AA = (J-1)*n+I % Matriz donde cada posición tiene el valor i+j
AA=AA';
AA=flipud(AA),
% Mostrar matriz como imagen
%imagesc(A);
image(AA);
%colormap(jet); % Cambia los colores según los valores
%colorbar; % Barra de colores
axis equal;
%axis([0.5 m+0.0.5 n+0.5]); % Ajustar ejes
axis([xx(1) xx(n*m) yy(n*m) yy(1)]); % Ajustar ejes
xticks(xx(1):hdis:xx(n*m));
%tene cuidado con el vdis, si es negativo debe restarse y si vdis es
%positivo de dejarse en positivo
yticks(yy(n*m):-vdis:yy(1))
set(gca, 'YDir', 'normal')
grid on;
hold on;
% hdis;
% vdis;

x1=xx(1):hdis:xx(n*m);
%y1=yy(n*m):-vdis:yy(1);
y1=yy(1):vdis:yy(n*m)

```

```
y1=flip(y1);
% Añadir texto con valores en cada celda
for i = 1:n
    for j =1:m
        text(x1(j), y1(i), num2str(AA(i, j)), 'FontSize', 14, 'Color',
'b','HorizontalAlignment', 'center', 'FontWeight', 'bold');
    end
end
% colocando los puertos de salida y entrada
text(long1,lat1-0.1,'A','FontSize', 14, 'Color', 'r','HorizontalAlignment', 'center',
'FontWeight', 'bold');
text(long2,lat2-0.1,'B','FontSize', 14, 'Color', 'r','HorizontalAlignment', 'center',
'FontWeight', 'bold');

% COORDENADAS EN GRADOS DE LOS OBSTACULOS
b= [] ; %latitud obstaculo
a= [] ; %longitud obstáculo
for i=1:length(a)
    text(a(i),b(i),'*r')
end
%imprimiendo arcos verticales
for j=1:m
    aa(1:n,j)=x1(j)*ones(n,1);
    bb(1:n,j)=y1;
    plot(aa(1:n,j),bb(1:n,j),'g')
end
%Imprimiendo arcos horizontales
for i=1:n
    aa1(1:m,i)=x1;
    bb1(1:m,i)=y1(i)*ones(m,1);
    plot(aa1(1:m,i),bb1(1:m,i),'b')
end
aa2=zeros(n,m-1);
for j=1:m-1
    aa2(1:2:n,j)=x1(j);
    aa2(2:2:n,j)=x1(j+1);
    bb2(1:n,j)=y1;
    plot(aa2(1:n,j),bb2(1:n,j),'m')
end
aa3=zeros(n,m-1);
for j=1:m-1
    aa3(2:2:n,j)=x1(j);
    aa3(1:2:n,j)=x1(j+1);
    bb3(1:n,j)=y1;
    plot(aa3(1:n,j),bb3(1:n,j),'y')
end

%title('Matriz con valores i+j');
```