Optimización de la Trayectoria de Ascenso de Aeronaves A320 mediante Reducción de Dimensionalidad y Computación Cuántica

Trabajo de grado para optar al título de

Magíster en Física

UNIVERSIDAD
DE ANTIOQUIA
1 8 0 3

Marco A. Erazo

Director:

Dr. Leonardo A. Pachón

Universidad de Antioquia
Facultad de Ciencias Exactas y Naturales
Medellín, Colombia

Diciembre 2024

Agradecimientos

Agradezco a mis padres Isabel C. Rosero, Guillermo H. Erazo y a mi familia por estar siempre presentes y ser la principal fuerza de mi vida.

Especial agradecimiento a mi amor Ángela A. Figueroa por su apoyo en este proyecto.

Agradezco al profesor Leonardo A. Pachón y a la Universidad de Antioquia por su disposición y por darme esta valiosa oportunidad.

Abstract

In this work, the optimization problem of the ascent trajectory of the A320 aircraft is addressed; it begins with the original formulation presented by Airbus in the public document *Airbus Quantum Computing Challenge* [1] published for research groups and companies. The problem presents a quantity ϕ to be minimized that is composed of a series of highly non-linear equations which in turn depend on multiple independent unknowns. The problem is initially analyzed as a single coupled system dependent on 364 variables; the possibility of simplification is demonstrated through the analysis of the Jacobian of the system of dynamic equations, achieving a significant reduction; a fully analytical formulation of the reduced function ϕ, dependent on 104 independent variables, is obtained. The success in the reduction allows for the subsequent approaches and to fully implement it in a single script for the computational work. The one obtained is a highly non-linear function in continuous variables; to be handled in quantum computing, a Quadratic Unconstrained Binary Optimization (QUBO) model is constructed for the function in a reduced number of variables; the analogous function in binary variables contains the information of the original function but is of a different mathematical nature. A comparison is made between the behaviors of a classical optimizer (differential evolution) and the quantum method (quantum annealing) for each function, both in continuous variables, as well as in discrete variables for the case of the constructed function. Finally, a comparison of different measured parameters is shown that accounts for the strength of the quantum perspective due to its intrinsic working principle of searching for a global optimum; the weakness of the classical optimizer is exposed, showing a wide dispersion reflecting its systematic fall into local minima. Furthermore, an alternate optimization made with classical methods of the full 104 continuous variable function is described, concluding with the postulation of a candidate for the global optimum of the general problem posed.

Resumen

En este trabajo se aborda el problema de optimización de la trayectoria de ascenso de la aeronave A320, se inicia con el planteamiento original presentado por Airbus en el documento público *Airbus Quantum Computing Challenge* [1] publicado para grupos de investigación y empresas. El problema presenta una cantidad ϕ a minimizar que se compone de una serie de ecuaciones altamente no lineales que a su vez dependen de múltiples incógnitas independientes. El problema inicialmente se analiza como un solo acoplado dependiente de 364 variables; se demuestra la posibilidad de simplificación mediante el análisis del Jacobiano del sistema de ecuaciones dinámicas, logrando una reducción significativa, se consigue una formulación plenamente analítica de la función ϕ reducida y dependiente de 104 variables independientes. El éxito en la reducción permite los planteamientos posteriores y plasmarla plenamente en un solo script para el trabajo computacional. La obtenida es una función en variables continuas altamente no lineal; para manejarse en computación cuántica se construye un modelo de Optimización Cuadrática Binaria sin Restricciones (QUBO) para la función en un número de variables reducidas; la función análoga en variables binarias contiene la información de la función original pero de naturaleza matemática diferente. Se hace una comparación entre los comportamientos de optimizadores clásico (evolución diferencial) y el método cuántico (recocido cuántico) para cada función, tanto en variables continuas, como en variables discretas para el caso de la función construida. Finalmente se muestra una comparativa de diferentes parámetros medidos que dan cuenta de la fortaleza de la perspectiva cuántica por su principio de funcionamiento intrínseco de búsqueda de un óptimo global, se expone la debilidad del optimizador clásico mostrando una amplia dispersión reflejando su caída sistemática en mínimos locales. Se describe además una optimización alterna hecha con métodos clásicos de la función completa 104 variables continúas, finalizando con la postulación de un candidato a óptimo global del problema general planteado.

Índice general

Lista de Símbolos y Abreviaturas

Abreviatura	Descripción
Aeronáutica y Mecánica de Vuelo	
A320	Familia de aeronaves comerciales de Airbus.
ATC	Air Traffic Control (Control de Tráfico Aéreo).
CAS	Calibrated Airspeed (Velocidad Calibrada del Aire).
CI	Cost Index (Índice de Costo). Relaciona el costo del tiempo con el costo del combustible.
FMS	Flight Management System (Sistema de Gestión de Vuelo).
ISA	International Standard Atmosphere (Atmósfera Estándar Internacional).
MCL	Maximum Climb (Empuje Máximo de Ascenso).
MMO	Maximum Operating Mach (Número de Mach Máximo Operativo).
TAS	True Airspeed (Velocidad Verdadera del Aire).
VMO	Maximum Operating Velocity (Velocidad Máxima Operativa).
Optimización y Computación	
BQM	Binary Quadratic Model (Modelo Cuadrático Binario).
DE	Differential Evolution (Evolución Diferencial). Un algoritmo de optimización clásico.
GP	Gaussian Process (Proceso Gaussiano).

– Continuación de la Lista de Abreviaturas y Símbolos –

Abreviatura	Descripción
KPI	Key Performance Indicator (Indicador Clave de Rendimiento).
LLM	Large Language Model (Modelo Lingüístico Grande).
NLP	Non-Linear Programming (Programación No Lineal).
QAOA	Quantum Approximate Optimization Algorithm (Algoritmo Cuántico de Optimización Aproximada).
QUBO	Quadratic Unconstrained Binary Optimization (Optimización Cuadrática Binaria sin Restricciones).
SVD	Singular Value Decomposition (Descomposición en Valores Singulares).
VQE	Variational Quantum Eigensolver (Solucionador Cuántico Variacional de Eigenvalores).

Símbolos Matemáticos y Físicos

Cx_0	Coeficiente de resistencia aerodinámica sin sustentación.
C_D	Coeficiente de resistencia aerodinámica.
Cz	Coeficiente de sustentación aerodinámica.
$F_{N_{MCL}}$	Fuerza de empuje máximo de ascenso.
F_L	Fuerza de sustentación.
g_0	Aceleración estándar de la gravedad.
k	Coeficiente de la polar aerodinámica (resistencia inducida).
L	Distancia total de la trayectoria.
m_i	Masa de la aeronave en el punto i.
Q	Matriz que define un problema QUBO.
S_{REF}	Superficie de referencia aerodinámica.
v_i	Velocidad verdadera (TAS) de la aeronave en el punto i.
Zp_i	Altitud de presión en el punto i.
α	Ángulo de ataque.

– Continuación de la Lista de Abreviaturas y Símbolos –

Abreviatura	**Descripción**
γ_i	Ángulo de la trayectoria de vuelo en el punto i.
η	Consumo específico de combustible.
λ_i	Fracción del empuje máximo utilizado en el punto i.
ϕ	Función de costo a minimizar.
ρ	Densidad del aire.

Capítulo 1

Introducción

Contexto y explicación del problema

La optimización de trayectorias de vuelo constituye un problema central para la industria aeronáutica, con implicaciones en la eficiencia operativa, el impacto ambiental y la rentabilidad de las aerolíneas. El aumento en la demanda de transporte aéreo y la necesidad de disminuir las emisiones de gases de efecto invernadero minimizando el consumo de combustible, han fomentado la investigación y el desarrollo en esta área. El concurso *Airbus Quantum Computing Challenge* del fabricante Airbus [1] ejemplifica este enfoque. Dicho concurso abordó, entre otros, la optimización de la fase de ascenso para aeronaves de la familia A320, cuestión que es la base del estudio académico aquí desarrollado.

La optimización de trayectorias de vuelo consiste fundamentalmente en determinar la configuración de variables para alcanzar un objetivo definido, como la minimización del consumo de combustible, del tiempo de vuelo, o una combinación métrica, cumpliendo un conjunto de restricciones operativas y de seguridad [2]. Estas restricciones comprenden límites de velocidad, altitud, ángulo de ascenso, aceleración y potencia del motor, entre otras variables que definen el estado y comportamiento de la aeronave [3].

La fase de ascenso, en particular, es de especial interés por su alto consumo de combustible en comparación con otras fases del vuelo, como el crucero [4]. La optimización de esta fase puede conducir a reducciones sustanciales en el consumo total de combustible y, en consecuencia, en las emisiones contaminantes [5]. Además, optimizar el ascenso puede resultar en una menor duración del vuelo, lo que a su vez puede mejorar la eficiencia de la red de rutas de una aerolínea y reducir los costos operativos [6].

El problema de optimizar la trayectoria de ascenso de una aeronave es complejo debido a la naturaleza no lineal de las ecuaciones que gobiernan la dinámica del vuelo [7], la gran cantidad de variables involucradas [8], y las múltiples restricciones operativas y de seguridad que deben ser consideradas [9]. Los métodos clásicos de optimización, como la programación lineal y no lineal, a menudo encuentran dificultades para resolver este tipo de problemas de manera eficiente, especialmente cuando se trata de un gran número de variables y restricciones [10].

En este contexto, el concurso de Airbus expone un planteamiento matemático, en términos de expresiones no están cohesionadas en una expresión final, para la optimización de la fase de ascenso de las aeronaves A320. Este modelo, que se describe en detalle en la sección de Análisis del Problema Original, considera una serie de variables que definen el estado de la aeronave en cada punto de una trayectoria discretizada en N partes, incluyendo la velocidad, el ángulo de ascenso, la masa, la distancia recorrida, el coeficiente de sustentación y la fracción del empuje máximo de ascenso [1]. El objetivo es minimizar una función de costo que combina el consumo de combustible y el tiempo de vuelo, ponderados por un "Índice de Costo"(Cost Index, CI) [11], sujeto a las ecuaciones de movimiento de la aeronave y a un conjunto de restricciones operativas que incluyen límites en la velocidad calibrada del aire (CAS), el número de Mach, la velocidad vertical, el coeficiente de sustentación y la configuración de empuje [1], este trabajo estará enfocado en el tratamiento matemático a partir del planteamiento original no en el modelo físico.

El concurso de Airbus estableció, además, una serie de "Key Performance Indicators" (KPI) para evaluar las soluciones propuestas. Estos KPI incluyen la provisión de un método cuántico o híbrido para el problema de optimización, una estimación de los recursos de computación cuántica requeridos para ejecutar el algoritmo, y una comparación del tiempo de cálculo entre el algoritmo cuántico y una implementación clásica del método de optimización propuesto [1].

El problema de investigación que se plantea en este trabajo es el desarrollo y la evaluación de un método eficiente para la optimización de la trayectoria de ascenso de aeronaves A320, basado en la reducción de dimensionalidad y la computación cuántica, que permita minimizar el consumo de combustible y el tiempo de vuelo, sujeto a las restricciones operativas y de seguridad relevantes. Se busca, en particular, explorar cómo la reducción de dimensionalidad del problema original de Airbus, a través del análisis del Jacobiano y la

identificación de variables independientes, puede simplificar el problema de optimización y hacerlo más tratable tanto para métodos de optimización clásicos como para algoritmos cuánticos.

La computación cuántica, un campo emergente con el potencial de revolucionar la capacidad de cómputo [12], ofrece nuevas posibilidades para resolver problemas de optimización complejos que son difíciles de abordar con métodos clásicos [13]. En particular, algoritmos como el "Quantum Approximate Optimization Algorithm"(QAOA) [14] y el "Variational Quantum Eigensolver"(VQE) [15] han mostrado resultados prometedores en la resolución de problemas de optimización combinatoria [16]. La tecnología de computación cuántica se halla en una etapa inicial de desarrollo limitado principalmente por el numero de cúbits y la coherencia cuántica [17]. En este contexto se aborda la preparación del problema de optimización de la trayectoria de ascenso para su eventual resolución mediante futuros computadores cuánticos.

El procedimiento metodológico adoptado en esta investigación comprendió secuencialmente: la simplificación del modelo matemático original de Airbus y la verificación del modelo resultante; la formulación simbólica de la función objetivo; la exploración de estrategias de muestreo para la generación de puntos en el espacio de soluciones; y la transformación de la función objetivo en una representación cuadrática binaria optimizable vía computación cuántica. De particular relevancia para este trabajo es la etapa de reducción de dimensionalidad. Al reducir el número de variables independientes de 364 a 104, se simplifica significativamente la complejidad del problema de optimización. Esto no solo facilita la aplicación de métodos de optimización clásicos, sino que también hace que el problema sea más adecuado para su eventual resolución en computadoras cuánticas, que actualmente tienen un número muy restringido de cúbits disponibles [18]. El análisis del Jacobiano [19], ha permitido identificar las dependencias lineales entre las ecuaciones de actualización de estado del modelo de Airbus y, en consecuencia, reducir el número de variables independientes. Este análisis, junto con la implementación de un método iterativo para calcular la masa y otras variables, ha sido clave para la simplificación del modelo. La búsqueda de puntos que cumplan con las restricciones del problema ha sido otro aspecto importante de esta investigación. Se han explorado diferentes métodos de muestreo, incluyendo la generación de puntos en una grilla hexaédrica, el muestreo aleatorio con perturbaciones gaussianas, la búsqueda de puntos adyacentes y finalmente, el

uso de secuencias de Sobol [20]. Cada uno de estos métodos tiene sus propias ventajas y desventajas en términos de eficiencia computacional, uniformidad del muestreo y capacidad para cubrir el espacio de soluciones [21]. La elección de un método u otro tiene implicaciones importantes para la calidad de la aproximación de la función objetivo y, en última instancia, para la eficiencia de la optimización.

La aproximación de la función objetivo mediante muestreo de la función original, específicamente una función cuadrática binaria, es determinante para la aplicación de la computación cuántica. Se intentó inicialmente utilizar polinomios de Chebyshev [22], que son conocidos por su precisión en la aproximación de funciones, sin embargo, la complejidad de la función objetivo y el gran número de variables involucradas (incluso después de la reducción de dimensionalidad) hicieron que este enfoque fuera computacionalmente inviable. Como alternativa, se optó por ajustar una función cuadrática binaria a un conjunto de puntos válidos obtenidos mediante la búsqueda con secuencias de Sobol. Esta función cuadrática, que puede ser expresada en la forma de un problema de optimización cuadrática sin restricciones binarias (QUBO) [23], es adecuada para su resolución en plataformas de computación cuántica como D-Wave [24].

Este trabajo contribuye al campo de la optimización de trayectorias de vuelo al proponer un método que combina la reducción de dimensionalidad, la búsqueda eficiente de puntos y la aproximación de la función objetivo para hacer que el problema sea tratable mediante computación cuántica. Si bien la optimización cuántica para el problema completo con N=53 está fuera del alcance de las capacidades de sistemas actuales, los resultados obtenidos para N=3 y N=6 o sea el trabajo en 10 dimensiones, proporcionan información sobre la estructura del problema y sugieren que la computación cuántica podría ofrecer ventajas significativas para mayores dimensiones.

Justificación de la investigación

La optimización de trayectorias de vuelo es un creciente campo de estudio en la industria aeronáutica, motivado por el aumento del transporte Aéreo y la necesidad de mitigar el impacto ambiental. En el presente se aborda la optimización de la trayectoria de ascenso de aeronaves, fase del vuelo con alto consumo de combustible y emisiones asociadas de gases de efecto invernadero. La investigación parte del problema propuesto en el .Airbus Quantum Computing Challenge"[1], que que busca explorar soluciones mediante compu-

tación cuántica. Aunque desarrollado en un entorno académico, el trabajo posee aplicabilidad en la industria aeronáutica por tratar un problema operacional con implicaciones prácticas. La justificación para esta investigación incluye: la búsqueda de eficiencia en la aviación, el potencial de la computación cuántica para la optimización y la aportación al conocimiento en la convergencia de ambas disciplinas.

Necesidad de la eficiencia en la aviación:

El consumo de combustible representa un componente principal de los costos operativos para las aerolíneas. Adicionalmente al factor económico, la combustión de dicho combustible genera emisiones de gases de efecto invernadero que contribuyen al cambio climático. La optimización de la trayectoria de ascenso permite minimizar el consumo de combustible, lo cual impacta la rentabilidad de las aerolíneas y la sostenibilidad medioambiental del sector aeronáutico. Una reducción en el consumo durante la fase de ascenso, incluso si es marginal por operación individual, al multiplicarse por el volumen global de vuelos se traduce en ahorros económicos agregados y en una disminución de la huella de carbono de la industria. Por consiguiente, la determinación de trayectorias de ascenso óptimas, sujeta a restricciones operativas y de seguridad, constituye una necesidad relevante para la aviación. Estudios como el de Alligier et al. [2] ilustran la aplicación de programación lineal entera mixta para la planificación robusta de vuelos ante la incertidumbre de las condiciones operacionales, lo que resalta la pertinencia de la optimización en contextos de aviación aplicada.

Potencial de la computación cuántica para la optimización:

La optimización de trayectorias de vuelo constituye un problema de mucha complejidad, debido a la no linealidad de las ecuaciones de movimiento, la alta dimensionalidad del espacio de variables y la existencia de múltiples restricciones. Dichas características hacen que la optimización usando métodos clásicos sea poco viable y en ese punto la opción del uso de computación cuántica espera ser una alternativa, la utilidad de los algoritmos cuánticos radica en su potencial exploración de resultados en el espacio de soluciones de forma más optima en comparación con los métodos clásicos, con el potencial de culminar el mismo trabajo de búsqueda de solución en tiempos humanamente razonables. Si bien la tecnología de computación cuántica se encuentra en etapas iniciales de desarrollo, existe un potencial reconocido para la resolución de problemas de optimización complejos.

Algoritmos como QAOA [14] y VQE [15] han mostrado capacidad en la resolución de problemas de optimización combinatoria, lo que sugiere vías para la optimización de trayectorias de vuelo. Se aborda la preparación del problema de optimización para su futura ejecución en computadoras cuánticas, en previsión del desarrollo de hardware con mayores capacidades.

Contribución al conocimiento científico:

Este trabajo contribuye al conocimiento científico al unir la optimización y la computación cuántica aplicada al sector aeronáutico. La reducción de complejidad del modelo matemático base de Airbus usando teoría matemática y la manipulación algebraica para el reconocimiento de variables libres, es un aporte de significativo que minimiza la dificultad del problema y lo torna más manejable para algoritmos de optimización tanto tradicionales como cuánticos. El decremento de la dimensionalidad de 364 a 104 variables no únicamente simplifica el problema, sino que también va en línea con su adecuación a las actuales restricciones de los computadores cuánticos. El sondeo de distintos algoritmos de muestreo, añadiendo la exploración de puntos cercanos y la utilización de secuencias de Sobol [20], facilita una apreciable perspectiva acerca de la efectividad y el alcance de distintas técnicas de exploración en dominios multidimensionales. El cálculo aproximado de la función objetivo a través de una función cuadrática binaria propicia para la optimización en D-Wave [24], significa un resultado metodológico que hace posible usar algoritmos cuánticos en un problema específico de la industria aeronáutica. La evaluación contrastada entre los algoritmos de optimización clásica y cuántica para distintas dimensiones del problema brinda conocimiento apreciable sobre la capacidad y la extensión posible de cada aproximación. Este trabajo establece un fundamento para indagaciones posteriores en el uso de la computación cuántica para la tarea de optimización de rutas de vuelo, conforme la tecnología cuántica evoluciona y se hace de mayor disponibilidad.

Capítulo 2

Mecánica de vuelo y aerodinámica

La mecánica de vuelo y la aerodinámica son conocimientos fundamentales en la comprensión del desplazamiento de las aeronaves en la atmósfera. Estas áreas son el soporte teórico para analizar, modelar y optimizar la trayectoria en las rutas de vuelo en cuanto a su uso de combustible, duración de vuelos y, en síntesis, todo elemento que afecte las capacidades operativas del avión. La comprensión específica de dichos temas resulta fundamental para ingenieros aeronáuticos, aviadores, supervisores de tráfico aéreo, administradores de operaciones y diversas especialidades en la industria de la aviación [25, 26].

La *aerodinámica* estudia el desplazamiento del aire y el efecto de las fuerzas sobre objetos que se mueven en él, en especial las aeronaves. Por su parte, la *mecánica de vuelo* se orienta hacia las ecuaciones que describen el movimiento de las aeronaves, integrando las fuerzas generadas por la aerodinámica, el efecto de la gravedad, el empuje de los motores y las limitaciones propias de la operación [25, 26]. En el presente apartado se expone una perspectiva completa sobre las variables y ecuaciones de mayor relevancia, detallando cada una y su conexión con las prestaciones generales de la aeronave, así como las restricciones habituales que se manifiestan durante una misión típica de vuelo.

2.1. Conceptos fundamentales de aerodinámica

Flujo de aire, densidad y presión

El vuelo de las aeronaves es su movimiento a través del aire y por ende las principales

características del mismo como lo son densidad del aire ($\rho(Zp_i)$), la presión (p) y la temperatura (T) afectan su desplazamiento. Estas varían con la altitud y son parámetros indispensables en los modelos teóricos atmosféricos estándar, como la Atmósfera Estándar Internacional (ISA, por sus siglas en inglés).

La densidad del aire $\rho(Zp_i)$ es especialmente relevante, ya que las fuerzas aerodinámicas, como la sustentación (F_L) y la resistencia (D), dependen de ella. La relación típica para la densidad atmosférica a diferentes altitudes, asumiendo un gradiente térmico lineal, es:

$$\rho(Zp_i) = \rho_0 \left(\frac{T}{T_0} \right)^{\alpha_0 - 1}, \tag{2.1}$$

donde ρ_0 es la densidad del aire al nivel del mar, T_0 la temperatura a nivel del mar, y α_0 es un exponente derivado de la relación entre la gravedad, el gas constante del aire R y el gradiente térmico L_Z. Este tipo de relación se deriva en condiciones atmosféricas estándar [27].

Velocidades características: TAS, CAS y Mach

En el estudio de aeronaves existen múltiples definiciones de velocidad, cada una diseñada para captar diferentes aspectos aerodinámicos y de medición:

- **TAS (True Airspeed):** Es la velocidad real del avión con respecto al aire que lo rodea. Afecta directamente las fuerzas aerodinámicas.
- **CAS (Calibrated Airspeed):** Es la velocidad indicada corregida por errores instrumentales y de posición. Se aproxima a la IAS (Indicated Airspeed) pero con correcciones. La CAS es útil porque al combinarse con la densidad local permite estimar las fuerzas aerodinámicas de manera consistente.
- **Número de Mach (M):** Es la razón entre la velocidad del flujo (o la velocidad del avión) y la velocidad local del sonido a:

  $$M = \frac{v_i}{a} = \frac{v_i}{\sqrt{\gamma R T}}, \tag{2.2}$$

  donde γ es la razón de calores específicos del aire ($\approx 1{,}4$), R es la constante del gas para el aire, y T la temperatura local.

El número de Mach es crucial para determinar el régimen de vuelo: subsónico, transónico o supersónico, ya que las características aerodinámicas cambian drásticamente según este parámetro [28].

Fuerzas aerodinámicas: sustentación y resistencia

La fuerza aerodinámica total sobre un ala o aeronave se puede descomponer en sustentación (F_L), perpendicular a la dirección del flujo, y resistencia (D), paralela a ella.

$$F_L = \frac{1}{2}\rho(Zp_i)v_i^2 S_{REF} Cz, \quad D = \frac{1}{2}\rho(Zp_i)v_i^2 S_{REF} C_D, \tag{2.3}$$

donde v_i puede ser la TAS, S_{REF} es el área de referencia aerodinámica, Cz es el coeficiente de sustentación y C_D el coeficiente de resistencia. Estos coeficientes dependen del ángulo de ataque, la forma del ala, la compresibilidad, la viscosidad del aire, entre otros factores. El coeficiente de sustentación Cz generalmente se incrementa con el ángulo de ataque hasta un valor máximo ($C_{L_{max}}$), mientras que el coeficiente de resistencia C_D aumenta con el ángulo de ataque y con la formación de estelas turbulentas. A menudo se representa la resistencia en función de Cz a través de la expresión aerodinámica:

$$C_D = Cx_0 + kCz^2, \tag{2.4}$$

donde Cx_0 es la resistencia parasitaria mínima y k un factor que describe el crecimiento cuadrático de la resistencia inducida por sustentación [25].

2.2. Mecánica de vuelo: ecuaciones del movimiento

Ecuación básica de la trayectoria

El movimiento de la aeronave en un plano vertical puede analizarse considerando un ángulo de ascenso γ_i, definido como el ángulo entre la trayectoria de vuelo y el eje horizontal. Si se asume simetría y vuelo coordinado, las fuerzas principales sobre el avión son: tracción o empuje ($F_{N_{MCL_i}}$), peso ($m_i g_0$), sustentación (F_L) y resistencia (D).

Para la componente de fuerzas a lo largo de la trayectoria, tenemos:

$$F_{N_{MCL_i}} \cos\alpha - D - m_i g_0 \sin\gamma_i = m_i \frac{dv_i}{dt}, \tag{2.5}$$

donde α es el ángulo entre el vector empuje y la trayectoria. Para la dirección perpendicular a la trayectoria:

$$F_L + F_{N_{MCL_i}} \sin\alpha - m_i g_0 \cos\gamma_i = m_i v_i \frac{d\gamma_i}{dt}. \tag{2.6}$$

Cuando el empuje se alinea aproximadamente con la dirección de vuelo, $\alpha \approx 0$, y simplificando las condiciones de ascenso, por ejemplo en régimen estable, las ecuaciones se simplifican [26, 29].

Masa y consumo de combustible

La masa de la aeronave (m_i) cambia durante el vuelo debido al consumo de combustible. A una razón de consumo $\dot{m}_f$, la masa total del avión disminuye con el tiempo:

$$\frac{dm_i}{dt} = -\eta \lambda_i F_{N_{MCL_i}}, \tag{2.7}$$

La ecuación considera η como el consumo específico de combustible y λ_i como el factor de empuje, que varía entre 0 y 1. Esta relación lineal constituye una aproximación preliminar. La masa desempeña un papel esencial en la dinámica del sistema, ya que incide directamente en la sustentación requerida y en la capacidad de aceleración disponible.

Altitud, velocidad vertical y alcance

A medida que el avión asciende a mayores altitudes (Zp_i), disminuye la densidad del aire $\rho(Zp_i)$. Esto afecta la sustentación y la resistencia, así como también el empuje disponible en motores turbofan. Una ecuación típica para el incremento de altitud en función de la velocidad vertical $Vz_i = v_i \sin\gamma_i$ es:

$$\frac{dZp_i}{dt} = v_i \sin\gamma_i. \tag{2.8}$$

El alcance horizontal durante un segmento de vuelo se obtiene de:

$$\frac{ds_i}{dt} = v_i \cos \gamma_i. \tag{2.9}$$

2.3. Variables de estado y control en la mecánica de vuelo

En mecánica de vuelo, las variables se dividen a menudo en:

- **Variables de Estado:** Aquellas que describen completamente el estado del sistema en un momento dado. Por ejemplo, la altitud (Zp_i), la velocidad (v_i), la masa (m_i), la distancia recorrida (s_i), el tiempo (t_i) y el ángulo de ascenso (γ_i).
- **Variables de Control:** Estas son las variables que el piloto o el sistema de control automático puede modificar. Por ejemplo, el empuje ($F_{N_{MCL_i}}$) o el factor de empuje (λ_i) que varía entre 0 y 1.

2.4. Limitaciones y restricciones operacionales

Límites de velocidad

Las aeronaves tienen restricciones de velocidad para garantizar la seguridad estructural y el confort de los pasajeros. Dos restricciones comunes son:

- **VMO (Maximum Operating Velocity):** Límite máximo de velocidad calibrada. Por ejemplo:

$$CAS(v_i, Zp_i) \leq VMO. \tag{2.10}$$

- **MMO (Maximum Operating Mach):** Límite máximo de número de Mach:

$$M(v_i, Zp_i) \leq MMO. \tag{2.11}$$

Estos límites garantizan que el avión no sufra problemas estructurales o aerodinámicos como la aparición de ondas de choque intensas al superar el régimen de diseño.

Tasa mínima de ascenso

La aviación comercial exige tasas mínimas de ascenso, ya sea por estándares de seguridad o por requisitos operacionales. Por ejemplo:

$$v_i \sin \gamma_i \geq Vz_{\min}, \tag{2.12}$$

donde $Vz_{\min}$ es la tasa mínima de ascenso requerida.

Límite en coeficiente de sustentación

El coeficiente de sustentación no puede superar un valor máximo antes de que el ala entre en pérdida:

$$Cz_i \leq Cz_{\max}. \tag{2.13}$$

Este límite impone restricciones en el ángulo de ataque y, en consecuencia, en la maniobrabilidad y capacidad de ascenso de la aeronave.

2.5. Estructura de problemas de optimización en mecánica de vuelo

Cuando se aborda la planificación del perfil de ascenso, se plantea un problema de optimización multivariable con restricciones. Los objetivos pueden incluir la minimización del consumo de combustible, la minimización del tiempo o una combinación de ambos mediante un índice de costo (CI, *Cost Index*), que es la relación entre el coste en combustible y el coste en tiempo.
El problema aquí tratado tiene la forma:

$$\min_{\text{control}} \phi = \text{consumo} + CI \times \text{tiempo}, \tag{2.14}$$

sujeto a las ecuaciones de movimiento, las limitaciones aerodinámicas y las restricciones operativas. Este tipo de problema es altamente no lineal.

2.6. Cálculo del consumo y tiempo

El cálculo del consumo total y del tiempo total de vuelo se realiza integrando las tasas correspondientes a lo largo de la trayectoria de ascenso. Utilizando las variables de estado y control definidas, se pueden establecer las siguientes relaciones:

Consumo de combustible

El consumo de combustible se modela a través de la disminución de la masa de la aeronave (m_i) debido al consumo específico de combustible (η) y al empuje aplicado ($\lambda_i F_{N_{MCL_i}}$). La relación diferencial es:

$$\frac{dm_i}{dt} = -\eta \lambda_i F_{N_{MCL_i}}. \tag{2.15}$$

Integrando a lo largo de la trayectoria, se obtiene el consumo total de combustible:

$$\Delta m = \int_{t_0}^{t_F} \eta \lambda_i F_{N_{MCL_i}} \, dt. \tag{2.16}$$

Cálculo del tiempo de vuelo

El tiempo total de vuelo (t_F) se obtiene integrando la tasa de cambio del tiempo a lo largo de la trayectoria de ascenso. Dado que t_i representa el tiempo acumulado hasta el punto i, la relación diferencial es:

$$\frac{dt_i}{dt} = 1, \tag{2.17}$$

lo que implica:

$$t_F = t_0 + \int_0^{N-1} \frac{dt_i}{ds_i} \, ds_i. \tag{2.18}$$

Sin embargo, debido a la discretización del problema, el tiempo se puede aproximar mediante:

$$t_F = \sum_{i=0}^{N-2} \frac{t_{i+1} - t_i}{Zp_{i+1} - Zp_i} \Delta Zp_i, \tag{2.19}$$

donde $\Delta Zp_i = Zp_{i+1} - Zp_i$.

Integración en la trayectoria

Dado que el problema está discretizado en N puntos, los consumos y tiempos se suman a lo largo de cada segmento de ascenso:

$$\text{Consumo Total} = \sum_{i=0}^{N-2} \eta \lambda_i F_{N_{MCL_i}} \Delta t_i, \tag{2.20}$$

$$\text{Tiempo Total} = \sum_{i=0}^{N-2} \Delta t_i. \tag{2.21}$$

Donde Δt_i es el tiempo transcurrido en el segmento i-ésimo, calculado a partir de la velocidad y el cambio de altitud.

2.7. Consideraciones aerodinámicas avanzadas

Compresibilidad y número de Mach

Con el incremento de la altitud del avión (Zp_i), la temperatura disminuye y la velocidad del sonido se reduce. Si la aeronave vuela a velocidad constante en TAS (v_i), el número de Mach ($M(v_i, Zp_i)$) puede aumentar, llegando a regímenes transónicos. Esto induce cambios significativos en la distribución de presiones alrededor del ala, incrementando la resistencia de onda. Por ello, en vuelos a gran altitud, el límite de Mach suele ser más restrictivo que el límite de CAS.

Polar aerodinámica y eficiencia

La diferencia entre el coeficiente de resistencia y el coeficiente de sustentación determina la eficiencia aerodinámica. La relación F_L/D indica cuán eficaz es el ala en generar sustentación con respecto a la resistencia. Una alta eficiencia (L/D elevado) es deseable ya que reduce el consumo de combustible para una sustentación dada. Múltiples diseños de alas, perfiles aerodinámicos y dispositivos hipersustentadores buscan optimizar esta relación.

Empuje del motor y su variación con la altitud

El empuje máximo disponible ($F_{N_{MCL_i}}(Zp_i)$), especialmente en motores turbofan, disminuye con la altitud debido a la menor densidad del aire y la menor masa de aire que

ingresa en el motor. A su vez, el régimen de revoluciones y ajustes del motor permiten variar el empuje mediante el factor de empuje (λ_i) entre un valor mínimo y el máximo disponible. Conociendo la curva de empuje máximo ($F_{N_{MCL_i}}(Zp_i)$), se puede modelar la reducción de empuje disponible al ascender.

2.8. Unificación de conceptos

La mecánica de vuelo y la aerodinámica se articulan en un sistema de ecuaciones y restricciones que rigen el comportamiento de una aeronave. El perfil de ascenso se define mediante los siguientes elementos:

1. Partida desde una altitud y velocidad iniciales determinadas (Zp_I, $v_0 = TAS(CAS_I, Zp_I)$).
2. Incremento de altitud mediante ajustes en el empuje y el ángulo de ascenso (λ_i, γ_i).
3. Mantenerse dentro de los límites de velocidad (CAS y Mach).
4. Asegurar una tasa mínima de ascenso ($Vz_{\min}$).
5. Controlar el coeficiente de sustentación para mantenerse dentro de $Cz_{\max}$.
6. Minimizar una función de coste que combina tiempo y consumo de combustible mediante el Cost Index (CI).

Este es el problema que se abordará en capítulos posteriores.

2.9. Ejemplo de ecuaciones y cálculos de rendimiento

Consideremos un ejemplo simplificado: un avión a una altitud Zp_i con velocidad TAS v_i. El número de Mach vendrá dado por:

$$M(v_i, Zp_i) = \frac{v_i}{\sqrt{\gamma R(Ts_0 + L_Z Zp_i)}}, \tag{2.22}$$

donde Ts_0 es la temperatura al nivel del mar y L_Z el gradiente térmico. La densidad a esa altitud:

$$\rho(Zp_i) = \rho_0 \left(\frac{Ts_0 + L_Z Zp_i}{Ts_0}\right)^{\alpha_0 - 1}. \tag{2.23}$$

La sustentación requerida para equilibrar el peso en vuelo nivelado (ignorando empuje vertical):

$$F_L = m_i g_0 \approx \frac{1}{2}\rho(Zp_i)v_i^2 S_{REF} Cz_i. \tag{2.24}$$

Se puede resolver entonces para el coeficiente de sustentación como sigue:

$$Cz_i = \frac{2m_i g_0}{\rho(Zp_i)v_i^2 S_{REF}}. \tag{2.25}$$

Si se conoce Cz_i, el C_D se deduce a partir de la polar aerodinámica:

$$C_D = Cx_0 + kCz_i^2. \tag{2.26}$$

La resistencia resultante es:

$$D = \frac{1}{2}\rho(Zp_i)v_i^2 S_{REF} C_D. \tag{2.27}$$

Finalmente, el empuje requerido para vuelo recto y nivelado es igual a la resistencia:

$$F_{N_{MCL_i}} = D. \tag{2.28}$$

Para lograr el ascenso, el empuje debe superar la suma de la resistencia y la componente del peso paralela a la trayectoria.

$$F_{N_{MCL_i}} - D = m_i g_0 \sin\gamma_i. \tag{2.29}$$

Al incorporar el consumo de combustible:

$$\frac{dm_i}{dt} = -\eta\lambda_i F_{N_{MCL_i}}. \tag{2.30}$$

Con estas ecuaciones, se forman sistemas diferenciales que describen la evolución de v_i, m_i, Zp_i, s_i y γ_i con el tiempo, sujetos a las restricciones de velocidad, coeficientes y empuje máximo.

2.10. Estrategias de optimización y herramientas matemáticas

El proceso de encontrar una trayectoria óptima que minimice el consumo más el coste en tiempo es complejo.
Entre las estrategias consideradas figuran:

- Procedimientos de control para optimización, como el principio del máximo de Pontryagin [30], que utilizan funciones de costo y multiplicadores de Lagrange.
- Métodos numéricos numéricos de programación no lineal (NLP), destinadas a la solución de sistemas de ecuaciones con restricciones.
- Aproximaciones metaheurísticas (por ejemplo, algoritmos genéticos u optimización por enjambre de partículas entre otros) para problemas cuya estructura dificulta la aplicación de métodos de solución directa.
- Métodos híbridos (potencialmente completamente cuánticos a futuro), aprovechando la capacidad de la computación cuántica para acelerar partes de la optimización.

2.11. Importancia práctica

La comprensión de estas variables y ecuaciones posee una relevancia que excede el dominio puramente conceptual. En la operación aeronáutica, la determinación de perfiles de ascenso por parte de pilotos y sistemas de gestión de vuelo (FMS) es necesaria para la minimización del consumo de combustible y la contención de costos. Esta determinación debe realizarse además en conformidad con las directrices del control de tráfico aéreo (ATC) y los protocolos ambientales vigentes. El manejo de la trayectoria durante el ascenso presenta una importancia acentuada en operaciones de corto radio. En estas, la proporción del tiempo total de vuelo correspondiente a las fases de ascenso y descenso supera a la de los vuelos de largo trayecto. En consecuencia, la reducción del consumo de combustible en dichas fases ejerce una influencia sobre la economía del operador aéreo.

Capítulo 3

Teorema de la función implícita

En el análisis matemático, a menudo encontramos ecuaciones que relacionan dos o más variables. Algunas veces, estas ecuaciones pueden ser manipuladas algebraicamente para expresar una variable explícitamente como función de las otras. Sin embargo, hay situaciones donde tal manipulación resulta impráctica, o incluso imposible. En estos casos, surge la noción de función implícita, y con ella, la necesidad de herramientas que permitan comprender y trabajar con estas relaciones indirectas. Entre estas herramientas, el Teorema de la Función Implícita ocupa un lugar central [31].

El Teorema proporciona un conjunto de condiciones bajo las cuales una ecuación que involucra múltiples variables define, al menos localmente, una o más de esas variables como función implícita de las restantes. Su importancia radica no solo en su capacidad para establecer la existencia de estas funciones implícitas, sino también en la información que proporciona sobre sus propiedades, como la diferenciabilidad [32].

3.1. Funciones implícitas

Conceptos:

Previo al análisis del Teorema, se define el concepto de función implícita como sigue:

Se considera una ecuación que vincula las variables x e y mediante la relación $F(x, y) = 0$, donde F denota una función de dos variables. En algunas situaciones, es factible despejar y como función de x, obteniendo una expresión $y = g(x)$. Esta expresión determina y en términos de x [33].

Sin embargo, existen ecuaciones donde tal despeje de y no se logra de manera directa o

global. Por ejemplo la ecuación $x^2 + y^2 - 1 = 0$, si bien se derivan dos soluciones para y, a saber, $y = \sqrt{1 - x^2}$ e $y = -\sqrt{1 - x^2}$, cada una de ellas corresponde únicamente a un segmento del círculo unitario representado por la ecuación $F(x, y) = 0$. Este escenario introduce la necesidad del concepto de función implícita. Aunque no es posible expresar y como una única función explícita de x para todos los puntos que satisfacen la ecuación, es posible considerar que la ecuación $F(x, y) = 0$ define a y implícitamente como función de x, al menos en un entorno local alrededor de un punto dado (x_0, y_0) que satisface la ecuación.

Intuitivamente es posible pensar que cerca del punto (x_0, y_0), existe una función $y = g(x)$ cuya gráfica çoincideçon la curva definida por la ecuación $F(x, y) = 0$. Esta función $g(x)$ es la función implícita que la ecuación define localmente [34].

3.2. El teorema de la función implícita

El Teorema proporciona las condiciones precisas bajo las cuales es posible afirmar la existencia de esta función implícita $g(x)$. Formalmente, el teorema puede presentarse así:

Teorema de la Función Implícita (caso de dos variables):

Sea $F(x, y)$ una función con dos variables, y sea (x_0, y_0) un punto tal que $F(x_0, y_0) = 0$. Supongamos que:

1. F es continuamente diferenciable en una vecindad del punto (x_0, y_0). Esto significa que las derivadas parciales $\frac{\partial F}{\partial x}$ y $\frac{\partial F}{\partial y}$ existen y son continuas en un entorno del punto [35].

2. La derivada parcial de F con respecto a y evaluada en (x_0, y_0) es no nula, es decir, $\frac{\partial F}{\partial y}(x_0, y_0) \neq 0$.

Entonces, existen intervalos abiertos I y J que contienen a x_0 e y_0, respectivamente, y una única función $g : I \to J$ tal que:

- $g(x_0) = y_0$
- $F(x, g(x)) = 0$ para todo x en el intervalo I.

Además, la función g es diferenciable en I, y su derivada puede calcularse como:

$$g'(x) = -\frac{\frac{\partial F}{\partial x}(x, g(x))}{\frac{\partial F}{\partial y}(x, g(x))}$$

según se demuestra en [31].

3.3. Análisis de las condiciones

Las condiciones del Teorema no son arbitrarias; cada una juega un papel crucial para garantizar la existencia y unicidad de la función implícita $g(x)$. La importancia de su significado:

1. Continuidad de las Derivadas Parciales:

La condición de que F sea continuamente diferenciable en una vecindad de (x_0, y_0) asegura que la superficie definida por $F(x, y)$ sea suave cerca de ese punto. Esta suavidad es esencial para poder aproximar localmente la superficie mediante un plano tangente, concepto fundamental para la diferenciación implícita [32].

2. Derivada Parcial con respecto a y no nula:

Esta condición es el corazón del Teorema. Intuitivamente, $\frac{\partial F}{\partial y}(x_0, y_0) \neq 0$ implica que la superficie definida por $F(x, y)$ no es "vertical.en el punto (x_0, y_0) con respecto al eje y. Si fuera vertical, un pequeño cambio en x podría corresponder a múltiples valores de y, violando la unicidad de la función implícita.

Geométricamente, esta condición garantiza que el plano tangente a la superficie en (x_0, y_0) no es paralelo al eje y. Esto permite "proyectar"la curva definida por $F(x, y) = 0$ sobre el eje x en una vecindad de (x_0, y_0), obteniendo así la gráfica de una función $y = g(x)$ [33].

3.4. El rol del jacobiano

Aunque en el caso de dos variables el jacobiano se reduce a la derivada parcial $\frac{\partial F}{\partial y}$, su importancia se manifiesta plenamente cuando extendemos el Teorema a funciones con más variables.

El jacobiano, en general, es una matriz formada por las derivadas parciales de una función vectorial. En el presente contexto, se considera una función $F(x_1, x_2, \ldots, x_n, y) = 0$, el jacobiano de interés es el determinante de la matriz formada por las derivadas parciales de F con respecto a las variables que se quiere considerar como implícitas (en este caso, solo y).

El requisito de que este jacobiano sea no nulo en el punto en cuestión generaliza la condición $\frac{\partial F}{\partial y}(x_0, y_0) \neq 0$ al caso de múltiples variables. Asegura que, localmente, la ecuación $F(x_1, x_2, \ldots, x_n, y) = 0$ define a y como una función implícita de las variables $x_1, x_2, \ldots, x_n$ [34].

3.5. Generalización a múltiples variables y ecuaciones

El Teorema puede generalizarse a sistemas de ecuaciones y funciones con múltiples variables. Supongamos que tenemos un sistema de m ecuaciones con $n + m$ variables:

$$\begin{aligned} F_1(x_1, \ldots, x_n, y_1, \ldots, y_m) &= 0 \\ F_2(x_1, \ldots, x_n, y_1, \ldots, y_m) &= 0 \\ &\vdots \\ F_m(x_1, \ldots, x_n, y_1, \ldots, y_m) &= 0 \end{aligned}$$

Queremos saber si este sistema define implícitamente a las variables $y_1, \ldots, y_m$ como funciones de las variables $x_1, \ldots, x_n$ en una vecindad de un punto $(x_1^0, \ldots, x_n^0, y_1^0, \ldots, y_m^0)$ que satisface el sistema.

En este caso, el jacobiano relevante es el determinante de la matriz $m \times m$ formada por las derivadas parciales de las funciones $F_1, \ldots, F_m$ con respecto a las variables $y_1, \ldots, y_m$:

$$J = \begin{vmatrix} \frac{\partial F_1}{\partial y_1} & \frac{\partial F_1}{\partial y_2} & \cdots & \frac{\partial F_1}{\partial y_m} \\ \frac{\partial F_2}{\partial y_1} & \frac{\partial F_2}{\partial y_2} & \cdots & \frac{\partial F_2}{\partial y_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial y_1} & \frac{\partial F_m}{\partial y_2} & \cdots & \frac{\partial F_m}{\partial y_m} \end{vmatrix}$$

El Teorema de la Función Implícita para este caso generalizado establece que si todas las funciones F_i son continuamente diferenciables en una vecindad del punto, y el jacobiano J evaluado en el punto es no nulo, entonces el sistema define implícitamente a $y_1, \ldots, y_m$ como funciones de $x_1, \ldots, x_n$ en una vecindad del punto. Además, estas funciones implícitas son diferenciables, y sus derivadas pueden calcularse usando la regla de la cadena y la inversa del jacobiano [31].

Ejemplo ilustrativo

Para reforzar la idea se plantea nuevamente la ecuación que define un círculo unitario:

$$F(x, y) = x^2 + y^2 - 1 = 0$$

Tomemos el punto $\left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$ que satisface la ecuación.
Las derivadas parciales son:

$$\frac{\partial F}{\partial x} = 2x$$

$$\frac{\partial F}{\partial y} = 2y$$

Evaluando en el punto, se tiene:

$$\frac{\partial F}{\partial x}\left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right) = \sqrt{2}$$

$$\frac{\partial F}{\partial y}\left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right) = \sqrt{2} \neq 0$$

Como la derivada parcial con respecto a y es no nula, el Teorema garantiza existencia de una función $y = g(x)$ definida implícitamente por la ecuación en una vecindad de $\left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$. Además, es posible calcular la derivada de $g(x)$:

$$g'(x) = -\frac{2x}{2y} = -\frac{x}{y}$$

Evaluando en el punto $x_0 = \frac{\sqrt{2}}{2}$, y considerando que $y_0 = g(x_0) = \frac{\sqrt{2}}{2}$, obtenemos:

$$g'\left(\frac{\sqrt{2}}{2}\right) = -\frac{\frac{\sqrt{2}}{2}}{\frac{\sqrt{2}}{2}} = -1$$

3.6. Aplicaciones e importancia

Entre las aplicaciones del Teorema de la Función Implícita se encuentran:

- **Cálculo de Derivadas:** Obtención de derivadas para funciones definidas implícitamente, sin requerir la explicitación funcional [35].
- **Análisis Geométrico Diferencial:** Estudio local de curvas y superficies (determinación de espacios tangentes, vectores normales, curvatura) especificadas mediante ecuaciones que relacionan sus coordenadas [36].
- **Optimización con Restricciones:** Análisis de problemas de optimización donde las condiciones de restricción se formulan como ecuaciones implícitas [32].

- **Modelización Económica:** Descripción y análisis de relaciones funcionales entre variables definidas a través de sistemas de ecuaciones, como en la teoría del equilibrio general[37]..
- **Ingeniería y Ciencias Físicas:** Formulación y estudio de sistemas (e.g., mecánicos, eléctricos, de control) en los que las variables de estado o parámetros se vinculan a través de ecuaciones implícitas [38]..

Capítulo 4

Optimización y métodos de muestreo

En este capítulo, se describen los métodos de optimización y muestreo empleados en este trabajo para abordar el problema de la optimización de la trayectoria de ascenso de aeronaves. La optimización es una rama fundamental de las matemáticas y la informática [39, 40] que busca encontrar la mejor solución a un problema, generalmente minimizando o maximizando una función objetivo sujeta a ciertas restricciones [41]. El muestreo, por otro lado, se refiere a las técnicas utilizadas para seleccionar un subconjunto de puntos de un espacio de búsqueda, con el objetivo de obtener información representativa sobre la función objetivo en ese espacio [**?**, 42].

El problema de optimización de la trayectoria de ascenso, tal como se plantea en este estudio, es complejo debido a la no linealidad de las ecuaciones de movimiento, la alta dimensionalidad del espacio de búsqueda y la presencia de múltiples restricciones [43]. Para enfrentar estos desafíos, se han explorado varios métodos de optimización y muestreo.

En las siguientes secciones, se describen con mas detalle tres métodos de optimización: Nelder-Mead, Evolución Diferencial y gp_minimize de la biblioteca skopt. Además, se exponen los métodos de muestreo utilizados en este trabajo: muestreo basado en hexágonos, muestreo aleatorio con perturbaciones gaussianas, búsqueda de puntos adyacentes y muestreo basado en secuencias de Sobol. Para cada método, se describen sus fundamentos, ventajas y desventajas, y su aplicabilidad al problema de optimización de la trayectoria de ascenso.

4.1. Métodos de optimización

Nelder-Mead

El método Nelder-Mead, también conocido como método simplex, es un algoritmo de optimización numérica que no requiere el cálculo de gradientes. Fue propuesto por John Nelder y Roger Mead en 1965 [41]. Este método es particularmente útil para problemas donde la función objetivo no es diferenciable o su gradiente es difícil de calcular.

Fundamento Teórico

Nelder-Mead opera en un espacio de búsqueda n-dimensional utilizando un simplex, que es un politopo con n+1 vértices. Cada iteración comprende la evaluación de la función objetivo en los puntos que constituyen los vértices del símplex. A partir de esta evaluación se determina el vértice a ser sustituido, lo cual se efectúa a través de transformaciones geométricas. Estas transformaciones incluyen:

- **Reflexión**: Se refleja el peor vértice a través del centroide de los vértices restantes. Matemáticamente, si x_{worst} es el peor vértice y x_{centroid} es el centroide, la reflexión se define como:

  $$x_{\text{reflected}} = x_{\text{centroid}} + \alpha(x_{\text{centroid}} - x_{\text{worst}})$$

  donde α es el factor de reflexión, típicamente $\alpha = 1$.

- **Expansión**: Si la reflexión produce un punto mejor que el mejor vértice actual, se expande el simplex en esa dirección. La expansión se calcula como:

  $$x_{\text{expanded}} = x_{\text{centroid}} + \gamma(x_{\text{reflected}} - x_{\text{centroid}})$$

  donde $\gamma > 1$ es el factor de expansión.

- **Contracción**: Si la reflexión produce un punto peor que el segundo peor vértice, se contrae el simplex en esa dirección. La contracción se define como:

  $$x_{\text{contracted}} = x_{\text{centroid}} + \beta(x_{\text{worst}} - x_{\text{centroid}})$$

donde $0 < \beta < 1$ es el factor de contracción.

- **Reducción**: Si ninguno de los pasos anteriores mejora la solución, se reduce el simplex hacia el mejor vértice. La reducción se realiza mediante:

$$x_i' = x_{\text{best}} + \delta(x_i - x_{\text{best}}) \quad \forall i \neq \text{best}$$

donde $0 < \delta < 1$ es el factor de reducción.

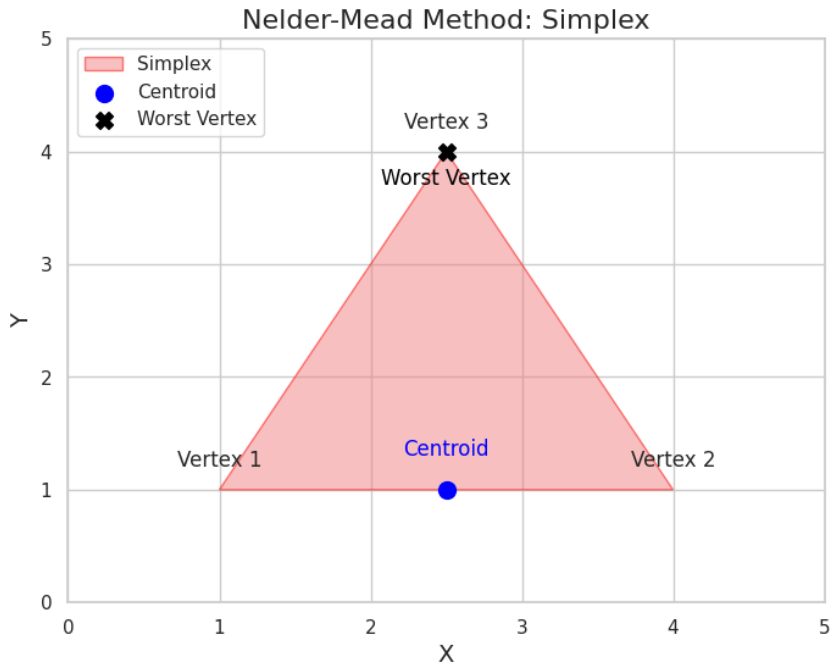

Figura 4.1: Representación gráfica del método Nelder-Mead utilizando un simplex en un espacio bidimensional.

El proceso iterativo concluye al satisfacerse un criterio de convergencia, como puede ser que la magnitud del simplex descienda por debajo de un umbral o que la diferencia entre los valores de la función objetivo en los vértices no exceda una tolerancia. [44].

Ventajas y desventajas

Las principales ventajas de Nelder-Mead son:

- **Simplicidad**: Es fácil de implementar y entender.
- **No requiere gradientes**: Es aplicable a funciones no diferenciables o con gradientes costosos de calcular.

- **Robustez**: Es relativamente robusto a la presencia de ruido en la función objetivo.

El método Nelder-Mead exhibe ciertas características que pueden constituir limitaciones en su aplicación:

- **Velocidad de Convergencia**: Manifestación de una tasa de convergencia reducida, fenómeno acentuado en problemas con alta dimensionalidad.
- **Sensibilidad a la configuración inicial**: Depende del desempeño del algoritmo respecto a la geometría (forma y tamaño) del símplex inicial seleccionado.
- **Estancamientos Locales**: Posibilidad de estancamiento de la búsqueda en puntos correspondientes a mínimos locales del espacio de soluciones.

Evolución diferencial

La Evolución Diferencial (DE) es un algoritmo de optimización global basado en poblaciones, propuesto por Rainer Storn y Kenneth Price en 1997 [43]. Pertenece a la familia de los algoritmos evolutivos y es conocido por su eficiencia y robustez en la resolución de problemas de optimización no lineales y no convexos.

Fundamento Teórico

DE opera sobre una población de individuos, donde cada individuo representa una solución candidata. En cada generación, el algoritmo realiza los siguientes pasos:

1. **Mutación**: Para cada individuo (llamado ïndividuo objetivo"), se genera un individuo mutante mediante la combinación lineal de otros individuos seleccionados aleatoriamente de la población. La generación del vector donante v_i se efectúa mediante la Ecuación

$$v_i = x_{r1} + F \cdot (x_{r2} - x_{r3}) \tag{4.1}$$

donde v_i es el individuo mutante, x_{r1}, x_{r2} y x_{r3} son individuos seleccionados aleatoriamente y F es un factor de escala que controla la amplitud de la mutación.

2. **Cruce**: Se combina el individuo mutante con el individuo objetivo para generar un individuo de prueba. El cruce se realiza generalmente mediante un operador de cruce binomial, donde cada componente del individuo de prueba se hereda del individuo mutante con una cierta probabilidad CR, y del individuo objetivo con probabilidad $1 - CR$.

$$u_{i,j} = \begin{cases} v_{i,j} & \text{si } \text{rand}_j(0,1) \leq CR \\ x_{i,j} & \text{de lo contrario} \end{cases}$$

donde $u_{i,j}$ es el j-ésimo componente del individuo de prueba, $v_{i,j}$ es el j-ésimo componente del individuo mutante, y $x_{i,j}$ es el j-ésimo componente del individuo objetivo.

3. **Selección**: Se evalúa la función objetivo para el individuo de prueba y se compara con el individuo objetivo. Si el individuo de prueba tiene un mejor valor de la función objetivo, reemplaza al individuo objetivo en la siguiente generación. De lo contrario, el individuo objetivo se mantiene.

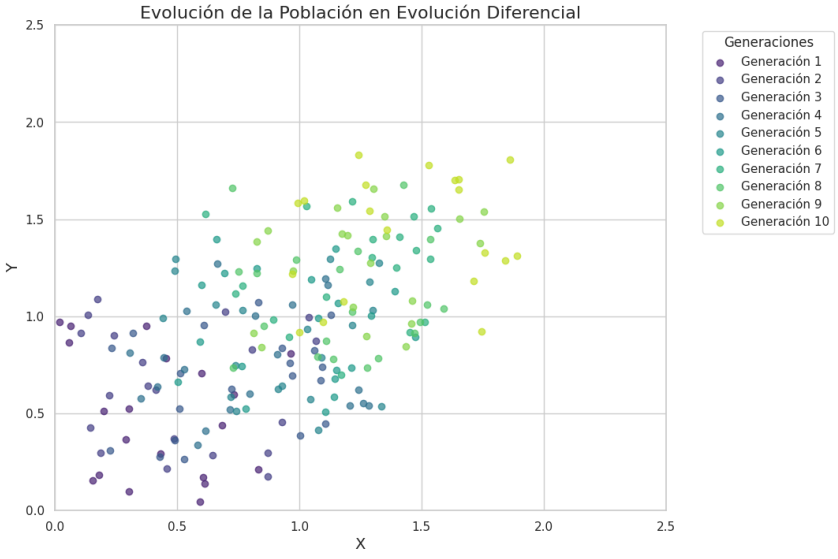

Figura 4.2: Evolución de la población en el algoritmo de Evolución Diferencial.

El proceso se repite hasta que se alcanza un número máximo de generaciones o se cumple un criterio de convergencia [45].

Ventajas y desventajas

Las principales ventajas de la Evolución Diferencial son:

- **Eficiencia**: Es un algoritmo relativamente rápido y eficiente, especialmente en problemas de alta dimensionalidad.
- **Robustez**: Es robusto a la presencia de ruido y a la forma de la función objetivo.
- **Facilidad de implementación**: Es relativamente fácil de implementar y tiene pocos parámetros de ajuste.

Sin embargo, DE también presenta algunas desventajas:

- **Sensibilidad a los parámetros**: Es altamente sensible a los parámetros de ajuste inicial y al punto de partida de inicio de búsqueda, el encontrar la combinación mas optima de hiperparmetros puede ser en si mismo un problema de optimización.
- **Convergencia prematura**: En la búsqueda del óptimo mediante población puede caer engañado rápidamente a un punto donde no tenga salida ni una búsqueda efectiva posterior, esto lleva a una rápida optimización pero a puntos que solo son solo valles, mínimos locales.

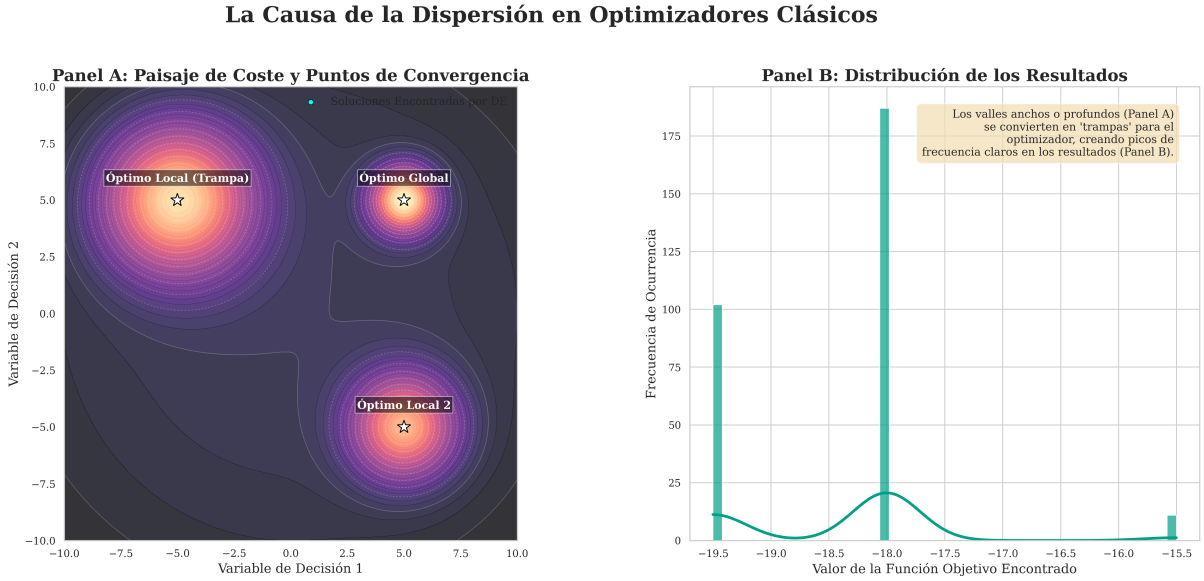

Figura 4.3: La visualización muestra el por qué los algoritmos de optimización a veces dan resultados diferentes en cada ejecución y como el método de optimización puede incluso quedar atrapado en mínimos locales de forma consistente para múltiples ejecuciones, este punto es de particular importancia en este trabajo pues muestra el fundamento esencial de por qué un algoritmo clásico es inherentemente imperfecto, dicho de otra forma, incluso en ejecuciones ideales puede caer en trampas locales lo que no le permite dar un resultado adecuado.

- **Costoso para funciones caras**: Dado que tiene que evaluar la función para miembro de la población, es computacionalmente costoso para funciones que requiera computo considerable la sola evaluación.

Minimización mediante procesos gaussianos (GP minimization)

La optimización bayesiana es una estrategia de optimización secuencial diseñada para funciones çaja negra"(black-box), es decir, funciones cuyo coste de evaluación es alto y de las cuales no se conoce su forma analítica ni sus derivadas [**?**]. El método 'gp_minimize', implementado en la biblioteca 'scikit-optimize' [42], es una herramienta robusta que materializa este enfoque utilizando Procesos Gaussianos (GP) como modelo subrogado.
Este método es particularmente adecuado para problemas complejos como la optimización de trayectorias, donde cada evaluación de la función objetivo (que podría implicar una simulación completa) es computacionalmente costosa. La idea central es construir un modelo probabilístico de la función objetivo y utilizarlo para seleccionar los puntos más prometedores a evaluar, equilibrando la exploración de nuevas regiones del espacio de búsqueda con la explotación de regiones ya conocidas por dar buenos resultados.

Fundamento Teórico

El proceso de 'gp_minimize' se basa en dos componentes clave: un modelo subrogado y una función de adquisición.

1. **Modelo Subrogado (Proceso Gaussiano - GP):** En lugar de trabajar directamente con la costosa función objetivo $f(x)$, la optimización bayesiana construye un modelo aproximado más barato, conocido como modelo subrogado. 'gp_minimize' utiliza un Proceso Gaussiano [40], que es un modelo no paramétrico que define una distribución de probabilidad sobre el conjunto de todas las posibles funciones que se ajustan a los datos observados. Un GP proporciona dos informaciones cruciales para cualquier punto x del espacio de búsqueda:

    - Una **predicción media ($\mu(x)$):** El valor más probable de la función en el punto x.

- Una **incertidumbre ($\sigma(x)$):** La varianza o desviación estándar asociada a esa predicción. La incertidumbre es baja en las regiones donde ya se han evaluado puntos y alta en las regiones inexploradas.

El GP se actualiza iterativamente con cada nueva evaluación de la función objetivo, volviéndose cada vez más preciso.

2. **Función de Adquisición:** Esta función utiliza las predicciones del GP ($\mu(x)$ y $\sigma(x)$) para determinar cuál es el siguiente punto más "útil"para evaluar. Su objetivo es cuantificar el beneficio esperado de evaluar la función en un punto x. La función de adquisición guía el equilibrio entre:

   - **Explotación (Exploitation):** Evaluar en puntos donde la media $\mu(x)$ es baja (es decir, donde el modelo predice un buen valor para la función objetivo).
   - **Exploración (Exploration):** Evaluar en puntos donde la incertidumbre $\sigma(x)$ es alta (es decir, en regiones del espacio de búsqueda poco conocidas).

   Existen varias funciones de adquisición, como la "Probabilidad de Mejora"(PI), la "Mejora Esperada"(EI), o el "Límite de Confianza Inferior"(LCB), que es una de las opciones por defecto en 'skopt'. La función LCB se define como:

$$\text{LCB}(x) = \mu(x) - \kappa\sigma(x) \tag{4.2}$$

   donde κ es un parámetro ajustable que controla el compromiso entre explotación (valores bajos de κ) y exploración (valores altos de κ). El algoritmo busca el punto que minimiza esta función de adquisición para elegir la siguiente evaluación.

El algoritmo general de 'gp_minimize' sigue un ciclo iterativo:

1. Se evalúa la función objetivo en un conjunto inicial de puntos (muestreados aleatoriamente o mediante una secuencia de baja discrepancia).
2. Se ajusta un Proceso Gaussiano a los puntos evaluados.
3. Se utiliza la función de adquisición para proponer el siguiente punto a evaluar.
4. Se evalúa la función objetivo en el nuevo punto.

5. Se añaden los nuevos datos al conjunto de observaciones y se repite desde el paso 2, hasta alcanzar un presupuesto de evaluaciones o un criterio de convergencia.

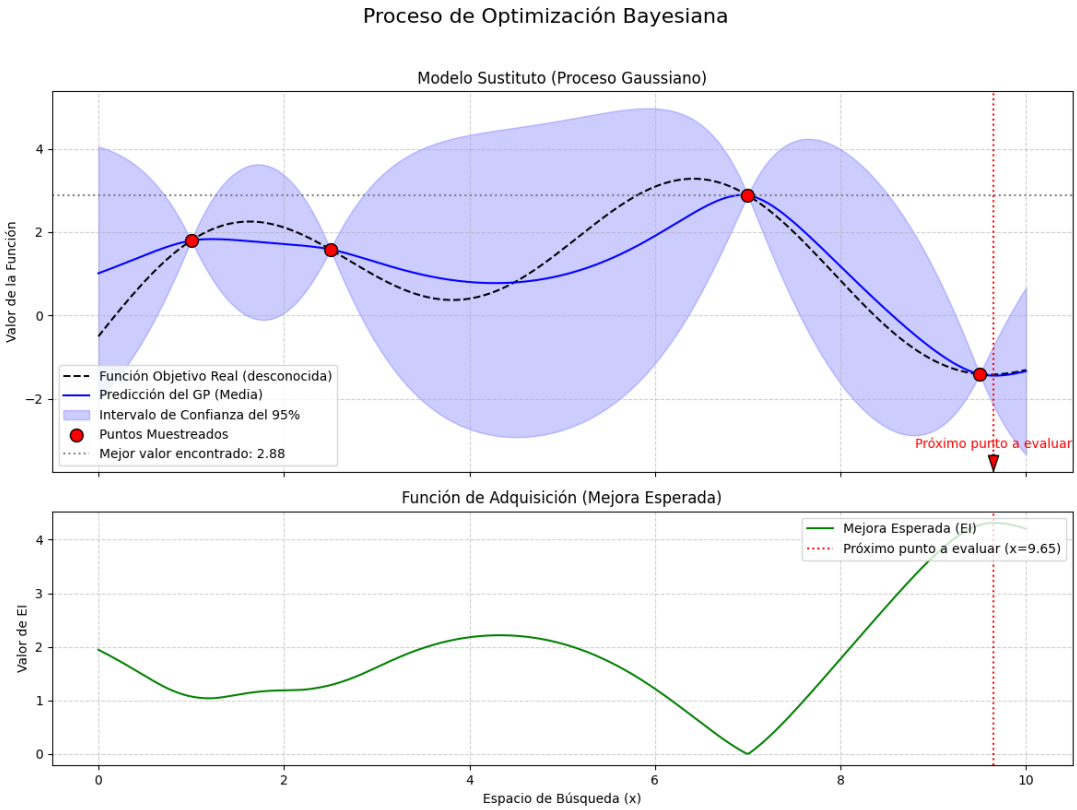

Figura 4.4: Ilustración del proceso de Optimización Bayesiana con ‘gp_minimize‘. El gráfico superior muestra la función objetivo real (línea punteada), las observaciones realizadas (puntos rojos), la predicción media del Proceso Gaussiano (línea azul) y su intervalo de confianza (área sombreada). El gráfico inferior muestra la función de adquisición (Mejora Esperada), cuyo máximo (línea vertical) indica el punto más prometedor para la siguiente evaluación, equilibrando regiones de bajo valor predicho y alta incertidumbre.

Ventajas y desventajas

Las principales ventajas de la optimización bayesiana con ‘gp_minimize‘ son:

- **Eficiencia de Muestreo:** Está diseñada para encontrar buenos resultados con un número mínimo de evaluaciones de la función objetivo, lo que es crucial para problemas costosos.
- **Manejo de Çaja Negra”:** No requiere información sobre el gradiente ni la convexidad de la función, lo que la hace muy versátil.
- **Balance Exploración-Explotación:** El uso de una función de adquisición permite

una búsqueda global inteligente, reduciendo el riesgo de quedar atrapado en mínimos locales.

Sin embargo, también presenta algunas desventajas:

- **Costo Computacional del Modelo:** Ajustar el Proceso Gaussiano tiene un coste computacional que escala cúbicamente con el número de puntos evaluados ($O(n^3)$). Esto puede hacer que el método sea menos eficiente si la función objetivo es muy barata de evaluar.
- **Sensibilidad a la Dimensionalidad:** Aunque es efectivo, su rendimiento puede degradarse en espacios de búsqueda de muy alta dimensionalidad (la "maldición de la dimensionalidad"), ya que cubrir el espacio se vuelve exponencialmente más difícil.
- **Ajuste de Hiperparámetros:** El rendimiento puede depender de la elección del kernel del Proceso Gaussiano y de los parámetros de la función de adquisición.

Muestra de rendimiento de los tres métodos revisados en el problema central de optimización

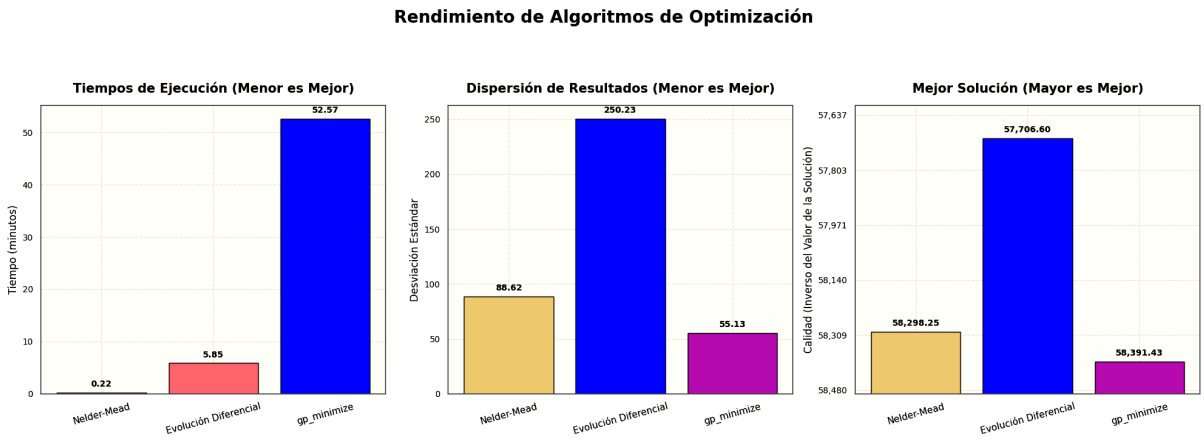

Figura 4.5: Comparativa de algoritmos en tiempo, dispersión y calidad, destacando la superioridad de Evolución Diferencial.

Al evaluar la función objetivo el problema central de optimización se obtuvieron diversos resultados, la gráfica muestra una comparativa del tiempo empleado, la dispersión de

los resultados y la calidad de la mejor solución encontrada por los tres métodos Nelder-Mead, evolución diferencial y optimización bayesiana (gp_minimize). La optimización se hizo para 10D lo que significa N=6 en el problema Airbus.

El propósito es que en condiciones semejantes se evalúen los 3 métodos. Sin embargo hay algunos matices, la optimización se ejecuta para Neider-Mead con una pequeña perturbación de tal forma que haya numerosos resultados y la comparativa en cuanto a su precisión se pueda hacer, por otra parte el costo computacional reflejado en el tiempo que tardaba cada método fue considerablemente mayor para el método de optimización bayesiana por esta razón y para que se pueda hacer una comparativa gráfica se le dio a este método menos oportunidad que a los otros 2 métodos.

En la gráfica finalmente se observa con claridad como a pesar de ejecutar un menor numero de veces el método de optimización bayesiana el tiempo es sustancialmente mayor, los tiempos para ejecutar el total del problema Airbus y teniendo presente que el enfoque es un problema real de 104 dimensiones este método es descartable en este punto por su excesivo costo computacional. Por otra parte, la gráfica que muestra la comparativa en cuanto a la dispersión muestra con claridad que el método Neider-Mead es mas consistente y presenta un comportamiento fiable determinista incluso después de agregarle artificialmente una perturbación en el código, esto comparado con evolución diferencial.

La tercera gráfica muestra una comparativa crucial y que permite finalmente elegir el método que se usará pues presenta el mejor valor objetivo (mínimo) alcanzado por cada optimizador en todas sus ejecuciones, evaluando su capacidad para encontrar una solución de más alta calidad. Esta evaluación se hizo durante muchos días y dando a cada método 1000 oportunidades de búsqueda, un numero mucho menor al método optimización bayesiana al sobrepasar considerablemente el tiempo disponible si una considerable y proporcional mejora en la calidad del punto encontrado [40]. Con lo anterior y teniendo en cuenta sus ventajas y desventajas en lo que sigue se usará evolución diferencial como el método de optimización clásica.

4.2. Métodos de muestreo

El muestreo constituye una componente relevante del proceso de optimización, en particular para funciones objetivo de alta dimensionalidad o complejidad elevada. Un método de muestreo apropiado debe permitir la exploración eficiente del espacio de soluciones y generar una representación fidedigna de la función objetivo. En esta sección se describen cuatro métodos de muestreo empleados en este estudio.

Muestreo basado en hexágonos

El muestreo basado en hexágonos constituye un método determinista cuyo objetivo es la cobertura uniforme del dominio de búsqueda mediante una teselación hexagonal. La selección de esta geometría se fundamenta en la propiedad del hexágono de posibilitar el empaquetamiento de círculos más denso en el plano bidimensional. Esta característica implica, en principio, una mayor densidad de cobertura espacial respecto a cuadrículas cartesianas estándar.

Fundamento teórico

Para un dominio bidimensional, los puntos de muestreo se disponen en los vértices y centros de los hexágonos regulares que conforman la teselación del área de interés. La posición de los puntos puede especificarse mediante coordenadas axiales (q, r), las cuales se transforman a coordenadas cartesianas (x, y) a través de las siguientes expresiones:

$$x = \text{width} \cdot (q \cdot 1{,}5) \tag{4.3}$$

$$y = \text{height} \cdot (r \cdot 2 + q) \tag{4.4}$$

donde *width* y *height* son el ancho y alto del hexágono, respectivamente.
Para generar una grilla hexagonal, se parte de un hexágono inicial y se generan iterativamente sus vecinos en todas las direcciones. Los vecinos de un hexágono con coordenadas (q, r) se pueden obtener añadiendo los siguientes desplazamientos:

$$(1,0),(1,-1),(0,-1),(-1,0),(-1,1),(0,1) \tag{4.5}$$

Este proceso se repite hasta que se cubre el área de interés o se alcanza un número máximo de puntos de muestreo.

Ventajas y desventajas

Las principales ventajas del muestreo basado en hexágonos son:

- **Uniformidad**: Proporciona una cobertura uniforme del espacio de búsqueda.
- **Eficiencia**: Es más eficiente que una grilla cuadrada en términos de la densidad de puntos de muestreo.
- **Estructura regular**: La estructura regular de la grilla hexagonal puede ser ventajosa para ciertos algoritmos de optimización o análisis.

Sin embargo, este método también tiene algunas desventajas:

- **Complejidad de implementación**: La generación de la grilla hexagonal y la gestión de los vecinos puede ser más compleja que en una grilla cuadrada.
- **Limitado a dos dimensiones**: La extensión a dimensiones superiores no es trivial y puede no ser tan eficiente como en el caso bidimensional.
- **Bordes irregulares**: La cobertura del espacio de búsqueda puede ser irregular en los bordes, dependiendo de la forma del área de interés.

Aplicabilidad al problema de optimización de la trayectoria de ascenso

El muestreo basado en hexágonos se utilizó en las etapas iniciales de este trabajo para explorar el espacio de de existencia de la función en dos dimensiones (por ejemplo, velocidad y ángulo de ascenso). Proporcionó una buena visualización de la función objetivo y permitió identificar regiones promisorias para la optimización. Sin embargo, su aplicabilidad a dimensiones superiores es limitada, y su complejidad de implementación motivó la exploración de otros métodos de muestreo más versátiles.

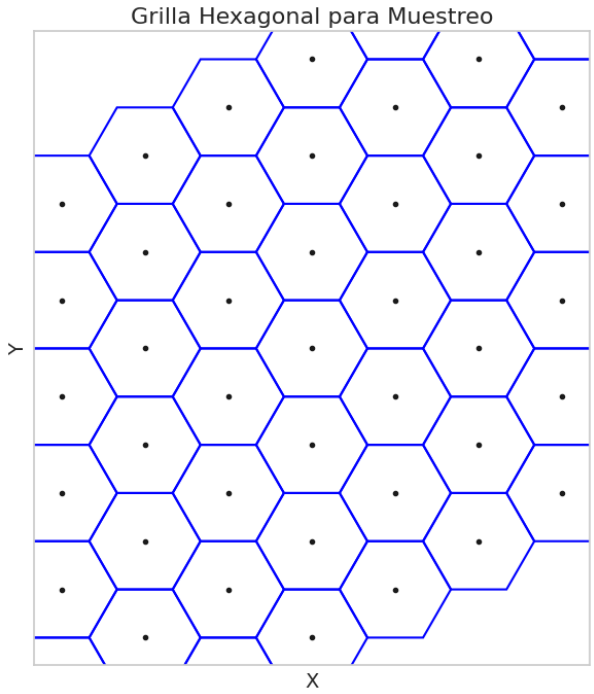

Figura 4.6: Grilla hexagonal utilizada para el muestreo basado en hexágonos.

Muestreo aleatorio con perturbaciones gaussianas

El muestreo aleatorio con perturbaciones gaussianas es un método estocástico que genera nuevos puntos de muestreo a partir de un punto existente, añadiendo perturbaciones aleatorias que siguen una distribución normal (gaussiana).

Fundamento Teórico

Dado un punto x en un espacio n-dimensional, se genera un nuevo punto x' mediante la siguiente fórmula:

$$x' = x + \mathcal{N}(\mu, \sigma^2) \tag{4.6}$$

donde $\mathcal{N}(\mu, \sigma^2)$ representa una distribución normal con media μ y varianza σ^2. Generalmente, se utiliza una media de cero ($\mu = 0$) y una desviación estándar σ que controla la magnitud de las perturbaciones.

El proceso de generación de nuevos puntos se puede repetir múltiples veces, partiendo de un punto inicial o de un conjunto de puntos previamente muestreados. Además, se

pueden introducir criterios de aceptación o rechazo de los nuevos puntos basados en el valor de la función objetivo o en restricciones del problema.

Ventajas y desventajas

Las principales ventajas del muestreo aleatorio con perturbaciones gaussianas son:

- **Simplicidad**: Es un método muy simple de implementar y entender.
- **Flexibilidad**: Se puede adaptar fácilmente a diferentes espacios de búsqueda y restricciones.
- **Eficiencia computacional**: La generación de nuevos puntos es computacionalmente económica.

Sin embargo, este método también tiene algunas desventajas:

- **No uniformidad**: El muestreo no es uniforme y puede resultar en una cobertura desigual del espacio de búsqueda.
- **Dependencia del punto inicial**: La exploración puede estar sesgada hacia la región donde se encuentra el punto inicial.
- **Dificultad para escapar de mínimos locales**: Si la desviación estándar de las perturbaciones es pequeña, el método puede tener dificultades para escapar de mínimos locales.

Aplicabilidad al problema de optimización de la trayectoria de ascenso

El muestreo aleatorio con perturbaciones gaussianas se utilizó en este trabajo como una forma rápida de generar puntos de prueba alrededor de soluciones prometedoras. Su simplicidad y eficiencia computacional lo hacen adecuado para una exploración inicial. Sin embargo, su falta de uniformidad y su dependencia del punto inicial limitan su efectividad como método principal de muestreo, especialmente para la aproximación de la función objetivo en altas dimensiones.

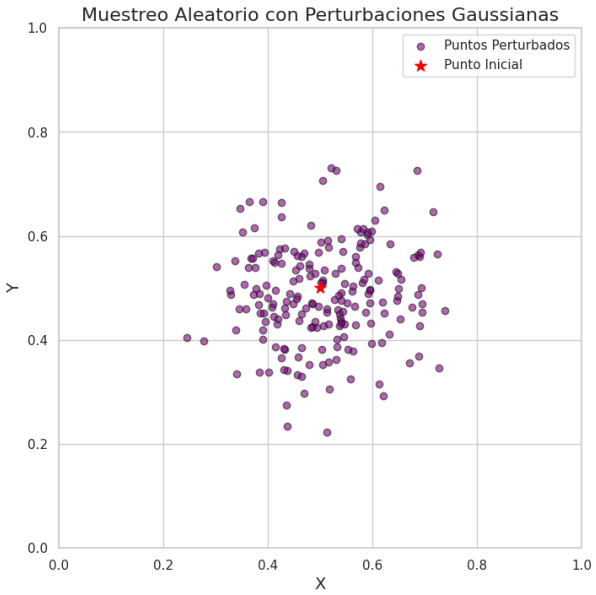

Figura 4.7: Muestreo aleatorio con perturbaciones gaussianas alrededor de un punto inicial.

Búsqueda de puntos adyacentes

La búsqueda de puntos adyacentes es un método iterativo que explora el espacio de soluciones generando y evaluando puntos vecinos a un conjunto de puntos previamente seleccionados. Este método se basa en la idea de que los puntos cercanos a una solución válida tienen una mayor probabilidad de ser también soluciones válidas o de proporcionar información útil sobre la función objetivo.

Fundamento Teórico

El algoritmo de búsqueda de puntos adyacentes se puede describir de la siguiente manera:

1. Se parte de un conjunto inicial de puntos, que pueden ser generados aleatoriamente o mediante algún otro método de muestreo.

2. Para cada punto en el conjunto actual, se generan sus vecinos. Los vecinos se definen como aquellos puntos que se encuentran a una cierta distancia del punto original, generalmente a lo largo de las direcciones de los ejes coordenados.

3. Se evalúa la función objetivo en cada uno de los puntos vecinos generados.

4. Los puntos vecinos que cumplen con los criterios de aceptación (por ejemplo, que el valor de la función objetivo esté dentro de un rango determinado o que satisfagan las restricciones del problema) se añaden a un nuevo conjunto de puntos.

5. Se repiten los pasos 2-4 utilizando el nuevo conjunto de puntos como punto de partida, hasta que se alcanza un número máximo de iteraciones o se cumple algún otro criterio de parada.

En este trabajo, se introdujeron dos parámetros adicionales para controlar la búsqueda:

- **mp (Preferencia Marginal):** Controla la preferencia por explorar puntos vecinos de aquellos puntos que han sido previamente rechazados. Un valor de $mp = 0$ indica que no hay preferencia, mientras que un valor de $mp = 1$ indica una preferencia absoluta por los vecinos de puntos rechazados. Valores intermedios de mp permiten un balance entre la exploración de nuevas regiones y la explotación de regiones cercanas a puntos rechazados.

- **mr (Preferencia de Radio):** Controla la preferencia por explorar puntos más alejados de un punto de referencia. Un valor de $mr = 0$ indica que no hay preferencia, mientras que un valor de $mr = 1$ indica una preferencia absoluta por los puntos más alejados. Valores intermedios de mr permiten un balance entre la exploración local y la exploración global.

Ventajas y desventajas

Las principales ventajas de la búsqueda de puntos adyacentes son:

- **Simplicidad**: Es un método conceptualmente simple y fácil de implementar.

- **Adaptabilidad**: Se puede adaptar a diferentes espacios de búsqueda y restricciones mediante la definición adecuada de la vecindad y los criterios de aceptación.

- **Control sobre la exploración**: Los parámetros mp y mr permiten controlar el balance entre la exploración local y global.

Sin embargo, este método también tiene algunas desventajas:

- **Costo computacional**: El número de vecinos a evaluar puede crecer exponencialmente con la dimensión del espacio de búsqueda.

- **Dependencia de la definición de vecindad**: La eficiencia y efectividad del método pueden depender fuertemente de la definición de la vecindad y de los parámetros mp y mr.

- **Limitaciones en la cobertura**: A pesar de la introducción de los parámetros mp y mr, el método puede tener dificultades para cubrir uniformemente todo el espacio de búsqueda, especialmente en altas dimensiones.

Aplicabilidad al problema de optimización de la trayectoria de ascenso

La búsqueda de puntos adyacentes se utilizó en este trabajo como un método para generar un conjunto de puntos válidos para la aproximación de la función objetivo. Su simplicidad y adaptabilidad permitieron explorar eficientemente el espacio de soluciones en torno a puntos iniciales prometedores. Los parámetros mp y mr se ajustaron para equilibrar la exploración local y global, y se encontró que una combinación de valores extremos ($mp = 0, mr = 0$ y $mp = 1, mr = 1$) producía los mejores resultados en términos de cobertura y eficiencia.

Sin embargo, el costo computacional de generar y evaluar todos los vecinos se volvió prohibitivo al aumentar la dimensionalidad del problema. Para N=53, el número de vecinos a evaluar en cada iteración sería del orden de 3^{104}, lo que hace que el método sea inviable para este caso.

Muestreo basado en secuencias de Sobol

El muestreo basado en secuencias de Sobol es un método cuasi-aleatorio que genera puntos de muestreo de baja discrepancia, lo que significa que están distribuidos de manera más uniforme en el espacio de búsqueda en comparación con un muestreo aleatorio simple. Las secuencias de Sobol son un tipo específico de secuencias de baja discrepancia que se construyen utilizando una base de 2 y permutaciones cuidadosamente elegidas de los bits de los enteros.

Fundamento Teórico

Las secuencias de Sobol se basan en la idea de generar una secuencia de puntos en un hipercubo unitario $[0, 1]^d$ de tal manera que la discrepancia entre la distribución empírica de los puntos y la distribución uniforme sea lo más baja posible. La discrepancia es una medida de la uniformidad de una distribución de puntos y se define como la máxima diferencia absoluta entre el volumen de una región rectangular y la fracción de puntos que caen en esa región.

La construcción de las secuencias de Sobol se realiza de forma recursiva, utilizando una serie de números de dirección v_i que se generan a partir de polinomios primitivos sobre el cuerpo finito $\mathbb{F}_2$. Para generar el i-ésimo punto x_i en una secuencia de Sobol de dimensión d, se utiliza la siguiente fórmula:

$$x_i = i_1 v_1 \oplus i_2 v_2 \oplus \cdots \oplus i_k v_k \tag{4.7}$$

donde i_j son los bits de la representación binaria del entero i, $\oplus$ es la operación XOR bit a bit, y k es el número de bits necesarios para representar i.
Los números de dirección v_i se calculan a partir de los coeficientes de un polinomio primitivo $p(x)$ de grado s sobre $\mathbb{F}_2$:

$$p(x) = x^s + a_1 x^{s-1} + \cdots + a_{s-1} x + 1 \tag{4.8}$$

donde $a_i \in \{0, 1\}$. Los primeros s números de dirección se eligen arbitrariamente, y los siguientes se calculan recursivamente utilizando la relación:

$$v_i = a_1 v_{i-1} \oplus a_2 v_{i-2} \oplus \cdots \oplus a_{s-1} v_{i-s+1} \oplus v_{i-s} \oplus \frac{v_{i-s}}{2^s} \tag{4.9}$$

para $i > s$.

Ventajas y desventajas

Las principales ventajas del muestreo basado en secuencias de Sobol son:

- **Baja discrepancia**: Las secuencias de Sobol tienen una discrepancia muy baja, lo que significa que los puntos de muestreo están distribuidos de manera muy uniforme en el espacio de búsqueda.
- **Eficiencia**: Son computacionalmente eficientes de generar, especialmente en comparación con otros métodos de muestreo de baja discrepancia.

- **Determinismo**: Son secuencias deterministas, lo que significa que la misma secuencia se generará siempre que se utilice la misma semilla. Esto facilita la reproducibilidad de los resultados.

Desventaja

- **Limitaciones en altas dimensiones**: En comparación con los otros métodos estudiados no encontramos desventajas aparte de la limitación computacional a alta dimensiones, pero esto seria independiente del método.

Aplicabilidad al problema de optimización de la trayectoria de ascenso

El muestreo basado en secuencias de Sobol se utilizó en este trabajo como el método principal para generar puntos de muestreo para la aproximación de la función objetivo en altas dimensiones (N=53). Su baja discrepancia y eficiencia computacional lo hacen adecuado para generar un gran número de puntos distribuidos uniformemente en el espacio de búsqueda.

Para el problema de la trayectoria de ascenso, se utilizó una implementación de las secuencias de Sobol disponible en la biblioteca ‘scipy.stats.qmc‘. Los puntos generados se escalaron y desplazaron para ajustarse a los rangos de las variables del problema. Además, se implementó un mecanismo de expansión iterativa del espacio de búsqueda para mejorar la cobertura y adaptarse a la forma de la función objetivo.

Los resultados experimentales mostraron que el muestreo basado en secuencias de Sobol permitió generar un conjunto de puntos válidos de alta calidad para la aproximación de la función objetivo, incluso en el caso de alta dimensionalidad (N=53). La uniformidad del muestreo y la eficiencia del algoritmo permitieron obtener una buena representación de la función objetivo con un número relativamente pequeño de puntos.

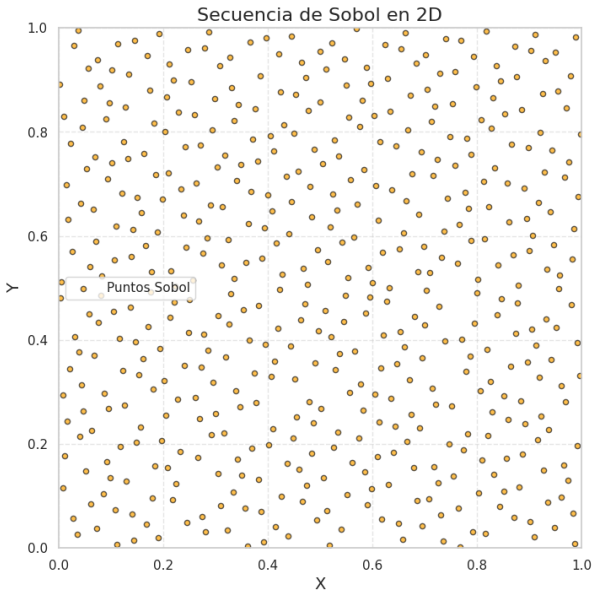

Figura 4.8: Distribución de puntos generados mediante secuencias de Sobol en un espacio bidimensional.

Capítulo 5

Optimización cuántica

El campo emergente de la computación cuántica promete revolucionar la forma en que abordamos problemas computacionales complejos. Un área de particular interés es la optimización, donde los algoritmos cuánticos podrían ofrecer ventajas significativas sobre los métodos clásicos. Este capítulo explora el fundamento teórico de la optimización cuántica, examinando los principios subyacentes, algoritmos clave y potenciales aplicaciones.

5.1. Principios básicos de computación cuántica

Antes de adentrarnos en la optimización, es crucial comprender los conceptos fundamentales de la computación cuántica. A diferencia de los bits clásicos, que representan 0 o 1, los qubits (bits cuánticos) pueden existir en una superposición de ambos estados simultáneamente. Esta propiedad, junto con otros fenómenos cuánticos como el entrelazamiento y la interferencia, permite a las computadoras cuánticas realizar cálculos que son inalcanzables para las computadoras clásicas.

Superposición

Un qubit puede representarse matemáticamente como una combinación lineal de los estados base $|0\rangle$ y $|1\rangle$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

donde α y β son amplitudes de probabilidad complejas que satisfacen $|\alpha|^2 + |\beta|^2 = 1$. La superposición permite explorar múltiples posibilidades simultáneamente, lo que

constituye la base para muchos algoritmos cuánticos.

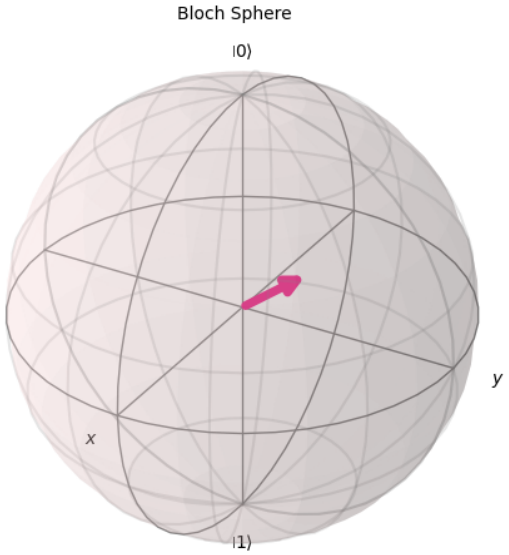

Figura 5.1: Representación de un qubit en la esfera de Bloch.

Como se describe en el libro de Sakurai [46], la esfera de Bloch es una representación geométrica útil para visualizar el estado de un qubit.

Entrelazamiento

El entrelazamiento es un fenómeno cuántico en el que dos o más qubits se correlacionan de tal forma que sus estados están intrínsecamente ligados, incluso si están separados por grandes distancias. Esta propiedad permite realizar operaciones conjuntas sobre múltiples qubits, aumentando la capacidad de procesamiento. Un ejemplo de estado entrelazado es el estado de Bell:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Puertas cuánticas

Las operaciones en computación cuántica se realizan mediante puertas cuánticas, que son análogas a las puertas lógicas en la computación clásica. Estas puertas manipulan los estados de los qubits y permiten implementar algoritmos cuánticos. Ejemplos de puertas cuánticas incluyen la puerta Hadamard, que crea superposiciones, y la puerta CNOT, que genera entrelazamiento.

La puerta Hadamard (H) transforma los estados base de la siguiente manera:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

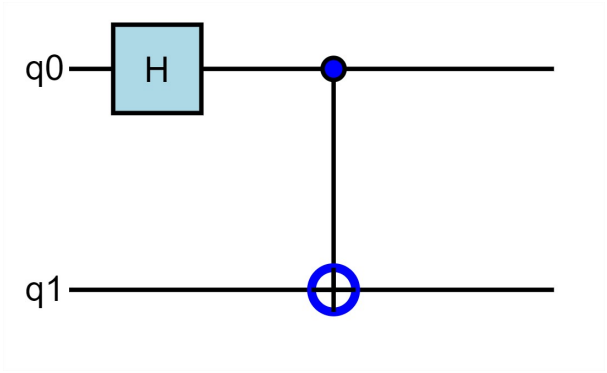

Figura 5.2: Circuito cuántico que ilustra la aplicación de una puerta Hadamard seguida de una puerta CNOT.

5.2. Algoritmos cuánticos para optimización

Diversos algoritmos cuánticos han sido propuestos para abordar problemas de optimización. Estos algoritmos aprovechan las propiedades únicas de la mecánica cuántica para potencialmente superar a los algoritmos clásicos en términos de eficiencia y velocidad.

Algoritmo variacional cuántico (VQE)

El Algoritmo Variacional Cuántico (VQE, por sus siglas en inglés) es un algoritmo híbrido cuántico-clásico diseñado para encontrar el estado fundamental de un sistema cuántico. Es ampliamente considerado como un candidato prometedor para resolver problemas de optimización. VQE opera en los siguientes pasos:

1. **Preparación de un estado prueba:** Un circuito cuántico parametrizado prepara un estado cuántico $|\psi(\theta)\rangle$, donde θ representa un conjunto de parámetros ajustables.

2. **Medición de la energía:** Se mide el valor esperado de la energía del sistema en el estado $|\psi(\theta)\rangle$.

$$\langle H\rangle = \langle\psi(\theta)|H|\psi(\theta)\rangle$$

donde H es el Hamiltoniano del sistema.

3. **Optimización clásica:** Un algoritmo de optimización clásico se utiliza para ajustar los parámetros θ con el objetivo de minimizar la energía.

Estos pasos se repiten iterativamente hasta que se alcanza la convergencia, produciendo una aproximación al estado fundamental y su energía asociada. La función objetivo del problema de optimización se codifica en el Hamiltoniano del sistema cuántico. VQE tiene aplicaciones en química cuántica, ciencia de materiales y optimización combinatoria.

Referencia: [15]

Recocido cuántico (Quantum Annealing)

El Recocido Cuántico es un método de optimización que utiliza fluctuaciones cuánticas para explorar el espacio de soluciones y encontrar el mínimo global de una función objetivo. Se inspira en el proceso de recocido en metalurgia, donde un material se calienta y luego se enfría lentamente para alcanzar un estado de baja energía.

En el recocido cuántico, el problema de optimización se mapea a un Hamiltoniano de Ising, que describe un sistema de espines interactuantes:

$$H = \sum_i h_i \sigma_i^z + \sum_{i<j} J_{ij} \sigma_i^z \sigma_j^z$$

donde h_i representa el campo magnético local en el espín i, J_{ij} es la interacción entre los espines i y j, y σ_i^z es la matriz de Pauli Z actuando sobre el espín i. El sistema se inicializa en un estado de superposición y luego se somete a un campo magnético transversal que disminuye gradualmente. Durante este proceso, el sistema evoluciona adiabáticamente hacia el estado fundamental del Hamiltoniano de Ising, que corresponde a la solución óptima del problema.

Referencia: [47]

Algoritmo de optimización aproximada cuántica (QAOA)

El Algoritmo de Optimización Aproximada Cuántica (QAOA) es otro algoritmo híbrido cuántico-clásico diseñado para resolver problemas de optimización combinatoria. QAOA emplea una secuencia de p capas unitarias parametrizadas, alternando entre operadores que codifican la función costo y operadores de mezcla que permiten la exploración del espacio de soluciones.

La función costo suele estar representada por un Hamiltoniano diagonal H_C, y el operador

de mezcla por un Hamiltoniano transversal $H_B = \sum_i \sigma_i^x$, donde σ_i^x es la matriz de Pauli X actuando sobre el qubit i.

El estado cuántico en QAOA se prepara aplicando alternadamente p veces las evoluciones unitarias generadas por H_C y H_B:

$$|\psi(\gamma, \beta)\rangle = U(H_B, \beta_p)U(H_C, \gamma_p) \cdots U(H_B, \beta_1)U(H_C, \gamma_1)|s\rangle$$

donde $|s\rangle$ es un estado inicial, típicamente una superposición uniforme de todos los estados posibles, y $\gamma = (\gamma_1, ..., \gamma_p)$, $\beta = (\beta_1, ..., \beta_p)$ son parámetros que se optimizan clásicamente para minimizar el valor esperado de la función costo:

$$F_p(\gamma, \beta) = \langle\psi(\gamma, \beta)|H_C|\psi(\gamma, \beta)\rangle$$

Cada capa i está compuesta por dos operadores unitarios:

* $U(H_C, \gamma_i) = e^{-i\gamma_i H_C}$: Evolución bajo el Hamiltoniano de costo H_C durante un tiempo γ_i. * $U(H_B, \beta_i) = e^{-i\beta_i H_B}$: Evolución bajo el Hamiltoniano de mezcla H_B durante un tiempo β_i.

El objetivo es encontrar los parámetros γ y β que minimizan el valor esperado de H_C en el estado $|\psi(\gamma, \beta)\rangle$. La optimización de estos parámetros se realiza mediante un algoritmo clásico, mientras que la evaluación de $F_p(\gamma, \beta)$ se realiza en un computador cuántico.

Modelo de Ising y QUBO

Muchos problemas de optimización combinatoria pueden formularse como un problema de **Optimización Cuadrática Binaria sin Restricciones (QUBO)**. En un problema QUBO, se busca minimizar una función objetivo cuadrática de variables binarias $x_i \in \{0, 1\}$:

$$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} Q_{ij} x_i x_j$$

donde Q es una matriz de coeficientes reales.

El modelo de Ising, que describe un sistema de espines interactuantes, está estrechamente relacionado con QUBO. De hecho, cualquier problema QUBO puede mapearse a un problema de Ising y viceversa. La relación entre las variables binarias x_i de QUBO y las variables de espín $s_i \in \{-1, 1\}$ del modelo de Ising es:

$$s_i = 2x_i - 1$$

Esta relación permite utilizar algoritmos como QAOA para resolver problemas formulados como QUBO, aprovechando la capacidad del algoritmo para encontrar el estado fundamental de un Hamiltoniano de Ising.

Referencias: [14, 48]

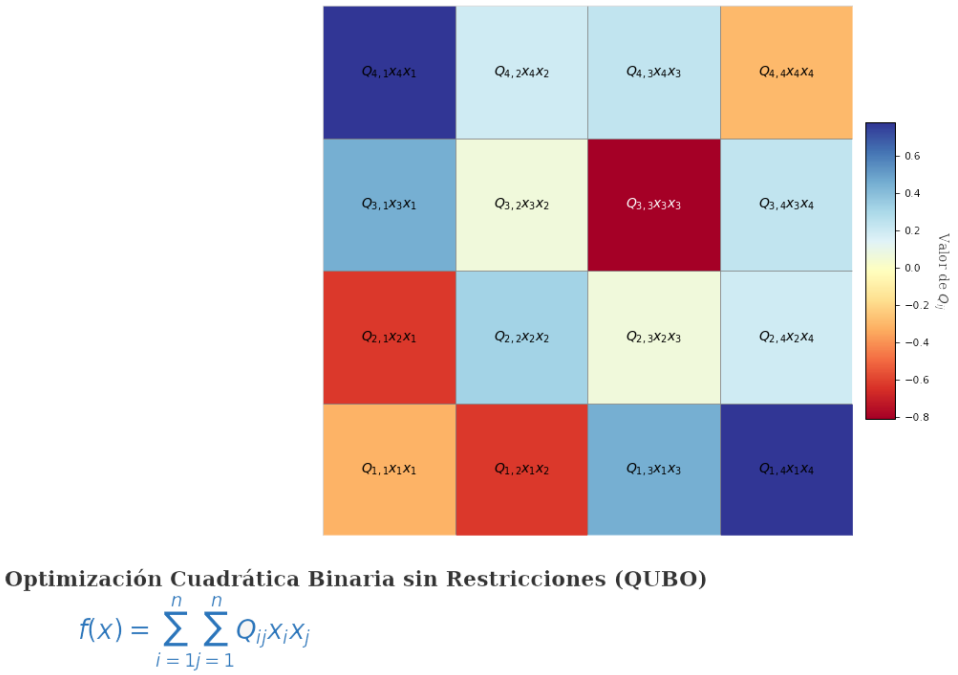

Figura 5.3: Función objetivo de un problema de Optimización Cuadrática Binaria sin Restricciones (QUBO). Donde x_i y x_j son variables binarias (0 o 1), y Q_{ij} son los coeficientes de una matriz simétrica que define las interacciones entre las variables. Cada cuadrado en la imagen representa un término cuadrático $Q_{ij}x_ix_j$ de la función objetivo. El color de cada cuadrado indica el valor del coeficiente Q_{ij}

5.3. Aplicación

La optimización cuántica tiene el potencial de impactar significativamente en una amplia gama de áreas, incluyendo:

- **Finanzas:** Optimización de carteras, gestión de riesgos, detección de fraude. [49]
- **Logística:** Optimización de rutas, gestión de inventarios, planificación de la cadena de suministro. [50]

- **Aprendizaje Automático:** Entrenamiento de redes neuronales, selección de características, agrupamiento de datos. [51]
- **Descubrimiento de Fármacos:** Diseño de moléculas, predicción de propiedades químicas, optimización de síntesis. [52]
- **Ciencia de Materiales:** Diseño de nuevos materiales, simulación de propiedades materiales, optimización de procesos de fabricación. [53]

Desafíos y perspectivas futuras

A pesar del enorme potencial, la optimización cuántica enfrenta varios desafíos. La construcción de computadoras cuánticas tolerantes a fallos es una tarea tecnológicamente compleja. Además, el desarrollo de algoritmos cuánticos eficientes y escalables requiere investigación continua.

El hardware cuántico actual es limitado en términos de tamaño y conectividad, ademas los algoritmos cuánticos como VQE y QAOA pueden verse afectados aumentando error por el hardware ruidoso.

No obstante, el campo de la computación cuántica está experimentando un rápido progreso. Se espera que las disponibilidades con hardware cuántico mejorado y el desarrollo de nuevos algoritmos impulsen avances significativos en la optimización cuántica en los próximos años.

Capítulo 6

Optimización de la trayectoria del Airbus A320

Este capítulo representa la aplicación de los marcos teóricos y metodológicos desarrollados en los capítulos precedentes. Se aborda directamente el desafío propuesto por Airbus para la optimización de la trayectoria de ascenso de la aeronave A320. Se detallará el proceso de formulación específica del problema, la aplicación de la reducción de dimensionalidad al modelo A320, las estrategias de muestreo adaptadas a la alta dimensionalidad resultante, el ajuste de la función objetivo para la computación cuántica y, finalmente, la ejecución y análisis comparativo de los enfoques de optimización clásico y cuántico sobre este problema aeronáutico real.

El primer paso para abordar la optimización de la trayectoria del A320 es definir formalmente el modelo matemático basado en las especificaciones del problema original y aplicar las técnicas de reducción de dimensionalidad introducidas en el Capítulo 3. Esta sección detalla cómo se particularizan las ecuaciones de la mecánica de vuelo (Capítulo 2) para el A320 y cómo mediante el análisis matematico del modelo es posible simplificar significativamente la complejidad del problema.

6.1. Modelo original del problema Airbus A320

El problema de optimización de la trayectoria de ascenso para el A320 fue planteado formalmente por Airbus como parte de su "Quantum Computing Challenge"[1]. Este planteamiento representa el modelo matemático completo que sirve como base para nuestro análisis y posterior simplificación.

La trayectoria de vuelo se optimiza entre un punto inicial I, donde la aeronave se encuentra en equilibrio ascendiendo con empuje MCL (Maximum Climb) a una velocidad CAS_I, y el primer nivel de crucero ubicado a la altitud Zp_F. El criterio de optimización es el costo ϕ de la trayectoria, que incorpora tanto el consumo de combustible como el tiempo de vuelo mediante el Índice de Costo CI:

$$\phi = \text{consumo} + CI \times \text{tiempo} \tag{6.1}$$

El problema matemático a resolver se formula como:

$$\min_{\{v_i,\gamma_i,m_i,t_i,s_i,Cz_i,\lambda_i\}_{1\le i\le N-1}} \phi(v_{N-1}, m_{N-1}, t_{N-1}, s_{N-1}, \lambda_{N-1}) \tag{6.2}$$

sujeto a las siguientes restricciones de igualdad, que representan las ecuaciones de movimiento discretizadas:

$$\begin{aligned} g_{v_i} = \frac{v_{i+1} - v_i}{Zp_{i+1} - Zp_i} - \frac{1}{2}\Bigg(&\frac{\lambda_{i+1}F_{N_{MCL_{i+1}}}}{m_{i+1}v_{i+1}\sin\gamma_{i+1}} - \frac{\frac{1}{2}\rho(Zp_{i+1})v_{i+1}S_{REF}(Cx_0 + kCz_{i+1}^2)}{m_{i+1}\sin\gamma_{i+1}} \\ &- \frac{g_0}{v_{i+1}} + \frac{\lambda_i F_{N_{MCL_i}}}{m_i v_i \sin\gamma_i} - \frac{\frac{1}{2}\rho(Zp_i)v_i S_{REF}(Cx_0 + kCz_i^2)}{m_i\sin\gamma_i} - \frac{g_0}{v_i}\Bigg) = 0 \quad {\scriptstyle (0\le i\le N-2)} \end{aligned} \tag{6.3}$$

$$\begin{aligned} g_{\gamma_i} = \frac{\gamma_{i+1} - \gamma_i}{Zp_{i+1} - Zp_i} - \frac{1}{2}\Bigg(&\frac{\frac{1}{2}\rho_{i+1}S_{REF}Cz_{i+1}}{m_{i+1}\sin\gamma_{i+1}} - \frac{g_0}{v_{i+1}^2\tan\gamma_{i+1}} + \frac{\frac{1}{2}\rho_i S_{REF}Cz_i}{m_i\sin\gamma_i} \\ &- \frac{g_0}{v_i^2\tan\gamma_i}\Bigg) = 0 \quad {\scriptstyle (0\le i\le N-2)} \end{aligned} \tag{6.4}$$

$$g_{m_i} = \frac{m_{i+1} - m_i}{Zp_{i+1} - Zp_i} + \frac{1}{2}\eta \left(\frac{\lambda_{i+1} F_{N_{MCL_{i+1}}}}{v_{i+1} \sin\gamma_{i+1}} + \frac{\lambda_i F_{N_{MCL_i}}}{v_i \sin\gamma_i} \right) = 0 \quad {\scriptstyle (0 \leq i \leq N-2)} \tag{6.5}$$

$$g_{t_i} = \frac{t_{i+1} - t_i}{Zp_{i+1} - Zp_i} - \frac{1}{2} \left(\frac{1}{v_{i+1} \sin\gamma_{i+1}} + \frac{1}{v_i \sin\gamma_i} \right) = 0 \quad {\scriptstyle (0 \leq i \leq N-2)} \tag{6.6}$$

$$g_{s_i} = \frac{s_{i+1} - s_i}{Zp_{i+1} - Zp_i} - \frac{1}{2} \left(\frac{1}{\tan\gamma_{i+1}} + \frac{1}{\tan\gamma_i} \right) = 0 \quad {\scriptstyle (0 \leq i \leq N-2)} \tag{6.7}$$

Estas ecuaciones son válidas para i desde 0 hasta $N-2$.
Adicionalmente, la trayectoria debe satisfacer un conjunto de restricciones operacionales y de seguridad en cada punto i (desde $i = 1$ hasta $N-1$):

$$g_{VMO_i} = CAS(v_i, Zp_i) - VMO \leq 0 \tag{6.8}$$

$$g_{MMO_i} = MACH(v_i, Zp_i) - MMO \leq 0 \tag{6.9}$$

$$g_{Vz_i} = Vz_{\min} - v_i \sin\gamma_i \leq 0 \tag{6.10}$$

$$Cz_i \leq Cz_{\max} \tag{6.11}$$

$$0 \leq \lambda_i \leq 1 \tag{6.12}$$

Estas inecuaciones aseguran que los limites de operación de la aeronave (límites de velocidad calibrada y Mach), mantenga una tasa mínima de ascenso, no exceda el coeficiente de sustentación máximo y utilice un nivel de empuje válido.
Las condiciones iniciales en el punto $i = 0$ (correspondiente a la altitud Zp_I) se definen como sigue:

$$v_0 = TAS(CAS_I, Zp_I) \tag{6.13}$$

$$\gamma_0 = \arcsin \left(\frac{F_{N_{MCL}}(Zp_0) - \frac{1}{2}\rho(Zp_0) v_0^2 S_{REF} \left[Cx_0 + k \left(\frac{m_0 g_0}{\frac{1}{2}\rho(Zp_0) v_0^2 S_{REF}} \right) \right]}{m_0 g_0} \right) \tag{6.14}$$

$$m_0 = m_I \tag{6.15}$$

$$t_0 = 0 \tag{6.16}$$

$$s_0 = 0 \tag{6.17}$$

$$Cz_0 = \frac{m_0 g_0}{\frac{1}{2}\rho(Zp_0)v_0^2 S_{REF}} \tag{6.18}$$

$$\lambda_0 = 1 \tag{6.19}$$

$$\tag{6.20}$$

efeq:airbus$_v$0a(6.19)),

Aquí, $Zp_0 = Zp_I$. La condición para γ_0 asume un estado equilibrado inicial con empuje máximo de ascenso (MCL).
La altitud en cada punto i se define mediante una discretización uniforme:

$$Zp_i = Zp_I + i\frac{Zp_F - Zp_I}{N-1} \quad \text{para } 0 \leq i \leq N-1 \tag{6.21}$$

El modelo utiliza varias funciones y parámetros auxiliares que definen las propiedades atmosféricas, aerodinámicas y de rendimiento del motor:

$$F_{N_{MCL_i}} = F_{N_{MCL}}(Zp_i) \quad \text{(Empuje máximo de ascenso)} \tag{6.22}$$

$$\rho_i = \rho(Zp_i) = \rho_0 \left(\frac{Ts_0 + L_Z Zp_i}{Ts_0}\right)^{\alpha_0 - 1} \quad \text{(Densidad del aire)} \tag{6.23}$$

$$M(v, Zp) = \frac{v}{\sqrt{\gamma_{air} R(Ts_0 + L_Z Zp)}} \quad \text{(Número de Mach)} \tag{6.24}$$

$$CAS(v, Zp) = \left[7RTs_0\left(\left(\left(\frac{Ts_0 + L_Z Zp}{Ts_0}\right)^{\alpha_0}\left(\left(1 + \frac{v^2}{7R(Ts_0 + L_Z Zp)}\right)^{3,5} - 1\right) + 1\right)^{1/3,5} - 1\ )\right]^{1/2} \tag{6.25}$$

$$TAS(CAS, Zp) = \left[7R(Ts_0 + L_Z Zp)\left(\left(\left(\frac{Ts_0 + L_Z Zp}{Ts_0}\right)^{-\alpha_0}\left(\left(1 + \frac{CAS^2}{7RTs_0}\right)^{3,5} - 1\right) + 1\right)^{1/3,5} - 1\ )\right]^{1/2} \tag{6.26}$$

Donde las constantes atmosféricas estándar (ISA) utilizadas son:

$$Ts_0 = 288{,}15\ K, \quad \rho_0 = 1{,}225\ kg \cdot m^{-3} \tag{6.27}$$

$$L_Z = -0{,}0065\ K \cdot m^{-1}, \quad g_0 = 9{,}80665\ m \cdot s^{-2} \tag{6.28}$$

$$\alpha_0 = -\frac{g_0}{RL_Z}, \quad R = 287{,}05287\ N \cdot m \cdot kg^{-1} \cdot K^{-1} \tag{6.29}$$

La función de costo ϕ se calcula considerando no solo la fase de ascenso, sino también una fase de aceleración a altitud constante (Zp_F) hasta alcanzar la velocidad de crucero (M_{CRZ}), y una fase de crucero hasta completar una distancia total L. El cálculo detallado implica determinar el estado al final de la aceleración (punto B) y luego el estado al final del crucero (punto F). Con ρ_F y siendo v_F la velocidad TAS correspondiente a M_{CRZ} en la altitud Zp_F:

$$\rho_F = \rho_0 \left(\frac{Ts_0 + L_Z Zp_F}{Ts_0}\right)^{-\left(\frac{g_0}{RL_Z}+1\right)} \tag{6.30}$$

$$v_F = M_{CRZ}\sqrt{1{,}4R(Ts_0 + L_Z Zp_F)} \tag{6.31}$$

Se definen las constantes A, B, C, D para la fase de aceleración:

$$A = -\frac{\rho_F S_{REF} Cx_0}{2m_{N-1}} - \frac{6km_{N-1}g_0^2}{\rho_F S_{REF} v_{N-1}^4} \tag{6.32}$$

$$B = \frac{16km_{N-1}g_0^2}{\rho_F S_{REF} v_{N-1}^3} \tag{6.33}$$

$$C = \frac{F_{N_{MCL_{N-1}}}}{m_{N-1}} - \frac{12km_{N-1}g_0^2}{\rho_F S_{REF} v_{N-1}^2} \tag{6.34}$$

$$D = \sqrt{B^2 - 4AC} \tag{6.35}$$

Usando estos coeficientes, se calcula el tiempo t_B masa m_B y distancia s_B acumulados al finalizar la fase de aceleración hasta v_F

$$t_B = t_{N-1} + \frac{2}{D}\left[\tanh^{-1}\left(\frac{2Av_{N-1}+B}{D}\right) - \tanh^{-1}\left(\frac{2Av_F+B}{D}\right)\right] \tag{6.36}$$

$$m_B = m_{N-1} - \eta\lambda_{N-1}F_{N_{MCL_{N-1}}}(t_B - t_{N-1}) \tag{6.37}$$

$$s_B = s_{N-1} + \frac{1}{A}\log\left(\frac{D-2Av_F-B}{D-2Av_{N-1}-B}\right) - \frac{B+D}{2A}(t_B - t_{N-1}) \tag{6.38}$$

Finalmente se modela el segmento de crucero que permite a la aeronave alcanzar la distancia total L. Durante esta fase, caracterizada por un vuelo a altitud Zp_F y velocidad v_F constantes, el consumo de combustible provoca una disminución de la masa. Partiendo de la masa m_B y el tiempo t_B al inicio de esta fase (punto B), la masa m_F y el tiempo t_F correspondientes a la finalización de la distancia $s_F = L$ se calculan como sigue:

$$m_F = m_B e^{-\frac{2\eta g_0\sqrt{kCx_0}}{v_F}(s_F - s_B)} \tag{6.39}$$

$$t_F = t_B + \frac{s_F - s_B}{v_F} \tag{6.40}$$

La función de costo final, que combina el consumo total de combustible y el tiempo total t_F ponderado por el Índice de Costo CI, se define como:

$$\boxed{\phi(v_{N-1}, m_{N-1}, t_{N-1}, s_{N-1}, \lambda_{N-1}) = -m_F + CI\left(t_B - \frac{s_B}{v_F}\right)} \tag{6.41}$$

Características del problema:

Seleccionando $N = 53$, correspondiente a una discretización aproximada de 500 ft entre $Zp_I = 10000$ ft y $Zp_F = 36000$ ft, el problema se convierte en:

- Número de variables: $7 \times (N-1) = 7 \times 52 = 364$. Estas variables son: $\{v_i, \gamma_i, m_i, t_i, s_i, Cz_i, \lambda_i\}$ para $i = 1, \ldots, 52$.

- Número de restricciones de igualdad (ecuaciones dinámicas): $5 \times (N-1) = 5 \times 52 = 260$. Estas son las ecuaciones (6.3) a (6.7).
- Número de restricciones de desigualdad (límites operacionales): $5 \times (N-1) = 5 \times 52 = 260$. Estas son las ecuaciones (6.8) a (6.12).
- Total de restricciones: $260 + 260 = 520$.

Se trata de un problema de optimización no lineal de gran escala con restricciones.

Parámetros del problema:

Los valores específicos para los parámetros del modelo A320 se consignan en la siguiente tabla:

Nº	**Símbolo**	**Definición**	**Valor**	**Unidad SI**
1	Cx_0	Coeficiente de la polar aerodinámica	0.014	-
2	k	Coeficiente de la polar aerodinámica	0.09	
3	Cz_{max}	Coeficiente de sustentación máximo	0.7	-
4	S_{REF}	Superficie de referencia aerodinámica	120	m^2 [L^2]
5	η	Consumo específico de combustible	0.06	kg(N.h)$^{-1}$ [$L^{-1}T$]
6	Zp_I	Altitud inicial	10000	ft [L]
7	Zp_F	Altitud final	36000	ft [L]
8	m_I	Masa inicial	60000	kg [M]
9	CAS_I	Velocidad Calibrada (CAS) inicial	250	kt [LT^{-1}]
10	VMO	Velocidad CAS máxima operativa	350	kt [LT^{-1}]
11	MMO	Número de Mach máximo operativo	0.82	-
12	M_{CRZ}	Número de Mach de crucero	0.80	-
13	$L(= s_F)$	Longitud total de la trayectoria	400	km [L]
14	Vz_{min}	Velocidad de ascenso mínima	300	ft.min^{-1} [LT^{-1}]
15	g_0	Aceleración de la gravedad	9.80665	m.s^{-2} [LT^{-2}]
16	CI	Índice de Costo	30	kg.min^{-1} [MT^{-1}]

El empuje máximo de ascenso ($F_{N_{MCL}}$), dependiente de la altitud Zp (en metros), se modela como:

$$F_{N_{MCL}}(Zp) = 140000 - 2{,}53 \cdot Zp \tag{6.42}$$

donde $F_{N_{MCL}}$ se expresa en Newtons (N).
Las conversiones de unidades utilizadas son:

$$1 \text{ ft} = 0{,}3048 \text{ m} \tag{6.43}$$

$$1 \text{ kt} = \frac{1852}{3600} \text{ m/s} \approx 0{,}5144 \text{ m/s} \tag{6.44}$$

6.2. Análisis de independencia diferencial entre las relaciones del sistema

El modelo de optimización de la trayectoria de ascenso del A320, descrito en la Sección 6.1, se compone de un conjunto de variables de estado y control interconectadas a través de ecuaciones dinámicas discretizadas. Específicamente, para una discretización de $N = 53$ puntos, el sistema involucra $7 \times (N-1) = 364$ variables: $\{v_i, \gamma_i, m_i, t_i, s_i, Cz_i, \lambda_i\}$ para $i = 1, \ldots, 52$. Estas variables están sujetas a $5 \times (N-1) = 260$ restricciones de igualdad dadas por las ecuaciones (6.3) a (6.7), las cuales representan la evolución del estado de la aeronave entre puntos consecutivos de la trayectoria.
Un número elevado de variables y restricciones puede incrementar significativamente la complejidad computacional de los algoritmos de optimización. Por lo tanto, resulta fundamental determinar si existen dependencias inherentes entre las ecuaciones del sistema que permitan reducir la dimensionalidad efectiva del problema. Este análisis se fundamenta en los principios del Teorema de la Función Implícita (discutido en el Capítulo 3), el cual establece condiciones bajo las cuales un sistema de ecuaciones define implícitamente algunas variables como funciones de otras. Una condición clave es la invertibilidad (o el rango) de la matriz Jacobiana asociada al sistema.
Aunque el objetivo no es resolver explícitamente para un subconjunto de variables, el análisis del Jacobiano del sistema de ecuaciones de actualización de estado ((6.3) a (6.7))

proporciona información cuantitativa sobre el número de grados de libertad reales del sistema, es decir, el número de variables independientes.

Construcción de la matriz jacobiana:

La matriz jacobiana J para un sistema se describe en el capitulo 3, y para este sistema en particular la matriz Jacobiana J se define como la matriz de las derivadas parciales de las $5 \times (N-1) = 260$ funciones de restricción de igualdad, $g_{v_i}, g_{\gamma_i}, g_{m_i}, g_{t_i}, g_{s_i}$ para $i = 0, \ldots, N-2$, con respecto a las $7 \times (N-1) = 364$ variables, $v_j, \gamma_j, m_j, t_j, s_j, Cz_j, \lambda_j$ para $j = 1, \ldots, N-1$.

La estructura de esta matriz J, de dimensiones 260×364, es:

$$J = \begin{bmatrix} \frac{\partial g_v(0)}{\partial v(1)} & \frac{\partial g_v(0)}{\partial v(2)} & \cdots & \frac{\partial g_v(0)}{\partial \gamma(1)} & \cdots & \frac{\partial g_v(0)}{\partial \lambda(N-1)} \\ \frac{\partial g_v(1)}{\partial v(1)} & \frac{\partial g_v(1)}{\partial v(2)} & \cdots & \frac{\partial g_v(1)}{\partial \gamma(1)} & \cdots & \frac{\partial g_v(1)}{\partial \lambda(N-1)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial g_\gamma(0)}{\partial v(1)} & \frac{\partial g_\gamma(0)}{\partial v(2)} & \cdots & \frac{\partial g_\gamma(0)}{\partial \gamma(1)} & \cdots & \frac{\partial g_\gamma(0)}{\partial \lambda(N-1)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial g_s(N-2)}{\partial v(1)} & \frac{\partial g_s(N-2)}{\partial v(2)} & \cdots & \frac{\partial g_s(N-2)}{\partial \gamma(1)} & \cdots & \frac{\partial g_s(N-2)}{\partial \lambda(N-1)} \end{bmatrix} \tag{6.45}$$

Cada una de las 260 filas corresponde a una de las ecuaciones de restricción de igualdad g_v, g_γ, g_m, g_t, o g_s, e i variando de 0 a $N-2 = 51$. Estas ecuaciones representan la dinámica que conecta el estado de la aeronave entre el punto i y el punto $i+1$. Cada una de las 364 columnas corresponde a una de las variables de optimización v, γ, m, t, s, Cz, o λ, y el índice j varíando de 1 a $N-1 = 52$. Las variables en el punto inicial $i = 0$ $(v_0, \gamma_0, m_0, t_0, s_0, Cz_0, \lambda_0)$ son condiciones fijas determinadas por los parámetros iniciales del problema y, por lo tanto, no forman parte de las variables respecto a las cuales se diferencia; no constituyen columnas en el Jacobiano J.

$$J_{(\text{func},i),(\text{var},j)} = \frac{\partial g_{\text{func}}(i)}{\partial \text{var}(j)}$$

Cálculo numérico del rango jacobiano:

Debido a la complejidad algebraica de las ecuaciones (6.3) a (6.7), el cálculo simbólico completo del rango del Jacobiano es intratable. En su lugar, se adopta un enfoque numérico:

1. **Diferenciación Simbólica:** Se utiliza una biblioteca de cálculo simbólico (como SymPy en Python) para obtener las expresiones analíticas de cada derivada parcial $\frac{\partial g_{\text{func}}(i)}{\partial \text{var}(j)}$. Este proceso se implementó mediante una función que toma la ecuación simbólica, la variable de diferenciación y los índices correspondientes.

2. **Evaluación Numérica:** Se define un punto específico en el espacio de variables.

   Esto implica asignar valores numéricos a todas las constantes del problema (Tabla Sección 6.5) y generar una trayectoria de ejemplo factible que proporcione valores para $v_i, \gamma_i, \ldots, \lambda_i$ en cada punto $i = 0, \ldots, N-1$. Los valores iniciales ($i = 0$) se calculan a partir de las condiciones dadas ((6.13) a (6.19)), y los valores para $i = 1, \ldots, N-1$ provienen de asignación de valores factibles.

3. **Sustitución y Construcción de Matriz Numérica:** Las expresiones simbólicas de las derivadas parciales se evalúan numéricamente sustituyendo los valores de las constantes y las variables en el punto elegido. Con estos valores numéricos, se construye la matriz Jacobiana J_{num} de 260×364.

4. **Cálculo del Rango:** Se calcula el rango de la matriz numérica J_{num} utilizando métodos numéricos estándar, como la descomposición en valores singulares (SVD), implementados en bibliotecas como NumPy (‘numpy.linalg.matrix_rank‘).

El pseudocódigo (1) ilustra el proceso de cálculo del Jacobiano numérico.

Resultados del análisis y determinación de variables independientes:

Aplicando el procedimiento descrito y utilizando los valores numéricos de los parámetros y una trayectoria de ejemplo para las variables, se obtuvo que para $N = 53$ el valor del rango del Jacobiano es $\boxed{260}$

Algorithm 1 Cálculo del Jacobiano Numérico

Entrada: Dimensión N, valores numéricos de variables y constantes
Salida: Matriz jacobiana numérica
/ Definición previa de funciones simbólicas */*
$g_v, g_\gamma, g_m, g_t, g_s \leftarrow$ Funciones simbólicas
$v(i), \gamma(i), m(i), t(i), s(i), C_z(i), \lambda(i) \leftarrow$ Variables simbólicas
$Z_{p_I}, C_{x_0}, \ldots \leftarrow$ Constantes simbólicas
function CalcularDerivadaParcial(función, variable, i_{func}, j_{var})
    $\partial \leftarrow \frac{\partial}{\partial \text{variable}(j_{\text{var}})}[\text{función}(i = i_{\text{func}})]$ ▷ Derivada simbólica
    return ∂
end function
function CalcularJacobianoNumérico(N, valVariables, valoresConstantes)
    numFunciones $\leftarrow 5 \times (N-1)$
    numVariables $\leftarrow 7 \times (N-1)$
    $J_{\text{simb}} \leftarrow$ Matriz($numFunciones, numVariables$) ▷ Inicializada con ceros
    funciones $\leftarrow [g_v, g_\gamma, g_m, g_t, g_s]$
    variables $\leftarrow [v, \gamma, m, t, s, C_z, \lambda]$
    for $i_{\text{func}} = 0 \ldots N-2$ **do**
        for $j_{\text{var\_idx}} = 0 \ldots N-2$ **do**
            $j_{\text{var}} \leftarrow j_{\text{var\_idx}} + 1$ ▷ Índice real de la variable
            for $k_{\text{func}} = 0 \ldots |\text{funciones}| - 1$ **do**
                for $l_{\text{var}} = 0 \ldots |\text{variables}| - 1$ **do**
                    func $\leftarrow$ funciones[k_{func}]
                    var $\leftarrow$ variables[l_{var}]
                    derivada $\leftarrow$ CalcularDerivadaParcial(func, var, i_{func}, j_{var})
                    fila $\leftarrow i_{\text{func}} \times |\text{funciones}| + k_{\text{func}}$
                    columna $\leftarrow j_{\text{var\_idx}} \times |\text{variables}| + l_{\text{var}}$
                    J_{simb}[fila, columna] $\leftarrow$ derivada
                end for
            end for
        end for
    end for
    $J_{\text{eval}} \leftarrow$ Sustituir valoresVariables y valoresConstantes en J_{simb}
    $J_{\text{final}} \leftarrow$ Convertir J_{eval} a matriz numérica
    return J_{final}
end function
/ Ejemplo de uso */*
valoresVars $\leftarrow \{v(1) = \ldots, \gamma(1) = \ldots, \ldots, \lambda(N-1) = \ldots\}$
valoresConst $\leftarrow \{Z_{p_I} = \ldots, C_{x_0} = \ldots, \ldots\}$
$J_{\text{num}} \leftarrow$ CalcularJacobianoNumérico(53, valoresVars, valoresConst)
rangoJ $\leftarrow$ CalcularRangoMatriz(J_{num})

El rango entonces, de la matriz Jacobiana evaluada numéricamente es 260. Resultado que tiene una implicación directa: de las 260 ecuaciones de actualización de estado que definen las restricciones de igualdad del sistema, todas son linealmente independientes. El número de variables independientes o grados de libertad del sistema se determina entonces como la diferencia entre el número total de variables y el rango del Jacobiano que representa el número de restricciones linealmente independientes efectivas.

$$\text{Variables Independientes} = \text{Número Total de Variables} - \text{Rango}(J) \tag{6.46}$$

$$\text{Variables Independientes} = 364 - 260 = 104 \tag{6.47}$$

Este cálculo revela que, aunque el problema se formula inicialmente con 364 variables, las 260 ecuaciones de estado interdependientes reducen el número de grados de libertad a 104. Esto significa que la trayectoria completa de ascenso, definida por las 364 variables, puede ser determinada unívocamente especificando los valores de un subconjunto de 104 variables independientes, siempre que se satisfagan las ecuaciones de estado.

Implicaciones para la optimización:
La determinación de que existen solo 104 variables independientes tiene consecuencias significativas para el proceso de optimización:

- **Reducción de Dimensionalidad:** El espacio de búsqueda efectivo para la optimización se reduce de 364 a 104 dimensiones. Esto mitiga considerablemente la "maldición de la dimensionalidad" y hacer que los algoritmos de optimización (tanto clásicos como cuánticos) sean más eficientes y tengan mayor probabilidad de encontrar soluciones de alta calidad.

- **Selección de Variables de Optimización:** En lugar de optimizar directamente sobre las 364 variables, se puede elegir un conjunto de 104 variables como las variables de decisión primarias. Las restantes 260 variables se calcularían entonces resolviendo (implícita o explícitamente) el sistema de ecuaciones de estado. Una elección común podría ser optimizar sobre las variables de control (λ_i, Cz_i u otras combinaciones) en cada segmento.

- **Enfoque de Muestreo y Ajuste:** Las estrategias de muestreo del espacio de

soluciones y el ajuste posterior de la función objetivo (discutidas en las secciones siguientes) pueden centrarse en este espacio reducido de 104 dimensiones, simplificando la tarea de aproximar la función de costo ϕ.

Es importante notar que el rango se calculó numéricamente en un punto específico. Sin embargo, dada la naturaleza física del sistema, se espera que este rango sea constante en la mayor parte del dominio factible. La identificación de 104 variables independientes proporciona la base teórica para simplificar el enfoque de optimización para el problema de ascenso del A320.

6.3. Reducción de dimensionalidad del modelo de Airbus

El análisis de independencia diferencial (Sección 6.2), fundamentado en el cálculo numérico del rango de la matriz Jacobiana asociada al sistema de ecuaciones dinámicas discretizadas (6.3) a (6.7), reveló una propiedad estructural fundamental del modelo de ascenso del A320. Para $N = 53$, el sistema consta de $M = 5 \times (N - 1) = 260$ ecuaciones de igualdad que relacionan un total de $P = 7 \times (N - 1) = 364$ variables de estado y control $\{v_i, \gamma_i, m_i, t_i, s_i, Cz_i, \lambda_i\}_{i=1}^{N-1}$. El rango calculado del Jacobiano, $\text{rank}(J) = 260 = M$, confirma que las 260 ecuaciones son linealmente independientes en el punto evaluado.
Según el Teorema de la Función Implícita (Capítulo 3), un sistema de M ecuaciones independientes con P variables define localmente M variables dependientes como funciones de las $P - M$ variables independientes restantes. En este caso, el número de grados de libertad del sistema es $P - M = 364 - 260 = 104$. Esto implica que la trayectoria completa, definida por las 364 variables, está unívocamente determinada por la especificación de un subconjunto de 104 variables independientes, siempre que se satisfagan las 260 ecuaciones de estado.

Selección de variables independientes y estructura iterativa:
Una elección estratégica para las variables independientes, motivada por su rol en la definición de la cinemática de la trayectoria, consiste en seleccionar las velocidades verdaderas $\{v_i\}_{i=1}^{N-1}$ y los ángulos de ascenso $\{\gamma_i\}_{i=1}^{N-1}$ para todos los puntos desde $i = 1$ hasta $N - 1$. Este conjunto precisamente contiene $2 \times (N - 1) = 104$ variables.

Con esta elección, el problema se reestructura como un proceso iterativo. Dado el estado completo de la aeronave en el punto i, denotado por $\mathbf{s}_i = (v_i, \gamma_i, m_i, t_i, s_i, Cz_i, \lambda_i)$, y especificando los valores de las variables independientes en el siguiente punto, v_{i+1} y γ_{i+1}, el objetivo es calcular las variables dependientes restantes en el punto $i+1$, es decir, $\mathbf{d}_{i+1} = (m_{i+1}, t_{i+1}, s_{i+1}, Cz_{i+1}, \lambda_{i+1})$. Este cálculo se realiza para i desde 0 hasta $N-2$, comenzando con el estado inicial $\mathbf{s}_0$ completamente determinado por las condiciones iniciales (6.13) a (6.19).

El procedimiento matemático para calcular $\mathbf{d}_{i+1}$ a partir de $\mathbf{s}_i$ y (v_{i+1}, γ_{i+1}) se basa en la manipulación algebraica y la resolución analítica de las ecuaciones de estado:

Paso 1: Expresión explícita para λ_{i+1}

La ecuación de conservación de masa $g_{m_i} = 0$ (ec. (6.5)) relaciona la variación de masa con el empuje aplicado. Despejando λ_{i+1} se obtiene:

$$\frac{1}{2}\eta\frac{\lambda_{i+1}F_{N_{MCL_{i+1}}}}{v_{i+1}\sin\gamma_{i+1}} = -\left(\frac{m_{i+1}-m_i}{Zp_{i+1}-Zp_i}\right) - \frac{1}{2}\eta\frac{\lambda_i F_{N_{MCL_i}}}{v_i\sin\gamma_i} \tag{6.48}$$

$$\implies \lambda_{i+1} = -\frac{2v_{i+1}\sin\gamma_{i+1}}{\eta F_{N_{MCL_{i+1}}}}\left(\frac{m_{i+1}-m_i}{Zp_{i+1}-Zp_i}\right) - \frac{\lambda_i F_{N_{MCL_i}} v_{i+1}\sin\gamma_{i+1}}{F_{N_{MCL_{i+1}}} v_i\sin\gamma_i} \tag{6.49}$$

Esta expresión define λ_{i+1} como una función que depende de (v_{i+1}, γ_{i+1}), de la aún desconocida m_{i+1}, y de las variables conocidas del estado $\mathbf{s}_i$ (específicamente $m_i, v_i, \gamma_i, \lambda_i$) y parámetros $(Zp_i, Zp_{i+1}, F_{N_{MCL_i}}, F_{N_{MCL_{i+1}}}, \eta)$. Formalmente:

$$\lambda_{i+1} = f_\lambda(v_{i+1}, \gamma_{i+1}, m_{i+1}; \mathbf{s}_i, \text{params}) \tag{6.50}$$

Paso 2: Expresión explícita para Cz_{i+1}

La ecuación de equilibrio de fuerzas perpendicular a la trayectoria, representada por $g_{\gamma_i} = 0$ (ec. (6.4)), relaciona el coeficiente de sustentación Cz_{i+1} con la curvatura de la trayectoria (cambio en γ) y otras variables. A partir de (6.4):

$$\frac{1}{2}\frac{\frac{1}{2}\rho_{i+1}S_{REF}Cz_{i+1}}{m_{i+1}\sin\gamma_{i+1}} = \frac{\gamma_{i+1}-\gamma_i}{Zp_{i+1}-Zp_i} - \frac{1}{2}\left(\frac{\frac{1}{2}\rho_i S_{REF}Cz_i}{m_i\sin\gamma_i} - \frac{g_0}{v_i^2\tan\gamma_i} - \frac{g_0}{v_{i+1}^2\tan\gamma_{i+1}}\right) \tag{6.51}$$

$$\Longrightarrow$$

$$Cz_{i+1} = \frac{4m_{i+1}\sin\gamma_{i+1}}{\rho_{i+1}S_{REF}}\left[\frac{\gamma_{i+1}-\gamma_i}{Zp_{i+1}-Zp_i} - \frac{1}{2}\left(\frac{\rho_i S_{REF} Cz_i}{2m_i\sin\gamma_i} - \frac{g_0}{v_i^2\tan\gamma_i} - \frac{g_0}{v_{i+1}^2\tan\gamma_{i+1}}\right)\right] \tag{6.52}$$

donde $\rho_i = \rho(Zp_i)$ y $\rho_{i+1} = \rho(Zp_{i+1})$ (ec. (6.23)). Similarmente, Cz_{i+1} es una función de (v_{i+1}, γ_{i+1}), de la incógnita m_{i+1}, y del estado $\mathbf{s}_i$ (específicamente v_i, γ_i, m_i, Cz_i) y parámetros $(Zp_i, Zp_{i+1}, \rho_i, \rho_{i+1}, S_{REF}, g_0)$:

$$Cz_{i+1} = f_{Cz}(v_{i+1}, \gamma_{i+1}, m_{i+1}; \mathbf{s}_i, \text{params}) \tag{6.53}$$

Paso 3: Solución Analítica para m_{i+1}

Contrario a una dependencia implícita general, una inspección detallada de la estructura algebraica revela que la sustitución de λ_{i+1} (ec. (6.49)) y Cz_{i+1} (ec. (6.52)) en la ecuación $g_{v_i} = 0$ (ec. (6.3)) conduce a una ecuación cuadrática en la variable m_{i+1}. Para demostrar esto, reescribimos $g_{v_i} = 0$ aislando los términos que dependen explícitamente de m_{i+1} en el lado derecho:

$$\underbrace{2\left(\frac{v_{i+1}-v_i}{Zp_{i+1}-Zp_i}\right) + \frac{g_0}{v_{i+1}} - \left(\frac{\lambda_i F_{N_{MCL_i}}}{m_i v_i\sin\gamma_i} - \frac{\rho_i v_i S_{REF}(Cx_0 + kCz_i^2)}{2m_i\sin\gamma_i} - \frac{g_0}{v_i}\right)}_{\text{Termino } K_1 \text{ (depende solo de } i, v_{i+1}, \gamma_{i+1})} =$$

$$= \frac{\lambda_{i+1}F_{N_{MCL_{i+1}}}}{m_{i+1}v_{i+1}\sin\gamma_{i+1}} - \frac{\rho_{i+1}v_{i+1}S_{REF}(Cx_0 + kCz_{i+1}^2)}{2m_{i+1}\sin\gamma_{i+1}} \tag{6.54}$$

Multiplicamos ambos lados por $2m_{i+1}v_{i+1}\sin\gamma_{i+1}$:

$$K_1 \cdot (2m_{i+1}v_{i+1}\sin\gamma_{i+1}) = 2\lambda_{i+1}F_{N_{MCL_{i+1}}} - \rho_{i+1}v_{i+1}^2 S_{REF}(Cx_0 + kCz_{i+1}^2) \tag{6.55}$$

Ahora, introducimos las dependencias lineales de λ_{i+1} y Cz_{i+1} respecto a m_{i+1}. Definimos coeficientes que dependen únicamente de variables conocidas en i y las independientes (v_{i+1}, γ_{i+1}):

$$I' = -\frac{2v_{i+1}\sin\gamma_{i+1}}{\eta F_{N_{MCL_{i+1}}}(Zp_{i+1} - Zp_i)} \tag{6.56}$$

$$J' = \frac{2v_{i+1}\sin\gamma_{i+1}m_i}{\eta F_{N_{MCL_{i+1}}}(Zp_{i+1} - Zp_i)} - \frac{\lambda_i F_{N_{MCL_i}} v_{i+1}\sin\gamma_{i+1}}{F_{N_{MCL_{i+1}}} v_i \sin\gamma_i} \tag{6.57}$$

$$H' = \frac{2\sin\gamma_{i+1}}{\rho_{i+1}S_{REF}}\left[\frac{2(\gamma_{i+1}-\gamma_i)}{Zp_{i+1}-Zp_i} - \left(\frac{\rho_i S_{REF} C z_i}{2m_i\sin\gamma_i} - \frac{g_0}{v_i^2\tan\gamma_i} - \frac{g_0}{v_{i+1}^2\tan\gamma_{i+1}}\right)\right] \tag{6.58}$$

Con estos coeficientes, las ecuaciones (6.49) y (6.52) se escriben como:

$$\lambda_{i+1} = I'm_{i+1} + J' \tag{6.59}$$

$$Cz_{i+1} = H'm_{i+1} \tag{6.60}$$

Sustituyendo (6.59) y (6.60) en (6.55):

$$\begin{aligned}
&(2K_1 v_{i+1}\sin\gamma_{i+1})m_{i+1} = \\
&= 2F_{N_{MCL_{i+1}}}(I'm_{i+1} + J') - \rho_{i+1}v_{i+1}^2 S_{REF}(Cx_0 + k(H'm_{i+1})^2) \\
&= (2F_{N_{MCL_{i+1}}}I')m_{i+1} + (2F_{N_{MCL_{i+1}}}J') - (\rho_{i+1}v_{i+1}^2 S_{REF}Cx_0) - (\rho_{i+1}v_{i+1}^2 S_{REF}kH'^2)m_{i+1}^2
\end{aligned} \tag{6.61}$$

Reorganizando todos los términos para formar una ecuación cuadrática estándar $ax^2 + bx + c = 0$ con $x = m_{i+1}$:

$$\begin{aligned}
\underbrace{(\rho_{i+1}v_{i+1}^2 S_{REF}kH'^2)}_{a} m_{i+1}^2 + \underbrace{(2K_1 v_{i+1}\sin\gamma_{i+1} - 2F_{N_{MCL_{i+1}}}I')}_{b} m_{i+1} + \\
+ \underbrace{(\rho_{i+1}v_{i+1}^2 S_{REF}Cx_0 - 2F_{N_{MCL_{i+1}}}J')}_{c} = 0
\end{aligned} \tag{6.62}$$

Los coeficientes a, b, c dependen únicamente de las variables en el estado i y de las variables independientes (v_{i+1}, γ_{i+1}), a través de K_1, I', J', H'.

La solución para m_{i+1} se obtiene directamente aplicando la fórmula cuadrática:

$$m_{i+1} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{6.63}$$

De las dos posibles soluciones (correspondientes a los signos $\pm$), se selecciona la físicamente relevante. Dado que se espera que la masa m_{i+1} sea positiva y cercana a m_i, generalmente solo una de las raíces cumple estas condiciones. En la implementación del código, se elige la raíz correspondiente al signo positivo (+) en el numerador, que corresponde a *numerator_2* en el código del apéndice B. Definimos la función f_m que encapsula esta solución analítica:

$$m_{i+1} = f_m(v_{i+1}, \gamma_{i+1}; \mathbf{s}_i, \text{params}) \tag{6.64}$$

Es fundamental notar que las variables intermedias A, L, H, I, J utilizadas en el código Python corresponden a agrupaciones algebraicas de los términos que aparecen en los coeficientes a, b, c de la ecuación cuadrática (6.62), y la fórmula implementada ‘(numerator_1 + numerator_2)/denominator‘ es algebraicamente equivalente a la solución de la fórmula cuadrática (6.63) con la elección del signo positivo.
Este resultado analítico elimina la necesidad de un solver numérico para m_{i+1} en cada paso, haciendo la evaluación de la trayectoria computacionalmente más eficiente y robusta, ya que evita posibles problemas de convergencia asociados a los métodos iterativos de búsqueda de raíces.

Paso 4: Cálculo de t_{i+1} y s_{i+1}
Una vez obtenido el valor analítico de $m^*_{i+1} = f_m(v_{i+1}, \gamma_{i+1}; \mathbf{s}_i, \text{params})$ usando (6.63), se calculan los valores correspondientes de λ^*_{i+1} y Cz^*_{i+1} usando las ecuaciones lineales (6.59) y (6.60) (evaluando I', J', H' con los valores conocidos). Finalmente, las variables de tiempo t_{i+1} y distancia s_{i+1} se obtienen directamente despejando de las ecuaciones (6.6) y (6.7), obteniendo así las siguientes expresiones:

$$t^*_{i+1} = t_i + \frac{1}{2}(Zp_{i+1} - Zp_i)\left(\frac{1}{v_{i+1}\sin\gamma_{i+1}} + \frac{1}{v_i \sin\gamma_i}\right) \tag{6.65}$$

$$s^*_{i+1} = s_i + \frac{1}{2}(Zp_{i+1} - Zp_i)\left(\frac{1}{\tan\gamma_{i+1}} + \frac{1}{\tan\gamma_i}\right) \tag{6.66}$$

Procedimiento Iterativo Completo (Paso $i \to i+1$):

El cálculo completo del estado $\mathbf{s}_{i+1}$ a partir de $\mathbf{s}_i$ y la elección de (v_{i+1}, γ_{i+1}) sigue la secuencia analítica:

1. **Calcular coeficientes:** Evaluar K_1, I', J', H' (ec. (6.54), (6.56)-(6.58)).
2. **Calcular coeficientes cuadráticos:** Evaluar a, b, c de la ecuación (6.62).
3. **Resolver para m_{i+1}:** Calcular $m^*_{i+1} = \frac{-b+\sqrt{b^2-4ac}}{2a}$ (ec. (6.63)).
4. **Calcular λ_{i+1}:** Evaluar $\lambda^*_{i+1} = I' m^*_{i+1} + J'$ (ec. (6.59)).
5. **Calcular Cz_{i+1}:** Evaluar $Cz^*_{i+1} = H' m^*_{i+1}$ (ec. (6.60)).
6. **Calcular t_{i+1}:** Evaluar $t^*_{i+1} = f_t(\dots)$ (ec. (6.65)).
7. **Calcular s_{i+1}:** Evaluar $s^*_{i+1} = f_s(\dots)$ (ec. (6.66)).

El estado resultante es $\mathbf{s}_{i+1} = (v_{i+1}, \gamma_{i+1}, m^*_{i+1}, t^*_{i+1}, s^*_{i+1}, Cz^*_{i+1}, \lambda^*_{i+1})$.

La función de costo reducida y el problema de optimización:

Este procedimiento iterativo analítico define una función de mapeo, $\mathcal{M}$, que toma como entrada el vector de las 104 variables independientes, $\mathbf{x}_{\text{indep}} = (v_1, \dots, v_{N-1}, \gamma_1, \dots, \gamma_{N-1}) \in \mathbb{R}^{104}$, y junto con el estado inicial $\mathbf{s}_0$, genera la secuencia completa de estados $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_{N-1})$:

$$\mathbf{S} = \mathcal{M}(\mathbf{x}_{\text{indep}}; \mathbf{s}_0) \tag{6.67}$$

La función de costo original, ϕ_{orig} (ec. (6.41)), depende explícitamente de las componentes del estado final del ascenso, $\mathbf{s}_{N-1} = (v_{N-1}, \gamma_{N-1}, m_{N-1}, t_{N-1}, s_{N-1}, Cz_{N-1}, \lambda_{N-1})$, que son parte de la salida $\mathbf{S}$ del mapeo $\mathcal{M}$. Por lo tanto, la función de costo puede ser expresada como una composición, resultando en una función que depende únicamente del vector de variables independientes $\mathbf{x}_{\text{indep}}$:

$$\boxed{\phi(\mathbf{x}_{\text{indep}}) = \phi_{\text{orig}}(\mathcal{M}(\mathbf{x}_{\text{indep}}; \mathbf{s}_0))} \tag{6.68}$$

Formalmente, $\phi : \mathbb{R}^{104} \to \mathbb{R}$. El problema de optimización se transforma en:

$$\min_{\mathbf{x}_{\text{indep}} \in \mathcal{D}} \phi(\mathbf{x}_{\text{indep}}) \tag{6.69}$$

donde $\mathcal{D} \subset \mathbb{R}^{104}$ es el dominio factible. Este dominio está definido implícitamente por las restricciones de desigualdad (6.8) a (6.12), las cuales deben ser verificadas en cada paso $j = 1, \ldots, N-1$ utilizando los valores de las variables dependientes $(m_j, t_j, s_j, Cz_j, \lambda_j)$ calculadas por el mapeo $\mathcal{M}$. Es decir, un vector $\mathbf{x}_{\text{indep}}$ es factible si y solo si la trayectoria $\mathbf{S} = \mathcal{M}(\mathbf{x}_{\text{indep}}; \mathbf{s}_0)$ satisface todas las restricciones de desigualdad en todos los puntos $j = 1, \ldots, N-1$.

6.4. Muestreo del espacio de existencia y ajuste de la función objetivo

Evaluación comparativa de estrategias de muestreo para el caso 10D

Una vez definido el problema de optimización en el espacio reducido de 10 dimensiones (10D), $\mathbf{x}_{\text{indep}}^{10D} = (v_1, \ldots, v_5, \gamma_1, \ldots, \gamma_5)$, correspondiente a $N = 6$ puntos de discretización (Sección 6.3), el paso siguiente es seleccionar un método eficiente para generar puntos válidos en este espacio. La validez de un punto $\mathbf{x}_{\text{indep}}^{10D}$ requiere que la trayectoria resultante, calculada mediante el procedimiento iterativo analítico, satisfaga todas las restricciones operacionales y físicas (6.8) a (6.12) en cada uno de los puntos intermedios $i = 1, \ldots, 5$. El objetivo es obtener un conjunto representativo de pares $(\mathbf{x}_{\text{indep}}^{10D}, \phi)$ para el posterior ajuste del modelo QUBO (Sección 6.4).

Dada la complejidad inherente a la verificación de restricciones en cada paso de la trayectoria, la eficiencia del método de muestreo es crucial. Se consideraron diversas estrategias, incluyendo las discutidas la sección 6.4: métodos basados en grillas (impracticables en 10D), perturbaciones gaussianas (eficientes para exploración local pero limitados para cobertura global), búsqueda de puntos adyacentes (con escalabilidad exponencial 3^{10} que lo hace inviable computacionalmente para obtener una muestra amplia) y métodos basados en poblaciones como Evolución Diferencial (DE). Adicionalmente, se evaluaron métodos de muestreo aleatorio simple como Direcciones Aleatorias (RDS) y secuencias de baja discrepancia como Sobol Secuencial (SSS).

Para fundamentar la elección del método de muestreo, se realizó un experimento numérico

comparativo en el espacio 10D. Se ejecutaron cinco algoritmos — Búsqueda Adyacente, Evolución Diferencial (DE), Perturbación Local (LPS), Direcciones Aleatorias (RDS) y Sobol Secuencial (SSS) — durante un tiempo de cómputo predefinido (900s o 3600s, según el método) en el mismo entorno computacional. Se registraron las siguientes métricas clave para cada método:

- **Número de Puntos Válidos:** La cantidad total de puntos $\mathbf{x}_{\text{indep}}^{10D}$ que generaron trayectorias completas y válidas.
- **Eficiencia de Muestreo (Puntos/Segundo):** Tasa de generación de puntos válidos.
- **Cobertura Espacial (Hipervolumen BBox):** Volumen del mínimo hiperrectángulo alineado a los ejes que contiene todos los puntos válidos generados ($V_{BBox,m}$).
- **Cobertura Relativa Normalizada:** Dado que SSS está diseñado para una exploración uniforme, su volumen BBox $V_{BBox,SSS}$ se utilizó como referencia (100 %). La cobertura relativa de otro método m se calculó como $C_{rel,m} = (V_{BBox,m}/V_{BBox,SSS}) \times 100\,\%$.

Los resultados de esta evaluación comparativa se resumen visualmente en las Figuras 6.1 y 6.2. La Figura 6.1 muestra una comparación directa del número de puntos válidos, la eficiencia y, crucialmente, la cobertura relativa alcanzada. La Figura 6.2 ofrece una perspectiva normalizada del rendimiento multidimensional.

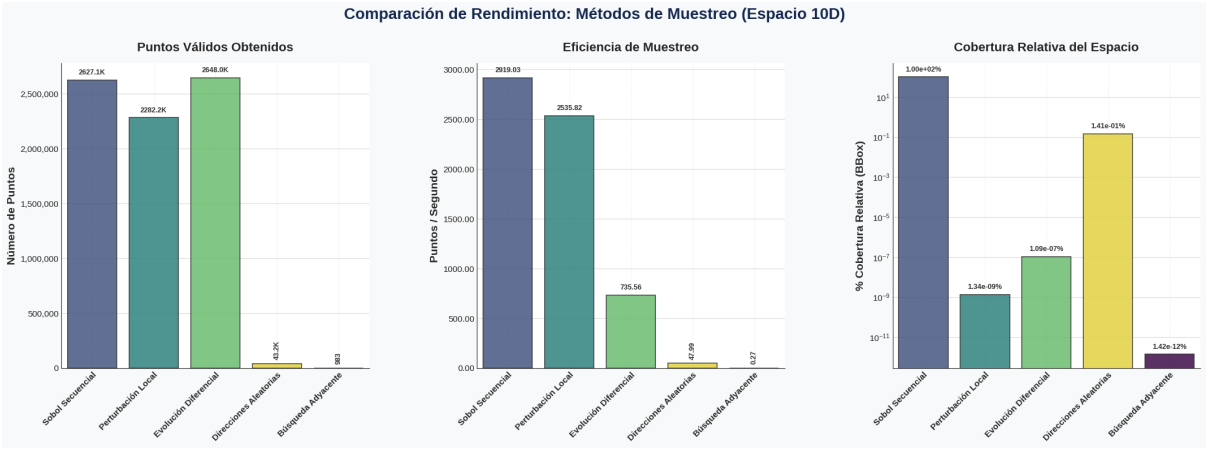

Figura 6.1: Comparación del rendimiento de cinco métodos de muestreo en el espacio 10D. Se muestra el número total de puntos válidos obtenidos, la eficiencia (puntos válidos por segundo) y el porcentaje de cobertura espacial del Bounding Box (BBox) relativo al alcanzado por Sobol Secuencial (SSS), dentro de tiempos de ejecución fijos (900s para LPS, RDS, SSS; 3600s para Búsqueda Adyacente y DE).

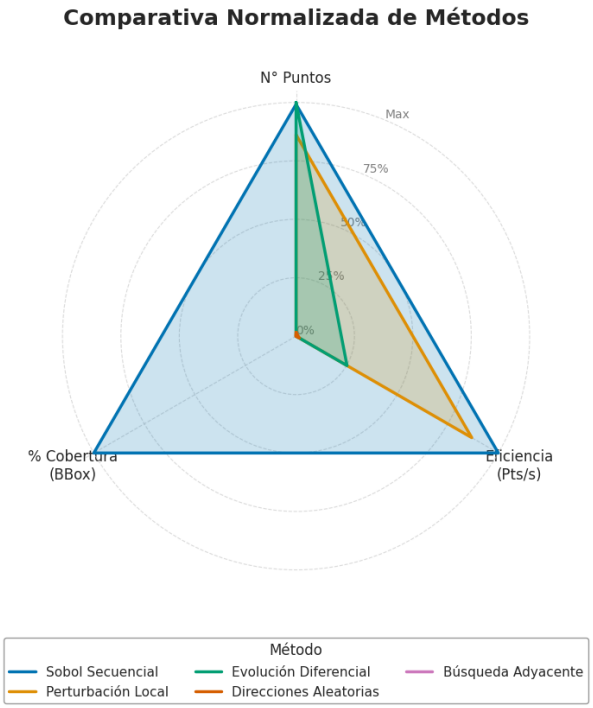

Figura 6.2: Gráfico de radar que ilustra el rendimiento normalizado (0 % al 100 % del máximo observado en la métrica) de los métodos de muestreo evaluados según el número de puntos, la eficiencia (puntos/segundo) y la cobertura relativa del BBox.

El análisis de los resultados revela diferencias significativas:

- **Generación de Puntos y Eficiencia:** SSS y LPS generaron la mayor cantidad de puntos válidos en sus 900 segundos de ejecución, demostrando una alta eficiencia en la producción de puntos. DE también generó un número elevado de puntos, pero requirió un tiempo considerablemente mayor (3600s), resultando en una eficiencia menor. RDS y, especialmente, Búsqueda Adyacente, fueron significativamente menos productivos.

- **Cobertura Espacial:** Esta métrica muestra la disparidad más crítica. SSS, por su naturaleza de baja discrepancia, logró explorar un hipervolumen BBox considerablemente mayor ($V_{BBox,SSS} \approx 9{,}57 \times 10^{10}$ unidades^{10}) que todos los demás métodos. La cobertura relativa (Figura 6.1, gráfico derecho) evidencia que RDS cubrió aproximadamente un 0.14 % del volumen BBox de SSS, mientras que los volúmenes cubiertos por DE, LPS y Búsqueda Adyacente fueron prácticamente insignificantes en comparación (órdenes de magnitud 10^{-7} % o menores). Estos resultados sugieren que, aunque métodos como LPS pueden generar puntos rápidamente, tienden a hacerlo en una región más localizada del espacio factible, mientras que SSS realiza

una exploración más amplia y sistemática.

- **Rendimiento Global:** El gráfico de radar (Figura 6.2) confirma el perfil equilibrado de SSS, destacando en cobertura mientras mantiene una alta generación de puntos. Los otros métodos muestran fortalezas en algunas métricas (e.g., eficiencia de LPS) pero debilidades marcadas en otras (principalmente cobertura).

Los datos numéricos completos que respaldan estas observaciones se encuentran detallados en la tabla del Apéndice [Referencia a tu Apéndice, ej. Apéndice D].
Basado en estos resultados experimentales, particularmente en la demostrada superioridad en la cobertura del espacio de búsqueda —una característica esencial para construir una aproximación global precisa de la función de costo ϕ—, se seleccionó el método de **Sobol Secuencial (SSS)** como la estrategia de muestreo principal para generar el conjunto de datos en el problema 10D. Aunque otros métodos puedan encontrar puntos válidos más rápidamente en fases iniciales, la capacidad de SSS para distribuir los puntos de manera más uniforme a través del dominio de las variables independientes $\mathbf{x}_{\text{indep}}^{10D}$ es fundamental para la calidad del ajuste posterior del modelo QUBO.

Ajuste de la función para la optimización

Después de discriminar en favor del método de muestreo de cuasi aleatorio Sobol, se hace una toma de cientos de miles de puntos, validados en la función objetivo en 10 dimensiones. Después del exhaustivo muestreo paso seguido se hace múltiples intentos de ajuste a una función que pueda ser optimizada mediante computación cuántica, se hicieron múltiples intentos usando polinomios de grado 2 o superior, no se consiguió un buen ajuste mostrando inconsistencias y los tiempos proyectados para ajustes adecuados no eran razonables.

Se intentó además un camino que para 2 y 4 dimensiones funcionó muy bien, por su consistencia, suavidad de la curva resultante, diferenciabilidad y facilidad para convertir en forma binaria, el juste a polinomios de Chebychev que se tenía como primera opción resultó ser computacionalmente intratable para dimensiones iguales o mayores a 8, los tiempos y el ajuste no eran óptimos al igual que el trabajo que dejaban para una conversión a binario era intratable.

Además, se ajustó el modelo con los datos tomados de forma estratificada, es decir que distintos rangos del valor final de la función f para convertir de forma directa a un problema en variables discretas, en este punto se estaba probando caminos alternativos que nos permitan trabajar en la optimización. Para ello se emplea KBinsDiscretizer de Scikit-learn. Las primeras cinco variables de entrada (x1-x5) son discretizadas en 17 bits, y las siguientes cinco (x6-x10) en 15. En este camino hubo mejor respuesta por parte de la función discreta binaria sin embargo aun no era optima, y los múltiples intentos con diferentes parámetros o codificaciones no daban los resultados buscados.

Finalmente y después de múltiples búsquedas tanto en la teoría como manipulación de múltiples parámetros en código, se usó el método de regresión ponderada, de esta forma es posible manipular y/o entrenar en el ajuste dando mayor carga a ciertos grupos de puntos tomados del muestreo, en este método se implementa además un termino en la suma de los cuadrados de los errores y es el termino L2 (Ridge) el cual es un término que advierte de sobre ajuste buscando una participación homogénea en las variables, el termino es la penalización $\lambda * sum[(coefs^2)]$ a la función de coste. Por otra parte, se usó la optimización a estos errores cuadrados el método de optimización convexa cvxpy, la cual es una una optimización mas flexible [39] que el método sklearn usado en intentos anteriores y entre otros muchos. El método resultó eficiente para el propósito de este trabajo y definió satisfactoriamente la función original.

Descripción del proceso de ajuste:

El paso previo fue la toma de datos conforme se describió en la subsección anterior.

El primer paso consistió entonces en el manejo de las variables y su conversión a la forma deseada:

- **Ponderación:** Para que el modelo priorice el ajuste en los valores más bajos, se asigna un peso w_i a cada punto i del conjunto de datos. Los puntos pertenecientes al α-percentil más bajo reciben un peso significativamente mayor (W_{alto}), mientras que el resto recibe un peso de 1.

$$w_i = \begin{cases} W_{\text{alto}} & \text{si } y_i \leq P_\alpha(y) \\ 1 & \text{si } y_i > P_\alpha(y) \end{cases} \tag{6.70}$$

Donde $P_\alpha(y)$ es el valor del percentil α del conjunto y.

- **Escalado:** El valor de la funcion objetivo $= y$ se mapea a un rango de $[0, 1]$, generando un nuevo vector y'.

$$y'_i = \frac{y_i - \text{mín}(y)}{\text{máx}(y) - \text{mín}(y)} \tag{6.71}$$

Paso 2: Discretización y binarización de variables de entrada

Las D variables de entrada continuas, $X = \{x_1, x_2, \ldots, x_D\}$, se transforman en variables binarias. Para cada variable continua x_j:

1. Se determina su rango empírico $[\text{mín}(x_j), \text{máx}(x_j)]$.
2. Este rango se divide en K_j intervalos o partes en las que el continuo es discretizado.
3. Se aplica la codificación transformando cada x_j en un vector de K_j variables binarias $q_j = \{q_{j,1}, q_{j,2}, \ldots, q_{j,K_j}\}$, donde cada variable puede tomar valores de 0 o 1.

Paso 3: Expansión a un espacio de características cuadráticas

A partir del conjunto total de variables binarias se construye una matriz de diseño Φ (no confundir con el ϕ que es el objetivo general de este trabajo) que contiene las características para el modelo de regresión cuadrático. Cada fila Φ_i de esta matriz, correspondiente a un punto de datos i, contiene:

- Un término de intercepción (bias): 1.
- Términos lineales: Todas las variables binarias $q_{j,k}$.
- Términos de interacción cuadrática: Todos los productos posibles $q_{j,k} \cdot q_{l,m}$.

Paso 4: Formulación y solución del problema de optimización

Finalmente entonces el objetivo se centra en encontrar el vector de coeficientes θ que minimiza la función de coste. Esta función combina la Suma Ponderada de los Errores al Cuadrado (WSSE) con un término de regularización L2. El problema de optimización puede escribirse así:

$$\underset{\theta}{\text{minimizar}} \left(\sum_{i=1}^{N} w_i \left(y'_i - \Phi_i \cdot \theta \right)^2 + \lambda \sum_{j=1}^{M} \theta_j^2 \right) \tag{6.72}$$

Donde:

- N es el número de puntos de datos.
- M es el número total de características en Φ.
- Φ_i es la fila i de la matriz de diseño.
- θ es el vector de coeficientes a encontrar.
- λ (lambda) es el hiperparámetro que controla la fuerza de la regularización.

Este problema de optimización convexa se resuelve utilizando un solver numérico especializado (libreria `cvxpy`).

Pseudocódigo del algoritmo

Algorithm 2 Ajuste Ponderado y Regularizado

Función AjustarModelo($X, y, \alpha, W_{\text{alto}}, \lambda$)
Entrada: Matriz de datos X, vector objetivo y, percentil α, peso W_{alto}, regularizador λ.
Salida: Vector de coeficientes θ, parámetros de escalado.
$P_\alpha \leftarrow$ CalcularPercentil(y, α)
$w \leftarrow$ CrearVectorPesos($y, P_\alpha, W_{\text{alto}}$) ▷ Paso 1
$y_{\text{mín}}, y_{\text{rango}} \leftarrow \text{mín}(y), \text{máx}(y) - \text{mín}(y)$
$y' \leftarrow (y - y_{\text{mín}})/y_{\text{rango}}$ ▷ Paso 1
$X_{\text{bin}} \leftarrow$ DiscretizarYEncode(X) ▷ Paso 2
$\Phi \leftarrow$ CrearFeaturesCuadraticas(X_{bin}) ▷ Paso 3
$\theta \leftarrow$ ResolverOptimizaciónConvexa(Φ, y', w, λ) ▷ Paso 4, usando la Ecuación 6.72
retornar $\theta, y_{\text{mín}}, y_{\text{rango}}$

Así entonces la funcion general a minimizar en este trabajo ϕ que se tiene originalmente de 104 variables para un N = 53, por disponibilidad computacional se tratará con N = 6 lo que significa 10 variables, 10D puede escribirse así $\phi = f(x_1, x_2, \ldots, x_{10})$, una función de variables continuas. Entonces, con los pasos aqui detallados dicha función ahora se escribe como una función cuadrática sobre un vector de variables binarias q. Esta forma se conoce como **QUBO (Quadratic Unconstrained Binary Optimization)** [23].
La expresión general en variables binarias pasa a ser la siguiente:

$$f(q) = \sum_i Q_{ii} q_i + \sum_{i<j} Q_{ij} q_i q_j \tag{6.73}$$

Donde:

- q es un vector largo que contiene todas las variables binarias, en este caso tiene una logitud de 160 $(q_0, q_1, \ldots, q_{159})$.
- Cada q_i al ser variables binarias solo pueden tomar los valores 0 o 1.

La función es una suma de dos tipos de términos:
Términos Lineales $\sum_i Q_{ii} q_i$:

- Este es el término $Q_{00}q_0 + Q_{11}q_1 + Q_{22}q_2 + \ldots$.
- Q_{ii} es el coeficiente que representa el 'coste' o 'energia' de que la variable binaria q_i sea igual a 1. Algo asi como un ponderado de cada valor un peso sobre cada qubit.

Términos Cuadráticos (o de 'Acoplamiento'): $\sum_{i<j} Q_{ij} q_i q_j$

- Este es el término $Q_{01}q_0q_1 + Q_{02}q_0q_2 + \cdots + Q_{158,159}q_{158}q_{159}$.
- Q_{ij} es el coeficiente que representa la fuerza de la interacción entre la variable q_i y la variable q_j. Es la fuerza de acoplamiento entre dos qubits.
- El término q_iq_j solo es distinto de cero (es 1) si y solo si las dos variables, q_i y q_j, son 1 al mismo tiempo.

La matriz Q La matriz creada a partir de los coeficientes Q_{ii} y Q_{ij} cuadrada y simetrica es llamada matriz Q

- Los elementos de la diagonal Q_{ii} son los coeficientes lineales.
- Los elementos que están fuera de la diagonal Q_{ij} son los coeficientes de las interacciones cuadráticas.

Al usar esta matriz, la función se puede escribir de forma más compacta en notación vectorial/matricial:

$$f(q) = q^{\mathsf{T}} Q q \tag{6.74}$$

$$\phi'(q) \equiv f(q) = q^{\mathsf{T}} Q q \tag{6.75}$$

Donde q es un vector columna y q^{T} es su transpuesto (un vector fila). En la ecuación (6.75) queda definida la funcion que se optimizara en la siguietne sección haciendo una comparativa con ϕ en 10D.

6.5. Optimización y análisis

El trabajo desarrollado hasta aquí culmina en la formulación de dos representaciones del problema de optimización, una función en variables continuas ϕ que se trabajará en esta sección en 10 dimensiones, y una función ϕ' que contiene la misma información pero que es discreta conforme se definió en (6.75), las funciones conservan la información del sistema pero sus naturalezas son distintas además se su dimensionalidad, la función ϕ' es una función en variables binarias q_i $(q_0, q_1, \ldots, q_{159})$ donde cada variable solo puede tomar uno de dos valores posibles 0 o 1. El valor $N = 6 - > 10\text{D}$ no es caprichoso, responde un punto de corte sobre la computación disponible al alcance de este trabajo. Sin embargo, la generalidad y en análisis construido se puede escalar directamente sin cambios metodológicos a dimensionalidades superiores.

Supuesto fundamental para la comparación

Para realizar una comparación fundamental entre los paradigmas de optimización es necesario que ambos lleven una representación idéntica del problema. El problema original de la trayectoria de ascenso es de naturaleza continua. Para su tratamiento mediante compu-

tación cuántica adiabática, se ha formulado un modelo de Optimización Cuadrática Binaria sin Restricciones (QUBO), representado por la matriz Q. En el análisis siguiente, se asume que este modelo QUBO no es una aproximación, sino una representación discreta, exacta y completa del problema de optimización original. Esta suposición metodológica permite aislar el rendimiento de los métodos de optimización en sí mismos, eliminando el error de modelado como variable que pudiera confundir. Por lo tanto, el optimizador clásico se enfrentará a la función continua $\phi(\mathbf{x})$, mientras que el optimizador cuántico abordará el equivalente discreto $\phi'(q)$.

Optimizador clásico

El primer pilar del experimento consiste en evaluar la estabilidad y fiabilidad del enfoque clásico. Se utilizará el algoritmo de Evolución Diferencial, algoritmo que ya antes usé y que es ampliamente utilizado para estos problemas de optimización [43]. Se realizaron aproximadamente 1000 experimentos con distintas variaciones de parámetros del código usando el método evolución diferencial, tales como el tamaño de la población, numero de iteraciones, estrategia de mutación entre otras, una vez elegida la mejor configuración posible después de una minuciosa comparativa de resultados se usó la configuración mostrada en la Tabla 6.1 sobre la función de coste continua de 10D, La hipótesis subyacente es que este método, si bien es rápido en una única ejecución, es propenso a quedar atrapado en óptimos locales, lo que producirá una distribución dispersa de soluciones finales. El resultado de este experimento será un histograma de frecuencias que visualizará la variabilidad de los mínimos encontrados, junto con un análisis estadístico de la dispersión (media, desviación estándar, mejor y peor resultado).

Tabla 6.1: Parámetros de Evolución Diferencial (*scipy.optimize*)

Parámetro	**Tipo**	**Descripción**	**Valor**
strategy	str	Estrategia de mutación	best1bin
maxiter	int	Máximo de iteraciones	100
popsize	int	Tamaño de la población	60
recombination	float	Probabilidad de cruce	0.7
tol	float	Tolerancia de convergencia	1e-4
x_0	ndarray	Punto inicial de referencia	[184.08, ..., 4.25]
bounds	list	Límites de variables	[(115,242.07)m/s,(0,π/2)]

Nota: La programación presentada en este trabajo fue hecha en Python, sin embargo para ejecutar las optimizaciones se tradujo la función, mediante librería especializada, a código de máquina específico para la arquitectura del hardware usado, esto para no tener las limitaciones de rendimiento del lenguaje interprete Python.

Optimizador cuántico

Se empleó un Recocido Cuántico Simulado que emula el comportamiento de un procesador cuántico adiabático, para resolver el problema formulado como QUBO. Este método opera directamente sobre la matriz Q de ϕ'.

Algorithm 3 Proceso de Optimización Cuántica

Prerrequisito: Un modelo polinómico (M_{poly}), un factor de penalización (P), y un **número de lecturas (N_{reads})**.

1. Formulación del Problema (Construcción del BQM)

$\text{BQM}_{objetivo} \leftarrow \text{Traducir}(M_{poly})$
$\text{BQM}_{penalizacion} \leftarrow \text{CrearConstraint}$
$\text{BQM}_{final} \leftarrow \text{BQM}_{objetivo} + P \cdot \text{BQM}_{penalizacion}$

2. Resolución Cuántica (Simulada)

solver $\leftarrow$ InicializarSampler(tipo="SimulatedAnnealing")
sampleset $\leftarrow$ solver.sample(BQM_{final}, num_reads $= N_{reads}$) ▷ Ejecutar N_{reads} veces.
mejor_solucion $\leftarrow$ sampleset.first() ▷ Extraer el estado de menor energía de todas las lecturas.

3. Interpretación del Resultado

energia_objetivo $\leftarrow$ $\text{BQM}_{objetivo}$.energy(mejor_solucion)
resultado_final $\leftarrow$ Desescalar(energia_objetivo)

return resultado_final

Nota: El proceso se implementó en Python utilizando el SDK "dimod" de D-Wave. El problema se formula como QUBO y el sampler lo resuelve encontrando el estado fundamental de su Hamiltoniano de Ising equivalente.

Comparativa y proyección a alta dimensionalidad

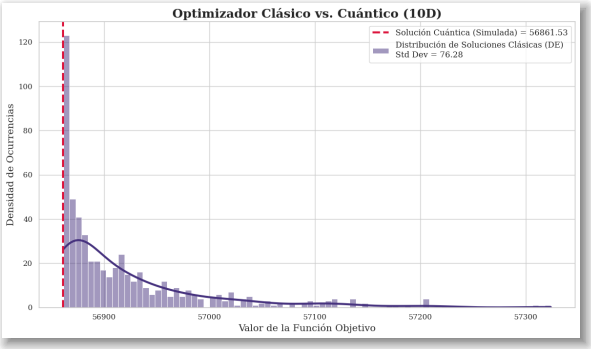

Figura 6.3: Comparativa de consistencia y precisión en 10D: la dispersión clásica frente a la consistencia del recocido cuántico en línea roja discontinua.

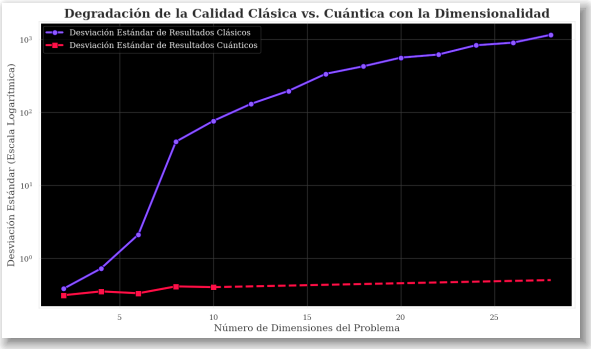

Figura 6.4: La inestabilidad del optimizador clásico crece exponencialmente al aumentar la dimensionalidad del problema. En rojo el resultado cuántico sobre el que siempre se debe tener presente es una dispersión de naturaleza distinta y siempre centrada en el óptimo global

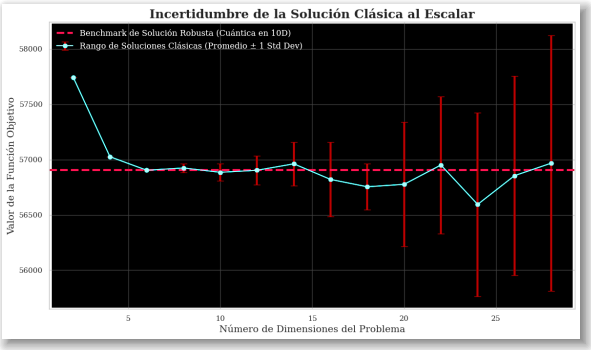

Figura 6.5: Este gráfico ilustra cómo la incertidumbre de la solución del optimizador clásico se magnifica con la dimensionalidad. Las barras de error, que representan la desviación estándar, crecen drásticamente, mostrando que el rango de posibles resultados se vuelve inmanejable. La línea discontinua representa el benchmark de una solución idealmente consistente.

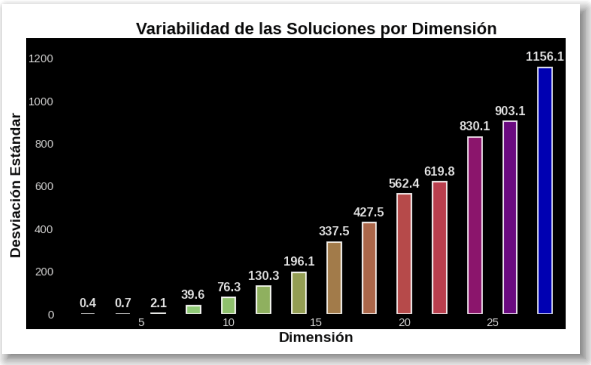

Figura 6.6: La desviación estándar del optimizador clásico vs dimensionalidad.

Se observa en 10D una amplia dispersión entre sus resultados, este método usado a pesar de ser el mas consistente y de mejor desempeño entre los métodos clásicos para este problema, presenta discrepancias entre los valores encontrados, haciendo un análisis del fundamento matemático que gobierna el método la explicación mas probable es su caída en mínimos locales (figura 4.3). El aumento en la dispersión a medida que sube la dimensionalidad nos da cuenta del aumento en la rugosidad de la hipersuperficie de

existencia de la función lo cual hace que la dispersión en las soluciones sea imposible de manejar a altas dimensionalidades. En la gráfica 6.3 se traza una única línea roja discontinua, la cual representa una única solución entre miles de experimentos cuánticos simulados, la simulación si bien nos lleva a un caso ideal permite hacer una comparación plena de los algoritmos clásico vs cuántico, en un trabajo posterior es posible explorar las dificultades en experimento real tales como el ruido, y como este puede interferir en un resultado tan óptimo como el aquí observado.

El resultado obtenido en computador cuántico ideal libera de las dificultades tecnológicas y aísla a una comparativa netamente centrada en el principio de funcionamiento, mostrando que la optimización de la energía global a partir del recocido cuántico resulta siempre entre cientos de experimentos un valor óptimo, y en milésimas de segundo, con una dispersión que aparte de ser de hasta 100 veces menor, por su principio de funcionamiento siempre esta al rededor del óptimo global no así en el caso clásico cuya dispersión puede ser al rededor de un óptimo local (figura 4.3). De forma análoga un computador cuántico podría representar menos de un segundo, de esta forma se ve claramente que el enfoque cuántico presenta clara superioridad en esta optimización.

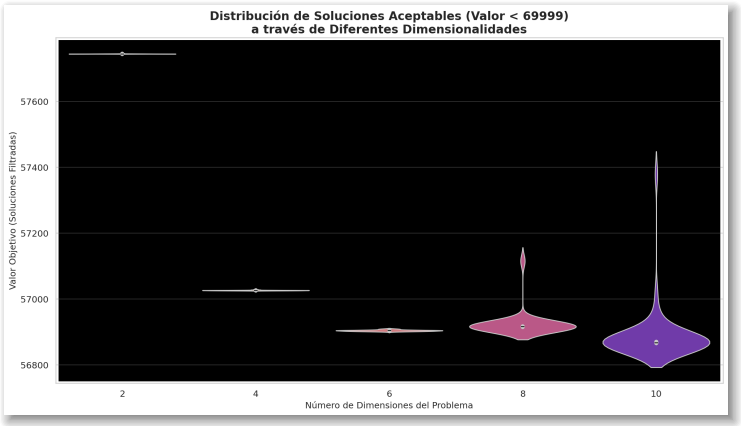

Figura 6.7: La inestabilidad del optimizador clásico crece exponencialmente al aumentar la dimensionalidad del problema.

El gráfico 6.4 sumado al gráfico 6.7 presenta el cambio de la desviación estándar de los resultados de la optimización clásica. A medida que la dimensionalidad crece hay un crecimiento exponencial en el su dispersión, lo que se puede leer como una disminución en la calidad del resultado obtenido, en el eje y la escala en la gráfica 6.4 es logarítmica, este desempeño del algoritmo clásico muestra la precariedad del método clásico en el problema de la dimensionalidad tratado y su empeoramiento en dimensionalidades superiores dada

la proyección mostrada en la figura 6.8 es un claro ejemplo de la maldición de la dimensionalidad donde el problema se vuelve intratable en dimensiones superiores. La existencia de cada vez más numerosos valles en la hipersuperficie hace que los optimizadores caigan en ellos haciendo cada vez más improbable el hallazgo de un óptimo absoluto, de ahí su ineficiencia en la búsqueda del mejor óptimo global. La confianza en la solución clásica disminuye en la medida que el problema representa de mejor forma el problema real.

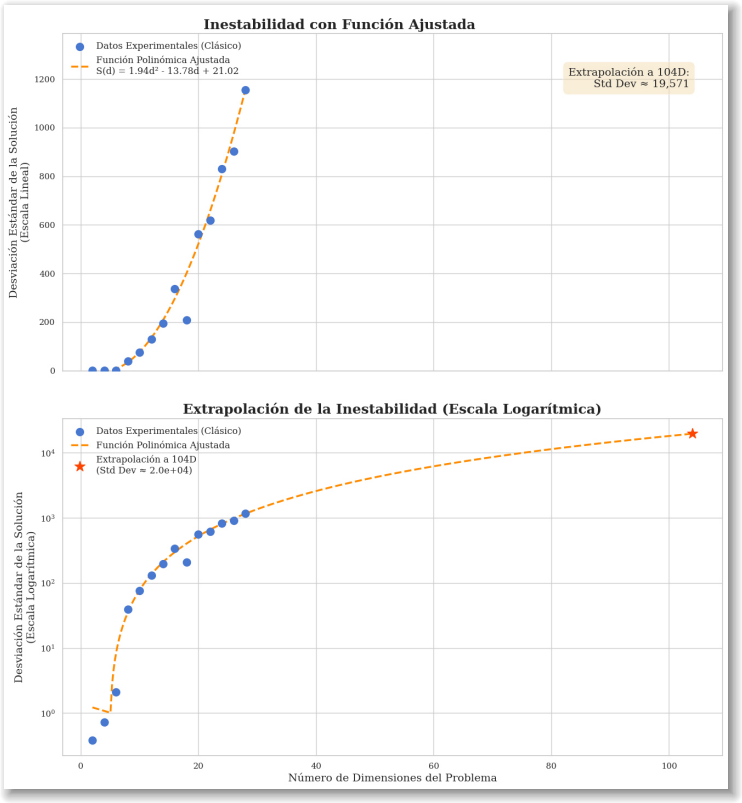

Figura 6.8: Inestabilidad del optimizador clásico crece exponencialmente al aumentar la dimensionalidad del problema, proyectado a 104D.

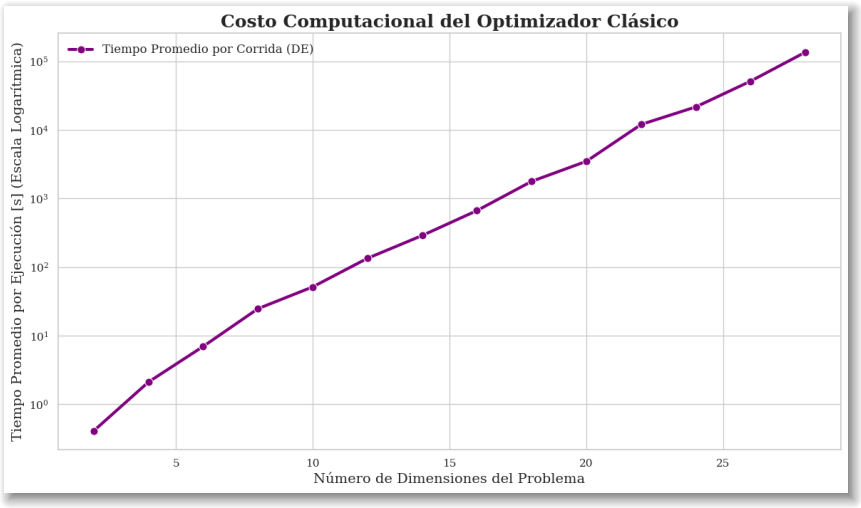

Figura 6.9: El gráfico muestra que el tiempo de ejecución clásico crece exponencialmente con la dimensionalidad, volviéndose rápidamente inviable usando este metodo.

Conforme se mostró existen ineficiencias relacionadas con la calidad del resultado dadas las condiciones particulares del problema, aquí se presenta un problema igualmente difícil de tratar y es el tiempo que tarda conforme la dimensionalidad aumenta, es una muestra que incluso si las soluciones fueran de alta calidad el solo tiempo sería una barrera difícil de esquivar. De nuevo la maldición de la dimensionalidad esta vez en la dimensión tiempo.

Para el caso 104D que es el total de dimensiones requerido por Airbus el total del tiempo usando, en una proyección exponencial seriá de $3{,}2\times10^{20}s$, un tiempo que supera cualquier consideración.

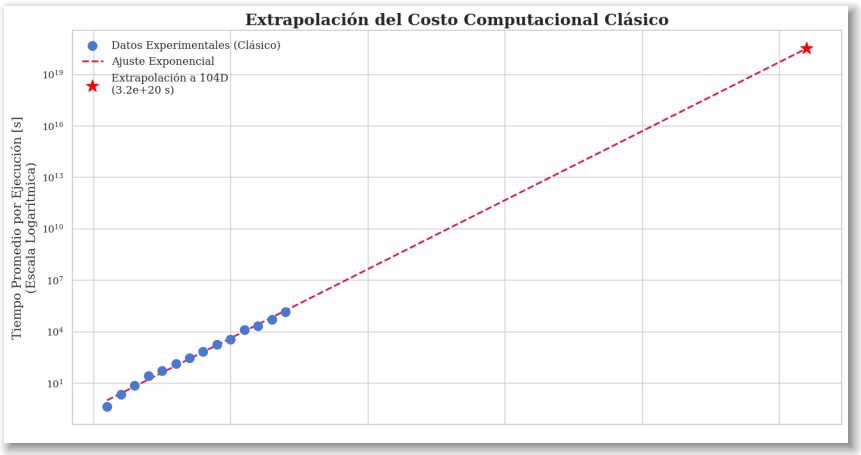

Figura 6.10: Este gráfico muestra la proyeccion del tiempo requerido para la dimensionalidad para N = 53, 104 dimensiones.

Para mapear un problema de 1664 variables lógicas, que es el número de qubits que se necesitan para esta optimización, los qubits físicos necesarios en un procesador de

annealing cuántico como el de D-Wave es mayor pues no es una correspondencia directa de uno a uno. El factor determinante es la diferencia fundamental entre la topología del problema lógico y la topología del hardware físico. En general un problema lógico definido por la matriz Q, podría tener interacciones entre cualquier par de qubits mientras que en una arquitectura física de un procesador D-Wave, como la topología Pegasus, cada qubit físico está conectado directamente a un número limitado de vecinos (hasta 15 en el caso de Pegasus) [54]. Para superar esta limitación de conectividad, se emplea el proceso "minor-embedding" [55]. En este proceso, un único qubit lógico que necesita más conexiones de las que un qubit físico puede ofrecer, se representa mediante una "cadena" de múltiples qubits físicos. Estos qubits físicos dentro de una cadena se acoplan ferromagnéticamente con una fuerza de acoplamiento de tal forma que los obliga a adoptar el mismo estado final (ya sea 0 o 1) al final del annealing, comportándose colectivamente como una sola variable lógica [24].

Para este problema el número de qubits físicos podría ser varias veces superior al número de qubits lógicos [56, 57]. Los 1664 qubits lógicos podrían exceder los 5,000 qubits físicos disponibles hoy en día [55], así aunque el número de variables lógicas (1664) es conceptualmente manejable, la demanda de conectividad del problema probablemente excedería la capacidad de embedding de los sistemas actuales. Sin embargo, el número de qubits físicos está en constante crecimiento y esta limitante puede ser prontamente superada.

En el proceso físico en el optimizador cuántico, el parámetro temporal crítico es el tiempo de recocido (t_a), que debe ser suficiente para permitir una evolución adiabática del sistema para garantizar una alta probabilidad de encontrar el estado fundamental. Los annealers operan con tiempos de recocido en el rango de 1 a 2,000 microsegundos, en una configuración de 20ν s, haciendo 1000 lecturas para fiabilidad considerable, se requerirá un tiempo de coherencia de aproximadamente $20ms$.

Optimización clásica en 104D

Aparte de la comparación metodológica sistemática realizada para la comparación de optimización en el espacio 10D, se realizó un extenso trabajo empleando tiempo y computación, esto permitía conocer problema y observar de primera mano el tipo de problemas que se presentaría o como se comportarían distintos optimizadores clásicos. Este proce-

so fue arduo y construido a partir de la intuición y la guía de los resultados y nuevos óptimos encontrados. Se intercambiaban los métodos y se modulaban sus configuraciones conforme evolucionaba.

El proceso fue híbrido y heurístico a lo largo de varios meses, el proceso combinó la capacidad de exploración global de algoritmos evolutivos con la eficiencia de refinamiento local de métodos basados en gradiente y los otros métodos también aquí mencionados. La metodología consistió en ciclos iterativos: primero, se utilizó la DE con diversas configuraciones de sus hiperparámetros para identificar regiones prometedoras del vasto espacio de soluciones; posteriormente, las mejores soluciones candidatas de cada ciclo servían como puntos de partida para optimizadores locales que afinaban la búsqueda en la vecindad de dichos puntos. Esta naturaleza adaptativa permitió navegar la compleja topografía de la función de coste, superando semana a semana óptimos locales que atrapaban a los algoritmos en ejecuciones aisladas.

Tras la búsqueda, se identificó un mínimo encontrado que postulo como candidato para ser óptimo global del problema general. El valor de la función de coste ϕ asociado a esta solución es de:

$$\boxed{\phi = -57745{,}39\, kg/s}$$

Las 104 variables independientes, $\{v_i, \gamma_i\}_{i=1}^{52}$, que definen esta trayectoria se detallan en la siguiente tabla.

Tabla 6.2: Variables independientes de la trayectoria óptima para el problema de 104D.

Índice i	Velocidad v_i ($\mathrm{m\,s^{-1}}$)	Ángulo de Ascenso γ_i (rad)	Índice i	Velocidad v_i ($\mathrm{m\,s^{-1}}$)	Ángulo de Ascenso γ_i (rad)
1	135,8347	0,642 845	2	138,2430	0,631 645
3	152,2185	0,818 820	4	136,9969	2,084 779
5	142,1637	0,614 226	6	157,1104	2,315 285
7	164,1562	0,531 933	8	174,6586	2,410 145
9	186,6152	1,443 332	10	199,3308	2,203 392
11	215,3360	1,308 845	12	221,4434	2,159 385
13	211,4662	1,081 694	14	182,3680	0,904 520
15	187,2567	0,477 581	16	212,3132	1,175 155
17	206,8116	1,143 725	18	207,8341	1,929 058
19	207,4008	2,352 793	20	202,1019	2,806 671
21	197,2976	2,494 457	22	194,7604	2,241 350
23	199,5212	1,849 247	24	204,2877	1,792 373
25	207,1849	1,717 232	26	208,1099	2,371 659
27	209,2211	2,374 771	28	210,9815	1,996 427
29	205,9138	1,546 862	30	202,1064	1,621 573
31	206,0758	1,989 921	32	207,2431	1,998 724
33	207,0302	2,243 599	34	207,4182	1,865 333
35	207,2824	2,342 086	36	205,9318	1,787 688
37	205,4989	1,786 697	38	207,2067	1,449 652
39	208,1546	2,010 177	40	207,4875	1,615 847
41	204,0129	1,597 631	42	202,1321	1,182 128
43	207,8688	0,845 505	44	209,7671	1,310 557

– Continúa en la siguiente página –

– Continuación de la Tabla 6.2–

Índice i	Velocidad v_i ($\mathrm{m\,s^{-1}}$)	Ángulo de Ascenso γ_i (rad)	Índice i	Velocidad v_i ($\mathrm{m\,s^{-1}}$)	Ángulo de Ascenso γ_i (rad)
45	205,3224	1,646 342	46	206,1329	0,724 118
47	209,4838	1,096 549	48	209,3489	1,084 294
49	207,9902	1,199 175	50	207,4503	0,711 623
51	207,4373	0,811 255	52	207,0235	0,531 520

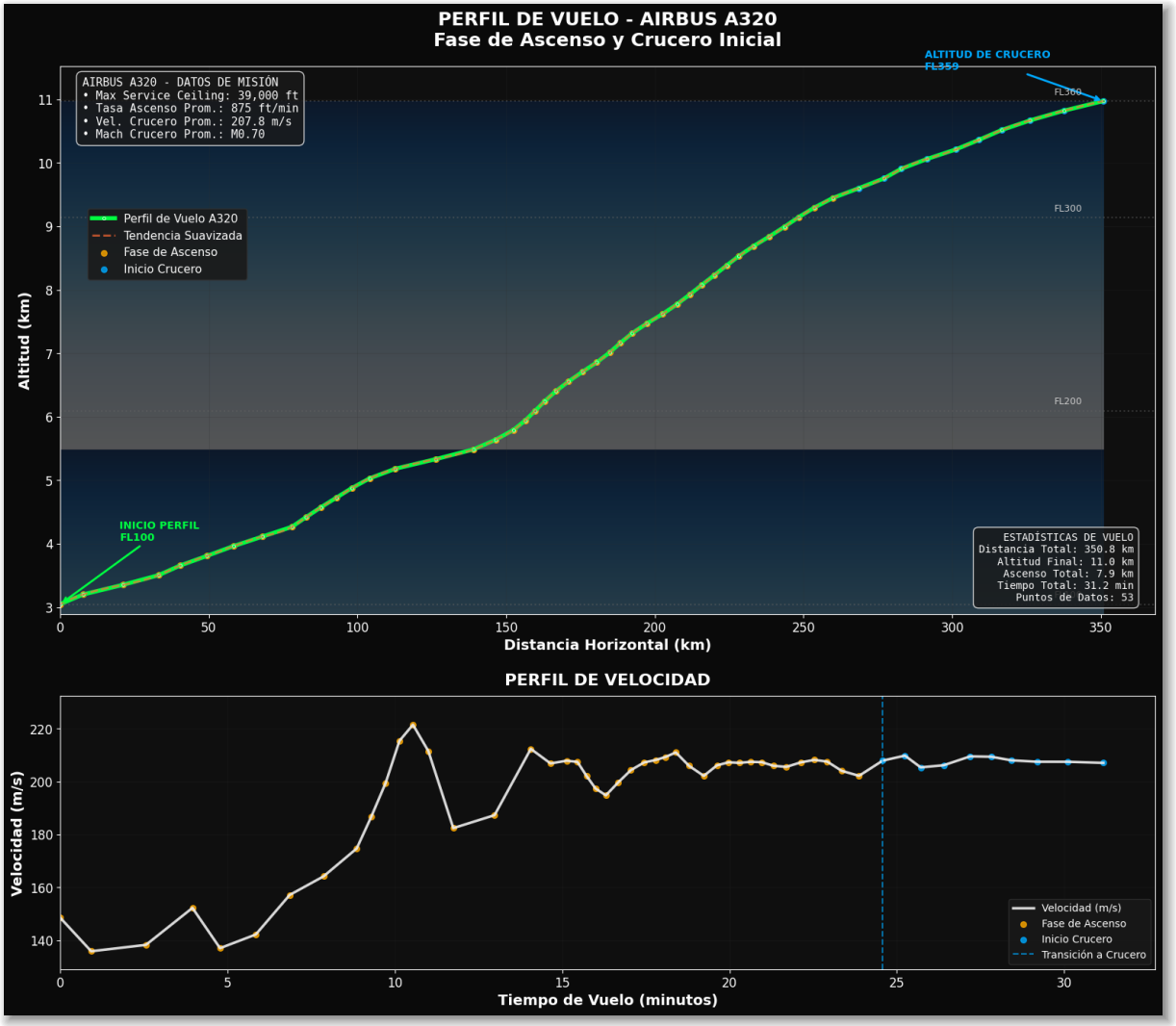

Figura 6.11: Perfil de vuelo del Airbus A320 trazado con los parámetros que minimizan el gasto en términos de combustible

Bibliografía

[1] Airbus. AIRBUS Quantum Computing Challenge. `https://www.airbus.com/en/innovation/cutting-edge/quantum-computing-challenge`, 2019. Consultado en 2023.

[2] Régis Alligier, David Gianazza, and Nicolas Durand. Robust aircraft sequence planning under uncertainties. *European Journal of Operational Research*, 242(2):663–672, 2015.

[3] Y. Lyu and J. M. De-la cruz. Flight trajectory optimization for single-aisle aircraft: A case study of the a320. *Journal of Air Transport Management*, 77:21–31, 2019.

[4] A. Ros-Suarez, M. Liz-Vargas, and J. Gueredíaga. An analysis of the fuel consumption in the climb phase of a flight. *The Aeronautical Journal*, 120(1229):1113–1128, 2016.

[5] D. P. Green. Fuel-burn reduction in the ascent and descent phases of flight. *Journal of Aircraft*, 48(3):745–755, 2011.

[6] R. Franke and B. Sridhar. Aircraft trajectory optimization for reduced environmental impact. *Aerospace Science and Technology*, 54:187–198, 2016.

[7] W. E. Hart and D. G. Luenberger. *Optimal control*. John Wiley & Sons, 2008.

[8] A. Murrieta-Trocoli, M. S-Cuesta, and D. Gonzalez-Arribas. A review of flight trajectory optimization methods. *Progress in Aerospace Sciences*, 97:1–22, 2018.

[9] D. Sasaki, Y. Okai, and Y. Terui. Flight path optimization considering operational constraints. *Transactions of the Japan Society for Aeronautical and Space Sciences*, 58(4):192–200, 2015.

[10] M. Jalalian and C. Dadkhah. A comparison of classical and metaheuristic optimization methods for aircraft trajectory planning. In *2017 25th Iranian Conference on Electrical Engineering (ICEE)*, pages 1651–1656. IEEE, 2017.

[11] Robert F. Stengel. *Flight dynamics*. Princeton university press, 2015.

[12] J. Foley, J. Tu, G. H. Yeoh, and C. Liu. *Computational fluid dynamics: a practical approach*. Butterworth-Heinemann, 2012.

[13] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

[14] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[15] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):4213, 2014.

[16] N. Wiebe, A. Kapoor, and K. M. Svore. Quantum algorithm for bootstrapping and refinancing. *Quantum Information & Computation*, 15(3-4):238–280, 2015.

[17] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[18] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[19] Tom M. Apostol. *Mathematical analysis*. Addison-Wesley, 1974.

[20] Ilya M. Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802, 1967.

[21] Art B. Owen. The quasi-monte carlo method. In *Handbook of computational statistics*, pages 311–345. Springer, 2003.

[22] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes: The art of scientific computing*. Cambridge university press, 2007.

[23] Fred Glover, Gary Kochenberger, and Yu Du. A tutorial on formulating and using qubo models. *arXiv preprint arXiv:1811.11538*, 2018.

[24] Mark W. Johnson, Mohammad H. S. Amin, Suzanne Gildert, Trevor Lanting, Firas Hamze, Neil Dickson, R. Harris, Andrew J. Berkley, J. Johansson, Paul Bunyk, et al. Quantum annealing with manufactured spins. *Nature*, 473(7346), 2011.

[25] John D. Anderson, Jr. *Fundamentals of Aerodynamics*. McGraw-Hill Education, 6th edition, 2017.

[26] Bernard Etkin and Lloyd Duff Reid. *Dynamics of Flight: Stability and Control*. Wiley, 3rd edition, 1996.

[27] Frank M. White. *Fluid Mechanics*. McGraw-Hill, 7th edition, 2011.

[28] John D. Anderson, Jr. *Modern Compressible Flow: With Historical Perspective*. McGraw-Hill, 3rd edition, 2003.

[29] Robert C. Nelson. *Flight Stability and Automatic Control*. McGraw-Hill, 2nd edition, 1998.

[30] Arthur E. Bryson and Yu-Chi Ho. *Applied Optimal Control: Optimization, Estimation and Control*. CRC Press, 1975.

[31] Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 3rd edition, 1987.

[32] James R. Munkres. *Analysis on Manifolds*. Addison-Wesley, 1991.

[33] Michael Spivak. *Calculus on Manifolds: A Modern Approach to Classical Theorems of Advanced Calculus*. Westview Press, 1976.

[34] Serge Lang. *Calculus of Several Variables*. Springer, 3rd edition, 1987.

[35] James Stewart. *Calculus: Early Transcendentals*. Cengage Learning, 8th edition, 2015.

[36] James Stewart. *Multivariable Calculus*. Cengage Learning, 8th edition, 2016.

[37] James Stewart. *Calculus: Concepts and Contexts*. Cengage Learning, 3rd edition, 2005.

[38] James Stewart. *Single Variable Calculus: Early Transcendentals*. Cengage Learning, 7th edition, 2012.

[39] Stephen Boyd and Lieve Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[40] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[41] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[42] Tim Head, MechCoder, Gilles Louppe, et al. scikit-optimize/scikit-optimize, 2021.

[43] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[44] M. J. D. Powell. The bobyqa algorithm for bound constrained optimization without derivatives. Cambridge NA Report NA2009/06, University of Cambridge, 2009.

[45] R. Becker and L. A. Wolsey. On the crossing number of a graph. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, 2008.

[46] Jun John Sakurai and Jim Napolitano. *Modern quantum mechanics*. Cambridge University Press, 2017.

[47] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355, 1998.

[48] Andrew Lucas. Ising formulations of many np problems. *Frontiers in physics*, 2:5, 2014.

[49] Román Orus, Samuel Mugel, and Enrique Lizaso. Quantum computing for finance: Overview and prospects. *Reviews in Physics*, 4:100028, 2019.

[50] Stuart Harwood, Claudio Gambella, Dimitar Trenev, Phillip Simon, Sam Fone, Sam Gane, Andreas Bärtschi, and Stephan Eidenbenz. Formulating and solving routing problems on quantum computers. *IEEE Transactions on Quantum Engineering*, 2:1–17, 2021.

[51] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[52] Yudong Cao, Jonathan Romero, Jonathan P. Olson, Matthias Degroote, Peter D. Johnson, Mária Kieferová, Ian D. Kivlichan, Tim Menke, Borja Peropadre, Nicolas P. D. Sawaya, et al. Quantum chemistry in the age of quantum computing. *Chemical reviews*, 119(19):10856–10915, 2019.

[53] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C. Benjamin, and Xiao Yuan. Quantum computational chemistry. *Reviews of Modern Physics*, 92(1):015003, 2020.

[54] D-Wave Systems. The d-wave advantage system: An application-focused performance benchmark. Technical report, D-Wave Systems Inc., Sep 2020.

[55] Vicky Choi. Minor-embedding in adiabatic quantum computation. *Quantum Information Processing*, 7(5):193–209, 2008.

[56] Katherine Klymko, Rajiv Kumar Manish, and Travis S Humble. Finding the optimal minor-embedding and chain strength for a d-wave quantum annealer. *Quantum Science and Technology*, 7(1), 2021.

[57] Kyle Boothby, Andrew D King, and A Roy. Fast clique minor-embedding in native chimera hardware. *Quantum Information Processing*, 15(1), 2016.

Apéndices

Apéndice A: Tabla de parámetros del Jacobiano

Valores numéricos de las variables utilizados para el cálculo del Jacobiano (N=53).

Índice i	v_i ($\mathrm{m\,s^{-1}}$)	γ_i (rad)	m_i (kg)	t_i (s)	s_i (m)	Cz_i	λ_i
0	128,6111	0,041 121 2	60 000,00	0,000	0,000	0,730 998 4	1,000 000 0
1	136,7234	2,357 475	59 987,33	19,878	2791,075	0,565 680 0	0,003 376 8
2	139,5766	1,463 918	59 948,49	54,138	7532,520	0,581 445 3	0,988 333 2
3	151,9533	1,983 589	59 900,18	89,319	12 616,615	0,484 308 5	0,363 105 4
4	137,5948	2,049 337	59 882,68	118,727	16 864,579	0,615 072 3	0,286 373 3
5	144,6064	1,382 418	59 838,14	155,352	22 051,905	0,549 139 2	0,944 055 6
6	157,9770	2,282 196	59 790,75	188,666	27 025,859	0,500 980 8	0,508 646 3
7	165,2785	1,798 581	59 756,07	214,962	31 282,666	0,415 433 5	0,937 665 5
8	176,2942	2,011 179	59 718,06	241,458	35 792,521	0,423 994 5	0,669 496 0
9	186,2265	1,623 401	59 681,91	267,713	40 559,450	0,329 382 0	0,901 020 1
10	201,3053	1,753 887	59 640,84	294,018	45 638,940	0,343 423 7	0,920 074 0
11	215,9504	1,558 583	59 601,20	318,881	50 828,194	0,243 517 6	0,961 154 0
12	223,4818	1,402 146	59 567,40	345,281	56 630,281	0,287 326 4	0,580 343 6
13	210,6844	1,389 911	59 552,60	373,582	62 765,968	0,270 196 0	0,066 941 4
14	183,2715	1,268 196	59 548,99	406,643	69 224,390	0,424 124 3	0,066 837 3
15	191,7700	0,754 476	59 499,65	454,681	78 278,722	0,353 734 4	0,998 570 1
16	215,0883	1,164 775	59 436,57	501,387	87 633,005	0,328 594 5	0,581 999 9
17	207,0902	1,557 546	59 415,93	531,768	94 059,609	0,291 568 5	0,236 920 9
18	205,4337	1,662 019	59 392,34	557,597	99 385,732	0,379 074 0	0,967 326 6
19	206,1377	2,474 555	59 369,73	578,545	103 692,300	0,292 532 2	0,307 633 6
20	201,9844	2,535 868	59 354,97	595,311	107 110,354	0,413 710 9	0,865 159 5
21	198,1272	2,684 455	59 340,09	611,733	110 392,964	0,328 486 2	0,338 695 1
22	196,5610	2,029 996	59 321,43	630,528	114 096,738	0,427 489 1	0,935 804 5
23	201,1868	2,079 021	59 295,06	651,509	118 265,459	0,349 567 5	0,797 231 9
24	203,9076	1,658 515	59 269,52	674,420	122 906,978	0,401 649 4	0,770 780 7
25	208,1503	1,606 368	59 240,74	699,901	128 154,941	0,347 092 6	0,837 160 5
26	209,0635	2,066 627	59 217,10	722,630	132 892,687	0,408 856 7	0,637 774 2
27	209,3965	2,160 879	59 196,45	742,016	136 945,904	0,340 621 2	0,932 990 6
28	211,2554	2,128 335	59 174,63	761,014	140 938,997	0,414 024 3	0,776 818 6
29	204,5425	1,465 633	59 158,08	784,834	145 872,739	0,376 959 8	0,356 959 3
30	200,9845	1,709 762	59 135,89	811,591	151 299,384	0,473 046 9	0,962 746 9
31	205,7364	1,870 584	59 107,73	835,191	156 093,142	0,374 446 2	0,895 106 8
32	206,3696	2,208 297	59 085,78	855,725	160 321,086	0,481 735 4	0,790 656 2
33	206,3256	2,247 994	59 065,90	874,365	164 164,410	0,370 564 6	0,933 488 5
34	208,2782	2,030 546	59 043,63	893,733	168 177,614	0,481 677 9	0,953 109 6
35	209,5615	2,395 760	59 022,44	912,398	172 073,227	0,376 767 3	0,939 822 6
36	206,6332	2,273 092	59 006,13	930,055	175 743,637	0,503 208 7	0,636 592 8
37	205,3361	1,650 319	58 985,45	951,820	180 221,992	0,419 970 2	0,959 676 8
38	207,5626	1,885 643	58 959,73	975,409	185 087,722	0,521 404 7	0,958 493 9
39	208,4137	2,038 572	58 937,22	996,440	189 458,943	0,407 995 6	0,960 082 1
40	208,5414	1,781 175	58 915,12	1018,058	193 963,447	0,526 002 0	0,912 431 4
41	205,2877	1,604 021	58 894,18	1042,602	199 037,397	0,452 140 6	0,688 630 0
42	203,7409	1,320 317	58 868,22	1071,538	204 950,985	0,554 001 5	0,989 202 6
43	209,3816	0,882 977	58 828,26	1110,633	213 045,643	0,469 602 6	0,993 479 1
44	210,1224	1,313 549	58 793,27	1149,325	221 157,018	0,552 174 7	0,736 894 4
45	205,2721	1,627 524	58 770,86	1177,670	227 048,748	0,480 561 8	0,871 098 3
46	206,0819	0,863 668	58 736,84	1214,561	234 639,350	0,566 908 7	0,998 161 7
47	209,6307	0,952 178	58 693,86	1260,090	244 096,995	0,507 482 4	0,988 045 1
48	209,8229	1,206 811	58 659,27	1298,469	252 144,245	0,579 623 9	0,943 427 9
49	208,7912	1,064 246	58 627,42	1334,665	259 717,719	0,523 131 5	0,992 501 8
50	208,9617	0,830 102	58 588,59	1378,639	268 902,154	0,592 962 3	0,999 347 4
51	208,7071	0,866 535	58 546,74	1427,022	279 005,029	0,552 122 6	0,995 132 3
52	209,5264	0,420 128	58 486,31	1499,370	294 143,645	0,596 308 6	0,989 111 8

Con estos valores numéricos, se construyó la matriz Jacobiana J_{num} de dimensiones $260 \times$

364. El cálculo del rango de esta matriz mediante descomposición en valores singulares (SVD) arrojó un valor de $\boxed{260}$.

Este resultado confirma que las 260 ecuaciones de estado son linealmente independientes en el punto evaluado. Por lo tanto, el número de variables independientes (grados de libertad) del sistema es:

$$\text{Variables Independientes} = \text{Número Total de Variables} - \text{Rango}(J) = 364 - 260 = 104$$

Apéndice B: Código de ϕ

```
from math import sin, tan, sqrt, pow, asin, atanh, log, exp, pi, nan
import numpy as np

# --- Constantes Globales ---
numero_de_dimensiones = 104
N_puntos = numero_de_dimensiones // 2 + 1
Cx_0 = 0.014
k = 0.09
Cz_max = 0.7
S_REF = 120.0
γ = 0.06 / 3600.0
Zp_I = 10000 * 0.3048
Zp_F = 36000 * 0.3048
m_I = 60000.0
CAS_I = 250 * 0.5144444444444445
VMO = 350 * 0.5144444444444445
MMO = 0.82
M_CRZ = 0.80
L = 400000.0
s_F = L
Vz_min = 1.52400
g_0 = 9.80665
CI = 30.0 / 60.0
m_0 = m_I
t_0 = 0.0
s_0 = 0.0
λ_0 = 1.0
Ts_0 = 288.15
ρ_0 = 1.225
L_z = -0.0065
R = 287.05287
α_0 = -g_0 / R / L_z

def Zp(i, Zp_I, Zp_F, N_puntos):
    return Zp_I + i * (Zp_F - Zp_I) / N_puntos


def F_N_MCL(i, Zp_I, Zp_F, N_puntos):
    zp = Zp(i, Zp_I, Zp_F, N_puntos)
    return 140000.0 - 2.53 * zp / 0.3048


def ρ(i, Zp_I, Zp_F, N_puntos, ρ_0, Ts_0, L_z, α_0):
    zp = Zp(i, Zp_I, Zp_F, N_puntos)
    return ρ_0 * ((Ts_0 + L_z * zp) / Ts_0)**(α_0 - 1)


def M(l, v, Zp_I, Zp_F, N_puntos, R, Ts_0, L_z):
    zp = Zp(l, Zp_I, Zp_F, N_puntos)
    return v[l] / sqrt(1.4 * R * (Ts_0 + L_z * zp))


def CAS(l, v, Zp_I, Zp_F, N_puntos, R, Ts_0, L_z, α_0):
    zp = Zp(l, Zp_I, Zp_F, N_puntos)

    base = 1 + (v[l]**2 / (7 * R * (Ts_0 + L_z * zp)))
    if base < 0: return nan

    arg = (7 * R * Ts_0) * (((Ts_0 / (Ts_0 + L_z * zp))**-α_0 *
                            (pow(base, 3.5) - 1) + 1)**(1 / 3.5) - 1)
    if arg < 0: return nan
    return sqrt(arg)


def calculate_initial_values(Zp_I, Zp_F, N_puntos, CAS_I, Ts_0, L_z, R, α_0, m_0, g_0, ρ_0, S_REF, Cx_0, k, M_CRZ):
    zp_i = Zp(0, Zp_I, Zp_F, N_puntos)
    TAS_I = sqrt(7 * R * (Ts_0 + L_z * zp_i) * ((((Ts_0 + L_z * zp_i) / Ts_0)**-α_0 * ((1 + CAS_I**2 / (7 * R * Ts_0))**3.5 - 1) + 1)**(1 /
       3.5) - 1))
    v_0 = TAS_I
    rho_0_val = ρ(0, Zp_I, Zp_F, N_puntos, ρ_0, Ts_0, L_z, α_0)
    Cz_0 = m_0 * g_0 / (0.5 * rho_0_val * v_0**2 * S_REF)
    f_n_0 = F_N_MCL(0, Zp_I, Zp_F, N_puntos)

    drag_term_initial = Cx_0 + k * Cz_0

    arg_asin = (f_n_0 - 0.5 * rho_0_val * v_0**2 * S_REF * drag_term_initial) / (m_0 * g_0)
    if arg_asin < -1.0 or arg_asin > 1.0: return nan, nan, nan, nan, nan
    γ_0 = asin(arg_asin)

    rho_F_val = ρ_0 * ((Ts_0 + L_z * Zp_F) / Ts_0)**(α_0 - 1)
    v_F = M_CRZ * sqrt(1.4 * R * (Ts_0 + L_z * Zp_F))
    return v_0, Cz_0, γ_0, rho_F_val, v_F


def m_ip(i, m, γ, v, Cz, λ, Zp_I, Zp_F, N_puntos, ρ_0, Ts_0, L_z, α_0, S_REF, g_0, Cx_0, k, η):

    zp_i = Zp(i, Zp_I, Zp_F, N_puntos)
    zp_ip1 = Zp(i+1, Zp_I, Zp_F, N_puntos)
    rho_i = ρ(i, Zp_I, Zp_F, N_puntos, ρ_0, Ts_0, L_z, α_0)
    rho_ip1 = ρ(i+1, Zp_I, Zp_F, N_puntos, ρ_0, Ts_0, L_z, α_0)
    f_n_i = F_N_MCL(i, Zp_I, Zp_F, N_puntos)
    f_n_ip1 = F_N_MCL(i+1, Zp_I, Zp_F, N_puntos)

    if sin(γ[i])==0 or tan(γ[i])==0 or tan(γ[i+1])==0 or v[i]==0 or f_n_ip1==0: return nan

    A = (v[i+1]-v[i])/(zp_ip1-zp_i)
    L = (-g_0/v[i+1] + (λ[i]*f_n_i)/(m[i]*v[i]*sin(γ[i]))
        - (0.5*rho_i*v[i]*S_REF*(Cx_0+k*Cz[i]**2))/(m[i]*sin(γ[i]))
        - g_0/v[i])
    H = (4*sin(γ[i+1])/(rho_ip1*S_REF)) * ((γ[i+1]-γ[i])/(zp_ip1-zp_i) +
        g_0/(2*v[i+1]**2*tan(γ[i+1])) - (rho_i*S_REF*Cz[i])/(4*m[i]*sin(γ[i])) +
        g_0/(2*v[i]**2*tan(γ[i])))
    I = (-2*v[i+1]*sin(γ[i+1])/(η*f_n_ip1)) * (1/(zp_ip1-zp_i))

    term_J_1 = (2*v[i+1]*sin(γ[i+1])/(η*f_n_ip1)) * (m[i]/(zp_ip1-zp_i))
    term_J_2 = (v[i+1]*sin(γ[i+1])*λ[i]*f_n_i)/(f_n_ip1*v[i]*sin(γ[i]))
    J = term_J_1 - term_J_2

    inner_sqrt = (A**2*v[i+1]**2*sin(γ[i+1])**2 - A*f_n_ip1*I*v[i+1]*sin(γ[i+1]) -
                  A*L*v[i+1]**2*sin(γ[i+1])**2 - 0.25*Cx_0*H**2*S_REF**2*k*rho_ip1**2*v[i+1]**4 +
                  0.25*f_n_ip1**2*I**2 + 0.5*f_n_ip1*H**2*J*S_REF*k*rho_ip1*v[i+1]**2 +
                  0.5*f_n_ip1*I*L*v[i+1]*sin(γ[i+1]) + 0.25*L**2*v[i+1]**2*sin(γ[i+1])**2)

    if inner_sqrt < 0: return nan
```

```
    numerator_1 = -2.0*A*v[i+1]*sin(γ[i+1]) + f_n_ip1*I + L*v[i+1]*sin(γ[i+1])
    numerator_2 = 2.0*sqrt(inner_sqrt)
    denominator = H**2*S_REF*k*rho_ip1*v[i+1]**2
    if denominator == 0: return nan
    return (numerator_1 + numerator_2)/denominator

def Cz_ip(i, m, γ, v, Cz, Zp_I, Zp_F, N_puntos, ρ_0, Ts_0, L_z, α_0, S_REF, g_0):
    zp_i = Zp(i, Zp_I, Zp_F, N_puntos)
    zp_ip1 = Zp(i+1, Zp_I, Zp_F, N_puntos)
    rho_i = ρ(i, Zp_I, Zp_F, N_puntos, ρ_0, Ts_0, L_z, α_0)
    rho_ip1 = ρ(i+1, Zp_I, Zp_F, N_puntos, ρ_0, Ts_0, L_z, α_0)
    if sin(γ[i]) == 0 or tan(γ[i+1]) == 0 or tan(γ[i]) == 0 or rho_ip1 == 0: return nan

    return (2 * m[i+1] * sin(γ[i+1]) * ((2 * γ[i+1] - 2 * γ[i]) / (zp_ip1 - zp_i) - (rho_i * S_REF * Cz[i]) / (2 * m[i] * sin(γ[i]))
                                        + g_0 / (v[i+1]**2 * tan(γ[i+1])) + g_0 / (v[i]**2 * tan(γ[i]))))/(rho_ip1 * S_REF)


def s_ip(i, s, γ, Zp_I, Zp_F, N_puntos):
    zp_i = Zp(i, Zp_I, Zp_F, N_puntos)
    zp_ip1 = Zp(i+1, Zp_I, Zp_F, N_puntos)
    if tan(γ[i+1])==0 or tan(γ[i])==0: return nan
    return s[i] + 0.5 * ((zp_ip1 - zp_i) / tan(γ[i+1]) + (zp_ip1 - zp_i) / tan(γ[i]))

def t_ip(i, t, γ, v, Zp_I, Zp_F, N_puntos):
    zp_i = Zp(i, Zp_I, Zp_F, N_puntos)
    zp_ip1 = Zp(i+1, Zp_I, Zp_F, N_puntos)
    if sin(γ[i+1])==0 or sin(γ[i])==0: return nan
    return t[i] + 0.5 * ((zp_ip1 - zp_i) / (v[i+1] * sin(γ[i+1])) + (zp_ip1 - zp_i) / (v[i] * sin(γ[i])))

def λ_ip(i, m, γ, v, λ, Zp_I, Zp_F, N_puntos, η):
    zp_i = Zp(i, Zp_I, Zp_F, N_puntos)
    zp_ip1 = Zp(i+1, Zp_I, Zp_F, N_puntos)
    f_n_i = F_N_MCL(i, Zp_I, Zp_F, N_puntos)
    f_n_ip1 = F_N_MCL(i+1, Zp_I, Zp_F, N_puntos)
    if f_n_ip1==0 or v[i]==0 or sin(γ[i])==0: return nan

    term1 = -2 * (v[i+1] * sin(γ[i+1])) / (η * f_n_ip1)
    term2 = (m[i+1] - m[i]) / (zp_ip1 - zp_i)
    term3 = (v[i+1] * sin(γ[i+1]) * λ[i] * f_n_i) / (f_n_ip1 * v[i] * sin(γ[i]))
    return (term1 * term2) - term3


def f(x, N_puntos, Zp_I, Zp_F, CAS_I, Ts_0, L_z, R, α_0, m_0, g_0, ρ_0, S_REF, Cx_0, k, M_CRZ, η, VMO, MMO, Cz_max, Vz_min, s_F, CI):
    mid = len(x) // 2
    x1 = x[:mid]
    x2 = x[mid:]

    N_calc = len(x1) + 1
    if N_calc != N_puntos: return nan

    v_0_val, Cz_0_val, γ_0_val, ρ_F, v_F = calculate_initial_values(Zp_I, Zp_F, N_puntos, CAS_I, Ts_0, L_z, R, α_0, m_0, g_0, ρ_0, S_REF, Cx_0
        , k, M_CRZ)
    if np.isnan(v_0_val): return nan

    v = np.zeros(N_calc, dtype=np.float64)
    γ = np.zeros(N_calc, dtype=np.float64)
    v[0], γ[0] = v_0_val, γ_0_val
    for i in range(len(x1)):
        v[i+1] = x1[i]
        γ[i+1] = x2[i] * pi / 180.0

    m = np.zeros(N_calc, dtype=np.float64); m[0] = m_0
    s = np.zeros(N_calc, dtype=np.float64); s[0] = s_0
    t = np.zeros(N_calc, dtype=np.float64); t[0] = t_0
    λ = np.zeros(N_calc, dtype=np.float64); λ[0] = λ_0
    Cz = np.zeros(N_calc, dtype=np.float64); Cz[0] = Cz_0_val

    for i in range(N_calc - 1):
        if v[i+1]*sin(γ[i+1]) < Vz_min: return nan
        cas_i1 = CAS(i+1, v, Zp_I, Zp_F, N_puntos, R, Ts_0, L_z, α_0)
        if np.isnan(cas_i1) or cas_i1 > VMO: return nan

        m_i1 = m_ip(i, m, γ, v, Cz, λ, Zp_I, Zp_F, N_puntos, ρ_0, Ts_0, L_z, α_0, S_REF, g_0, Cx_0, k, η)
        if np.isnan(m_i1): return nan
        m[i+1] = m_i1

        Cz_i1 = Cz_ip(i, m, γ, v, Cz, Zp_I, Zp_F, N_puntos, ρ_0, Ts_0, L_z, α_0, S_REF, g_0)
        if np.isnan(Cz_i1): return nan
        Cz[i+1] = Cz_i1

        λ_i1 = λ_ip(i, m, γ, v, λ, Zp_I, Zp_F, N_puntos, η)
        if np.isnan(λ_i1): return nan
        λ[i+1] = λ_i1

        s_i1 = s_ip(i, s, γ, Zp_I, Zp_F, N_puntos)
        if np.isnan(s_i1): return nan
        s[i+1] = s_i1

        t_i1 = t_ip(i, t, γ, v, Zp_I, Zp_F, N_puntos)
        if np.isnan(t_i1): return nan
        t[i+1] = t_i1

        M_i1 = M(i+1, v, Zp_I, Zp_F, N_puntos, R, Ts_0, L_z)
        if not (0 <= λ[i+1] <= 1 and Cz[i+1] <= Cz_max and M_i1 <= MMO):
            return nan

    N_idx = N_calc - 1
    f_n_final = F_N_MCL(N_idx, Zp_I, Zp_F, N_puntos)

    A = (-ρ_F * S_REF * Cx_0) / (2 * m[N_idx]) - (6 * k * m[N_idx] * g_0**2) / (ρ_F * S_REF * v[N_idx]**4)
    B = (16 * k * m[N_idx] * g_0**2) / (ρ_F * S_REF * v[N_idx]**3)
    C = (f_n_final / m[N_idx]) - (12 * k * m[N_idx] * g_0**2) / (ρ_F * S_REF * v[N_idx]**2)

    inner_sqrt_D = B**2 - 4 * A * C
    if inner_sqrt_D < 0: return nan
    D = sqrt(inner_sqrt_D)

    arg_atanh1 = (2*A*v[N_idx] + B)/D
    arg_atanh2 = (2*A*v_F + B)/D
    if abs(arg_atanh1) >= 1 or abs(arg_atanh2) >= 1: return nan
    t_B = t[N_idx] + (2/D) * (atanh(arg_atanh1) - atanh(arg_atanh2))

    m_B = m[N_idx] - η * λ[N_idx] * f_n_final * (t_B - t[N_idx])

    log_arg = (D-2*A*v_F-B)/(D-2*A*v[N_idx]-B)
    if log_arg <= 0 or A == 0: return nan
    s_B = s[N_idx] + (1/A) * log(log_arg) - ((B+D)/(2*A)) * (t_B - t[N_idx])

    if s[N_idx] <= s_B and s_B <= s_F:
        m_F_val = m_B * exp((-2 * η * g_0 * sqrt(k * Cx_0) / v_F) * (s_F - s_B))
        return -m_F_val + CI*(t_B - s_B/v_F)
    else:
        return nan

def ϕ(x):
    return f(x, N_puntos, Zp_I, Zp_F, CAS_I, Ts_0, L_z, R, α_0, m_0, g_0, ρ_0, S_REF, Cx_0, k, M_CRZ, η, VMO, MMO, Cz_max, Vz_min, s_F, CI)
```

Código para copiar y pegar en entorno de desarrollo Python

Apéndice C: Código de prueba de verificación numérica provista por Airbus

```
import numpy as np
from math import sin, asin, sqrt, tan, log, exp, atanh

numero_de_dimensiones = 104
N = int(numero_de_dimensiones / 2 + 1)

# Parámetros
Cx_0 = 0.014
k = 0.09
Cz_max = 0.7
S_REF = 120
η = 0.06 / 3600 # kg(N.s)^-1
Zp_I = 10000 * 0.3048
Zp_F = 36000 * 0.3048
pi = np.pi

m_I = 60000
CAS_I = 250 * 0.5144444444444445
VMO = 350 * 0.5144444444444445
MMO = 0.82
M_CRZ = 0.80
L = 400000
s_F = L

Vz_min = 300 * 0.3048 / 60.0
g_0 = 9.80665
CI = 30/60

Ts_0 = 288.15
ρ_0 = 1.225
L_z = -0.0065
R = 287.05287
α_0 = -g_0/(R*L_z)

m_0 = m_I
t_0 = 0
s_0 = 0
λ_0 = 1

def Fun(x):
    array_entrada = np.array(x)
    x1, x2 = np.array_split(array_entrada, 2)
    N_ = len(x1)+1
    x1 = np.array(x1)
    x2 = np.array(x2)

    def Zp(i):
        return Zp_I + i*(Zp_F - Zp_I)/(N_-1)

    def F_N_MCL(i):
        return 140000 - 2.53*(Zp(i)/0.3048)

    def ρ(i):
        return ρ_0 * ((Ts_0 + L_z*Zp(i))/Ts_0)**(α_0 - 1)

    def TAS_from_CAS(CAS_, Zp_):
        return sqrt(7*R*(Ts_0 + L_z*Zp_) * ((((Ts_0 + L_z*Zp_)/Ts_0)**(-α_0)) * (((1 + CAS_**2/(7*R*Ts_0))**3.5)-1)+1)**(1/3.5)-1)

    TAS_I = TAS_from_CAS(CAS_I, Zp_I)
    v_0 = TAS_I
    Cz_0 = (m_0*g_0)/(0.5*ρ(0)*v_0**2*S_REF)
    γ_0 = asin((F_N_MCL(0)-0.5*ρ(0)*v_0**2*S_REF*(Cx_0+k*Cz_0**2))/(m_0*g_0))
    ρ_F = ρ_0*((Ts_0 + L_z*Zp_F)/Ts_0)**(α_0 -1)
    v_F = M_CRZ*sqrt(1.4*R*(Ts_0 + L_z*Zp_F))

    v = np.zeros(N_)
    γ = np.zeros(N_)
    m = np.zeros(N_)
    t = np.zeros(N_)
    s = np.zeros(N_)
    λ = np.zeros(N_)
    Cz = np.zeros(N_)

    v[0] = v_0
    γ[0] = γ_0
    m[0] = m_0
    t[0] = t_0
    s[0] = s_0
    λ[0] = λ_0
    Cz[0] = Cz_0

    for i in range(1, N_):
        v[i] = x1[i-1]
        γ[i] = x2[i-1]*pi/180.0

# ================================================================================
#       VALORES DADOS POR AIRBUS PARA PROBAR LA EXPRESIÓN NUMÉRICAMENTE
# ================================================================================
    v[N_-1] = 223.61
    m[N_-1] = 59042
    t[N_-1] = 880.8
    s[N_-1] = 168717.2
    λ[N_-1] = 1
# ================================================================================
# ================================================================================

    def A():
        return - (ρ_F * S_REF * Cx_0)/(2*m[N_-1]) - (6*k*m[N_-1]*g_0**2)/(ρ_F*S_REF*v[N_-1]**4)
    def B():
        return (16*k*m[N_-1]*g_0**2)/(ρ_F*S_REF*v[N_-1]**3)
    def C():
        return (F_N_MCL(N_-1)/m[N_-1]) - (12*k*m[N_-1]*g_0**2)/(ρ_F*S_REF*v[N_-1]**2)
    def D():
        return sqrt(B()**2 - 4*A()*C())

    t_B = t[N_-1] + (2/D())*(atanh((2*A()*v[N_-1]+B())/D()) - atanh((2*A()*v_F+B())/D()))
    m_B = m[N_-1] - η*λ[N_-1]*F_N_MCL(N_-1)*(t_B - t[N_-1])
    s_B = s[N_-1] + (1/A())*log((D()-2*A()*v_F-B())/(D()-2*A()*v[N_-1]-B())) - ((B()+D())/(2*A()))*(t_B - t[N_-1])
    m_F = m_B*exp((-2*η*g_0*sqrt(k*Cx_0)/v_F)*(s_F - s_B))
    t_F = t_B+(s_F - s_B)/v_F

    _val  = -m_F + CI*(t_B - s_B/v_F)
    return  _val

def fun(*args):
    if len(args) == 1 and isinstance(args[0], (list, tuple, np.ndarray)):
        args = np.array(args[0])
    else:
        args = np.array(args)
    try:
        a = -Fun(args)
        result = np.where(a > 0, a, 70000)
        return result
    except Exception as e:
        print(f"Error al procesar los argumentos: {e}")
        return None

x_prueba = np.zeros((N-1)*2)
phi_check = Fun(x_prueba)
print("Valor esperado        :   58273  .65") # kg/s
print("Valor de    en el CHECK =", phi_check)
```

Código para copiar y pegar en entorno de desarrollo que ejecute Python