OPTIMIZACIÓN DE TRAYECTORIAS MEDIANTE TRAYECTORIAS TIPO

José Manuel Picón Tagua

9 de diciembre de 2009

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

El concepto de operaciones basadas en trayectorias (Trajectory-Based-Operations, TBO) propuesto por NextGen y SESAR para el futuro sistema de gestión del tráfico aéreo (Air Traffic Management, ATM) permitirá, entre otros, aumentar la capacidad del espacio aéreo, mejorar la seguridad y volar trayectorias que permitan disminuir el consumo de combustible.

Un problema importante en este escenario es la optimización de trayectorias de aviones sujetas a ligaduras. El problema de trayectorias óptimas ha sido tratado extensamente en el pasado pero, en general, su implementación es compleja. Por una parte, los modelos empleados de actuaciones de la aeronave suelen contener importantes simplificaciones no reflejando convenientemente la dinámica del sistema. Por ejemplo, se simplifica el modelo dinámico de la aeronave despreciando la variación de la masa [1] o se emplean modelos de actuaciones de la aeronave con polares incompresibles [2]. Por otra parte, las leyes obtenidas suelen ser bastante complicadas de volar lo cual exige un cambio profundo en la estructura del espacio aéreo y en los sistemas embarcados de hoy en día. Por ejemplo, se obtienen leyes que exigen una variación continua del número de Mach [3] [4].

Por ello, se considera el uso de trayectorias tipo, las cuales son flight intents[1] que pueden ser descritos por un número pequeño de parámetros, estableciendo como libres algunos de ellos. Estas trayectorias tipo modelan las trayectorias considerando los segmentos usualmente volados por las aerolíneas, incluyendo procedimientos estándares, regulaciones de tráfico aéreo, etc., asegurando por tanto que pueden ser voladas. El uso de las trayectorias tipo permite formular la optimización de trayectorias como un problema de optimización multivariable sujeta a ligaduras.

[1] El *aircraft intent* representa la información necesaria y suficiente que describe completa y unívocamente cómo opera la aeronave durante un período de tiemopo determinado. A diferencia del *aircraft intent*, el *flight intent* puede no contener suficiente información para determinar la trayectoria con unicidad. Puede ser tan general que sus instrucciones sean únicamente volar de un punto a otro (sin especificar cómo), o contener tanto detalle que coincida con un *aircraft intent*.

Las trayectorias tipo han sido anteriormente empleadas por Slattery y Zhao [5] quienes empleaban perfiles de descenso predefinidos (rápido, nominal y lento) para obtener trayectorias de descenso en procesos de automatización del tráfico aéreo. Barrer [6] consideró también trayectorias tipo, llamadas path objects, como medio para estandarizar procedimientos de vuelo. También se han empleado en el ámbito de la resolución de conflictos. Por ejemplo, Vilaplana [7] definió una maniobra de desviación lateral para resolver conflictos en ruta o Isaacson y Robinson [8] las usaban para resolver conflictos dentro del área terminal. Como trabajo previo en el departamento, las trayectorias tipo han sido usadas para modelar trayectorias globales, de aeropuerto a aeropuerto, por de Augusto [9] o para la optimización en la resolución de conflictos por Valenzuela y Rivas [10].

1.2. Objetivos

En vista de lo anterior, los objetivos del presente proyecto son dos:

1. La implementación y evaluación de algoritmos para la optimización multivariable sujeta a restricciones.

2. La aplicación de dichos algoritmos en la optimización de trayectorias modeladas mediante trayectorias tipo y su comparación con trayectorias óptimas obtenidas mediante otros métodos.

Capítulo 2

Algoritmos de optimización

En este capítulo se cubre el primero de los objetivos del proyecto. En primer lugar (sección 2.1) se presentan los algoritmos que no requieren del cálculo de derivadas. En particular se consideran los algoritmos de enjambre de partículas, recocido simulado y algoritmos genéticos. Todos ellos se aplican a tres problemas de prueba con el fin de elegir apropiadamente los valores de los parámetros que los configuran.

En las secciones 2.2 y 2.3 se presentan algoritmos que tratan de hallar el óptimo resolviendo problemas lineales y cuadráticos, respectivamente, por lo que requieren del cálculo de derivadas.

Finalmente, en la sección 2.4 se comparan todos los algoritmos presentados en base a tres problemas de prueba con el fin de tener un criterio para escoger aquel algoritmo que mejor resuelva un tipo de problema.

Antes de dar paso a los algoritmos de optimización implementados, conviene recordar el problema de la optimización. De forma relajada, se puede enunciar como encontrar el mínimo (o máximo) de una función sujeta a restricciones. El enunciado matemático se puede ver en las ecuaciones (2.1). Dentro del problema de la optimización podemos encontrarnos con funciones objetivo lineales o no lineales, al igual que sucede con las restricciones.

$$\begin{aligned} min \quad & f(\mathbf{x}) \\ s.a \quad & g_j(\mathbf{x}) \leq 0, \quad j = 1,\ldots,q \\ & h_j(\mathbf{x}) = 0, \quad j = q+1\ldots,m \\ & l_i \leq x_i \leq u_i, \quad i = 1,\ldots,n \end{aligned} \tag{2.1}$$

2.1. Algoritmos de optimización sin cálculo de derivadas

Aunque existen otros tipos de algoritmos sin cálculo de derivadas, un amplio grupo de éstos se encuadran dentro de la computación evolutiva, la cual es una

rama de la inteligencia artificial, o más concretamente de la inteligencia computacional, que implica problemas de optimización. Los métodos de este tipo son iterativos, tales como el crecimiento o el desarrollo en una población. Esta población es seleccionada y procesada para alcanzar un determinado fin. Habitualmente, estos métodos están inspirados en mecanismos biológicos o físicos.

El uso de estas técnicas surgió en la década de 1950, aunque no fue hasta 10 años después cuando tres interpretaciones distintas de los principios de Darwin se desarrollaron de forma independiente dando lugar al origen de la computación evolutiva.

La programación evolutiva fue introducida por Lawrence J. Fogel en EEUU, mientras que John Henry Holland llamó a su método «algoritmo genético». En Alemania, Ingo Rechenberg y Hans-Paul Schwefel introdujeron estrategias evolutivas.

Las técnicas evolutivas implican principalmente algoritmos de optimización metaheurística, de los que a grandes rasgos podemos nombrar:

- Algoritmos evolutivos: algoritmos genéticos, programación evolutiva o estrategias evolutivas.
- Inteligencia cultural: colonia de hormigas, enjambre de partículas u otros similares.

Los algoritmos evolutivos son una rama de la computación evolutiva que implican la implementación de mecanismos inspirados en la evolución biológica tales como reproducción, mutación, selección natural o supervivencia. Las soluciones candidatas juegan al papel de individuos en una población, y la función objetivo juega el papel del entorno en el cual las soluciones «sobreviven». La evolución de la población usa repetidamente los operadores descritos anteriormente.

Múltiples aspectos de los procesos evolutivos son estocásticos. La reproducción o mutación están fuertemente influenciadas por números aleatorios. Por otra parte, los procesos de selección pueden ser deterministas o estocásticos. En el segundo caso, los mejores individuos tienen mayores posibilidades de sobrevivir que el resto, pero incluso los individuos más débiles tienen algunas posibilidades de sobrevivir. Para más información puede consultar las referencias [11], [12] ó [13].

Si bien los algoritmos genéticos y el enjambre de partículas pertenecen al grupo de algoritmos evolutivos, también implementaremos el recocido simulado, que es un algoritmo de búsqueda metaheurística y cuya descripción detallada se encuentra en la sección 2.1.2.

2.1.1. Enjambre de partículas (PSO)

La optimización por enjambre de partículas (*Particle Swarm Optimization*, PSO) es una técnica estocástica de optimización basada en poblaciones, desarrollada por el Dr. Eberhart y el Dr. Kennedy en 1995 [14], inspirada en el comportamiento social de las bandadas de pájaros. El PSO comparte algunos aspectos

con el resto de algoritmos evolutivos. El sistema es inicializado con una población aleatoria de soluciones (posición y velocidad) y busca óptimos actualizando la posición en cada iteración. Las soluciones, llamadas partículas, recorren el espacio de las variables siguiendo a las partículas óptimas. La ventaja que posee frente a otros algoritmos evolutivos reside en el bajo número de parámetros a ajustar y en la simplicidad de las operaciones.

El PSO se encuadra dentro de los métodos que hacen uso de la inteligencia de enjambre. Cada partícula recibe información del óptimo que hasta el momento ha encontrado todo el enjambre, lo que es conocido como componente social. Al mismo tiempo, cada partícula hace uso del óptimo que ha encontrado hasta el momento, lo que se conoce como componente cognitiva. En la Figura 2.1 se puede ver el comportamiento habitual, donde entran en juego tanto los óptimos como la inercia de las partículas.

El algoritmo es:

```
Inicializar todas las partículas
Mientras no se alcancen las condiciones de parada
  Para cada partícula
    Calcular la velocidad
    Limitar la velocidad
    Actualizar la posición
    Almacenar el mínimo de la partícula
  End
  Almacenar el mínimo global de la función objetivo
End
```

Para cada partícula m y en cada iteración k la velocidad y la posición se determinan según las expresiones (2.2)-(2.4).

$$\mathbf{v}_m^{[k+1]} \leftarrow \omega^{[k]}\mathbf{v}_m^{[k]} + c_1\text{rnd}(\hat{\mathbf{x}}_m^{[k]} - \mathbf{x}_m^{[k]}) + c_2\text{rnd}(\hat{\mathbf{g}}^{[k]} - \mathbf{x}_m^{[k]}), \tag{2.2}$$

$$\mathbf{v}_m^{[k+1]} \leftarrow min\left(\mathbf{v}_m^{[k+1]}, v_{max}\right) \cdot \frac{\mathbf{v}_m^{[k+1]}}{\left\|\mathbf{v}_m^{[k+1]}\right\|}, \tag{2.3}$$

$$\mathbf{x}_m^{[k+1]} \leftarrow \mathbf{x}_m^{[k]} + \mathbf{v}_m^{[k+1]}, \tag{2.4}$$

$$m = 1, \ldots, m_{max},$$

donde $\mathbf{v}_m^{[k]}$ es el vector velocidad de la partícula m en la iteración k, $\omega^{[k]}$ es un coeficiente de inercia que varía linealmente según

$$\omega^{[k]} = \omega_0 + (\omega_T - \omega_0)\frac{(k-1)}{(k_{max}-1)}$$

siendo ω_0 y ω_T las inercias inicial y final respectivamente y k_{max} es el número máximo de iteraciones, c_1 es la componente cognitiva o peso del vector distancia

al mínimo de la partícula, rnd es un escalar aleatorio entre 0 y 1, $\hat{\mathbf{x}}_m^{[k]}$ es el mejor valor encontrado por la partícula m hasta la iteración k, $\mathbf{x}_m^{[k]}$ es el vector posición de la partícula m en la iteración k, c_2 es la componente social o peso del vector distancia al óptimo de todas las partículas, $\hat{\mathbf{g}}^{[k]}$ es el mejor valor encontrado por el enjambre hasta la iteración k, v_{max} es el módulo máximo de la velocidad para cada partícula y cada iteración y m_{max} es el número de partículas.

Haciendo un recuento de parámetros, tenemos: ω_0, ω_T, c_1, c_2, v_{max} y m_{max} . Teniendo en cuenta la referencia [15] podemos encontrar, al igual que en multitud de artículos similares, los valores típicos: $c_1 \approx 2$, $c_2 \approx 2$, $\omega_0 \approx 1$, v_{max} en torno al 20 % del mínimo dominio admisible de x_n y $\omega_T \approx 0{,}4$ para que no se acumule energía en el sistema y no diverjan las partículas. Por tanto, en este algoritmo queda como parámetro de ajuste el número total de partículas m_{max}.

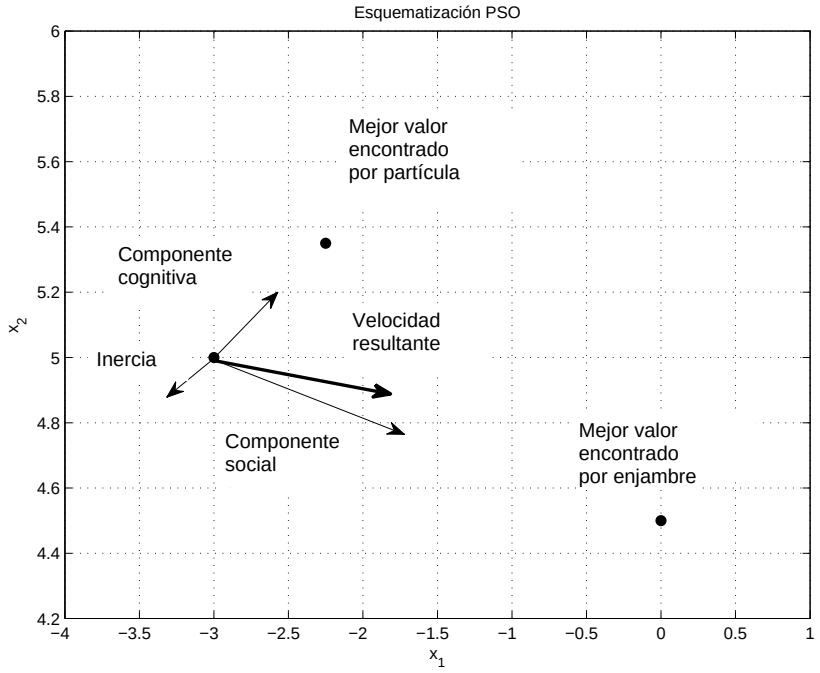


Figura 2.1: Ejemplo del algoritmo PSO.

2.1.2. Recocido simulado (SA)

La optimización por recocido simulado (*Simulated Annealing*, SA) es un algoritmo de búsqueda metaheurística para problemas de optimización global, es decir, encontrar una buena aproximación al óptimo global de una función en un espacio de búsqueda grande. El nombre e inspiración viene del proceso de recocido del acero, que consiste en calentar y luego enfriar controladamente un material para aumentar el tamaño de sus cristales y reducir sus defectos. El calor causa que los átomos se salgan de sus posiciones iniciales (un mínimo local de energía) y se muevan aleatoriamente; el enfriamiento lento les da mayores probabilidades de

encontrar configuraciones con menor energía que la inicial. El método fue descrito de forma independiente por S. Kirkpatrick, C. D. Gelatt y M. P. Vecchi en 1983 [16], y por V. Černý en 1985 [17]. El método es una adaptación de algoritmo Metropolis-Hastings, un método de Monte Carlo.

En cada iteración, el recocido simulado considera algunos vecinos del estado actual s, y probabilísticamente decide entre cambiar el sistema al estado s' o quedarse en el estado s. Los posibles s' se obtienen de forma aleatoria con una distribución de probabilidad uniforme en el círculo centrado en s y con radio que decrece linealmente con las iteraciones

$$\text{Radio} \leftarrow \text{paso}_0 + (\text{paso}_T - \text{paso}_0)\frac{k-1}{k_{max}-1},$$

donde paso_0 es el radio de la circunferencia centrada en s que envuelve a los posibles s' en la primera iteración, mientras que paso_T es el radio de la circunferencia centrada en s que envuelve a los posibles s' en la última iteración, y decrece linealmente entre ambos valores. Las probabilidades se escogen para que el sistema tienda finalmente a estados de menor energía. Este proceso se repite hasta que se alcanzan las condiciones de parada. La probabilidad de hacer la transición al nuevo estado s' es una función $P(\delta E, T)$ de la diferencia de energía $\delta E = E(s') - E(s)$ entre los dos estados, y de la variable T, llamada temperatura. Si δE es negativo, es decir, la transición disminuye la energía, el movimiento es aceptado con probabilidad $P = 1$. Una cualidad importante del método SA es que la probabilidad de transición P es siempre distinta de cero, aún cuando δE sea positivo, es decir, el sistema puede pasar a un estado de mayor energía (peor solución) que el estado actual. Esta cualidad intenta evitar que el sistema se quede atrapado en un óptimo local (la situación se idealiza en la Figura 2.2). Cuando la temperatura tiende al mínimo, la probabilidad tiende a cero asintóticamente. Así, cada vez el algoritmo acepta menos movimientos que aumenten la energía. La función de probabilidad está definida en la ecuación (2.5).

$$P = min(e^{-\frac{\delta E}{T}}, 1). \tag{2.5}$$

Por tanto, el algoritmo es:

```
Inicializar la partícula
Mientras no se alcancen las condiciones de parada
  Calcular la energía de s
  Obtener s' y calcular su energía
  Obtener la probabilidad de salto a s'
  Saltar a s' o mantenerse en s
  Almacenar el mínimo de la función objetivo
End
```

Tenemos tres parámetros en este algoritmo: paso_0, paso_T y T.

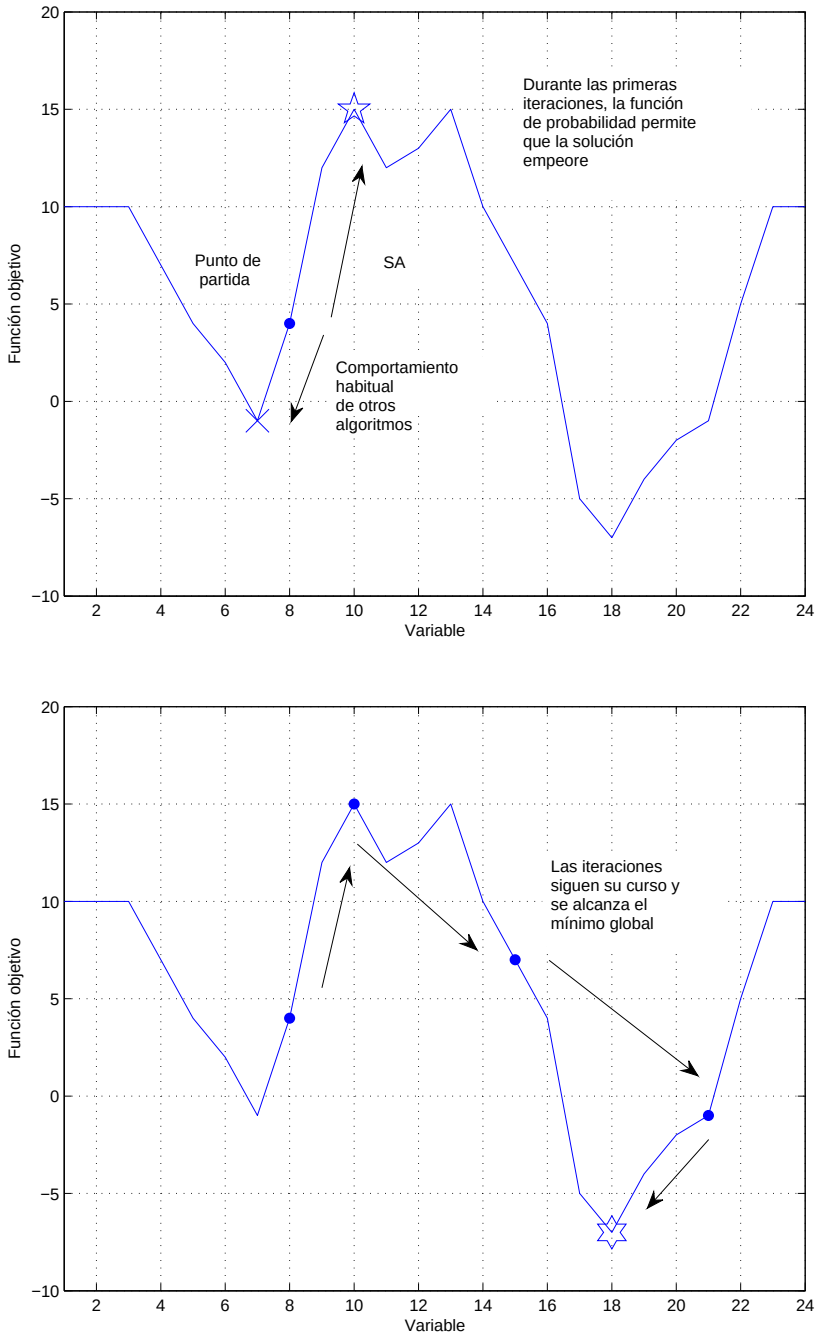


Figura 2.2: Ejemplo del algoritmo SA.

2.1.3. Algoritmo Genético (GA)

Un algoritmo genético (*Genetic Algorithm*, GA) es una técnica de búsqueda usada en computación para encontrar soluciones exactas o aproximadas a problemas de optimización o búsqueda. Los algoritmos genéticos son un tipo particular

de algoritmos evolutivos que usan técnicas inspiradas en la teoría de la evolución, como la mutación, selección o cruce. El algoritmo se basa en dos operaciones clave: reproducción y selección.

En la reproducción se obtienen nuevas generaciones de soluciones, llamadas hijos, a partir de las anteriores, llamadas padres. Para ello se hace uso del cruce genético, es decir, se combinan las características (variables) de las soluciones padre para dar lugar a hijos. Aunque el cruce genético se puede hacer de múltiples formas, el algoritmo implementado extrae un número entre cero y el número de variables de la función objetivo. Se seleccionan de forma aleatoria dos padres y del primero de ellos se toman las variables hasta el aleatorio anterior, mientras que del segundo se toman desde el aleatorio hasta el final. También es una técnica común la mutación, en la que las características de las soluciones hijo se ven alteradas con respecto a las de los padres mediante la introducción de factores aleatorios. La implementación de la mutación consiste en extraer un vector aleatorio, con componentes entre $-0{,}5$ y $0{,}5$ y de la misma longitud que $\mathbf{x}$. Tal vector se multiplica componente a componente por el coeficiente de mutación y por $\mathbf{x}$ y el resultado se suma a $\mathbf{x}$.

En la selección se evalúa la validez de las soluciones de una generación para obtener los padres de la siguiente generación. La mayoría de los métodos evalúan toda la población, mientras que otros hacen un muestreo estadístico y extrapolan los resultados al resto de la población ya que el proceso de evaluación puede ser computacionalmente muy costoso. En el algoritmo implementado se evalúa toda la población.

En la Figura 2.3 se muestra el funcionamiento del algoritmo. Cada columna de la figura representa un padre/hijo mientras que cada fila representa una iteración. La implementación del algoritmo considera las soluciones iniciales como soluciones padre. Como vemos, la mutación hace que el algoritmo pueda obtener hijos diferentes a sus padres, y la selección retiene a las mejores soluciones en cada iteración.

El algoritmo es:

```
Mientras no se alcancen las condiciones de parada
  Combinar las soluciones padre
  Introducir mutaciones
  Obtener los hijos
  Seleccionar las mejores soluciones hijo
  Almacenar el mínimo de la función objetivo
End
```

Como parámetros de este algoritmo tenemos: número de padres en cada iteración, número de hijos que se obtienen de cada generación y coeficiente de mutación (toma valor entre 0 y 1).

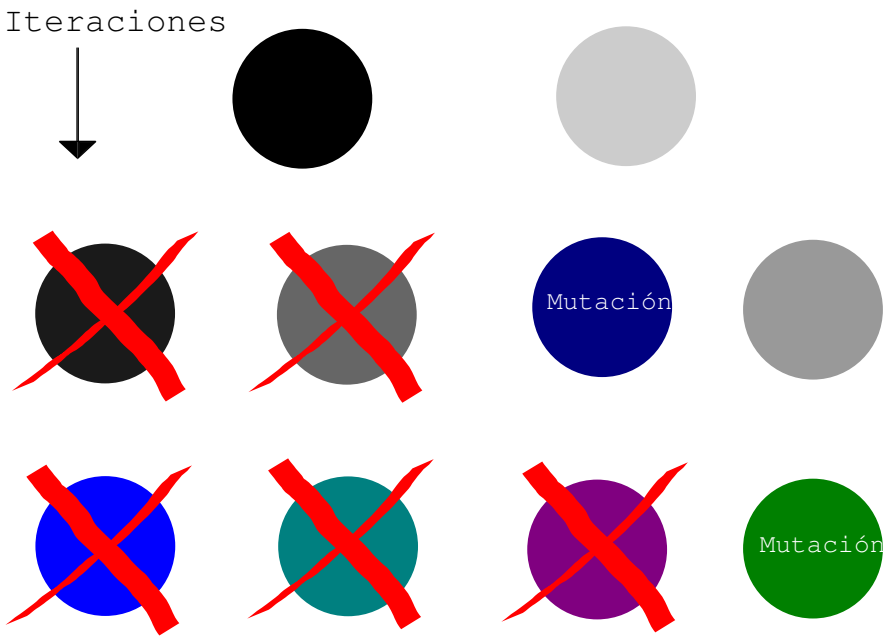


Figura 2.3: Ejemplo del algoritmo GA.

2.1.4. Optimización sujeta a ligaduras: ε level comparison

Hasta ahora hemos presentado algoritmos que, sin calcular derivadas, son capaces de resolver problemas de optimización sin restricciones. En esta sección se van a presentar modificaciones que se pueden hacer a tales algoritmos para adaptarlos a problemas con restricciones.

La optimización no lineal sujeta a ligaduras no lineales sin cálculo de derivadas puede ser clasificada en cuatro grupos en función de la forma en que se tratan las restricciones (más detalles en la referencia [18]):

1. Las restricciones se usan únicamente para decidir si un punto es admisible o no. Son los conocidos como *death penalty methods*. Se empieza con uno o más puntos admisibles y continúa buscando nuevos puntos en la región admisible. Cuando se genera un nuevo punto y éste es no admisible, bien se corrige para que sea admisible o bien se descarta. Estos métodos presentan el problema de la generación de soluciones iniciales admisibles.

2. La violación de restricciones, la cual es la suma de la violación de todas las restricciones, es combinada con la función objetivo. El método de la función de penalización está dentro de esta categoría. En él se define una función objetivo extendida añadiendo la violación de restricciones a la función objetivo como una penalización. La principal dificultad es la selección de un valor apropiado para el coeficiente de penalización que ajusta la magnitud de la penalización.

3. La función objetivo y la violación de restricciones son usadas de forma independiente. En esta categoría, función objetivo y violación de restricciones son optimizadas en orden de forma que la violación de restricciones precede a la función objetivo. De esta forma, un punto situado en la región admisible es preferible a otro situado fuera de dicha región sin importar el valor de la función objetivo. Takahama y Sakai (referencias [19] y [20]) introdujeron el ε *constrained method*.

4. La función objetivo y las restricciones son optimizadas por métodos multiobjetivo. En esta categoría, los problemas de optimización restringida son resueltos como problemas multiobjetivo en los cuales la función objetivo y las restricciones son optimizados. En muchos casos, resolver problemas de optimización multiobjetivo es más difícil que resolver problemas de optimización simple.

De acuerdo con los resultados presentados en las referencias [18], [19] y [20] se decidió usar los métodos ε *level comparison*, pertenecientes al grupo 3 anteriormente expuesto.

Los problemas de optimización no lineal sujeta a ligaduras son muy importantes y comunes. Consideremos el problema (2.6) donde $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ es un vector de dimensión n, $f(\mathbf{x})$ es la función objetivo, $g_j(\mathbf{x})$ y $h_j(\mathbf{x})$ son q y $l-q$ restricciones de desigualdad e igualdad respectivamente. Las funciones f, g_j y h_j son reales y puede ser lineales o no lineales.

$$\begin{array}{lll} min & f(\mathbf{x}) & \\ s.a & g_j(\mathbf{x}) \leq 0, & j = 1, \ldots, q \\ & h_j(\mathbf{x}) = 0, & j = q+1 \ldots, l \\ & l_i \leq x_i \leq u_i, & i = 1, \ldots, n \end{array} \tag{2.6}$$

Se define la función violación de restricciones $\Phi(\mathbf{x})$ según las expresiones (2.7) ó (2.8), es decir, $\Phi(\mathbf{x}) = 0$ si $\mathbf{x}$ es admisible y $\Phi(\mathbf{x}) > 0$ si $\mathbf{x}$ es no admisible. En la expresión (2.8) p es un número entero mayor que cero y par.

$$\Phi(\mathbf{x}) = max\left(0, g_1(\mathbf{x}), \ldots, g_q(\mathbf{x}), \|h_{q+1}(\mathbf{x})\|, \ldots, \|h_l(\mathbf{x})\|\right) \tag{2.7}$$

$$\Phi(\mathbf{x}) = \sum_{j}^{q} \|max\left(0, g_j(\mathbf{x})\right)\|^p + \sum_{q+1}^{l} \|h_j(\mathbf{x})\|^p \tag{2.8}$$

El ε *level comparison* (operador $<_\varepsilon$) se define como una relación de ordenación para el par función objetivo y violación de restricciones $(f(\mathbf{x}), \Phi(\mathbf{x}))$. Si la violación de restricciones es mayor que cero, el punto está en la región no admisible y su valor es bajo. Sean f_1 (f_2) y Φ_1 (Φ_2) las funciones objetivo y violación de restricciones en el punto $\mathbf{x_1}$ ($\mathbf{x_2}$) respectivamente. Para todo $\varepsilon \geq 0$, el ε *level comparison* se define como sigue:

$$(f_1,\Phi_1) <_\varepsilon (f_2,\Phi_2) \Leftrightarrow \begin{cases} f_1 < f_2 & \text{si } \Phi_1,\Phi_2 \leq \varepsilon \\ f_1 < f_2 & \text{si } \Phi_1 = \Phi_2 \\ \Phi_1 < \Phi_2 & \text{en otro caso} \end{cases} \tag{2.9}$$

$$(f_1,\Phi_1) \leq_\varepsilon (f_2,\Phi_2) \Leftrightarrow \begin{cases} f_1 \leq f_2 & \text{si } \Phi_1,\Phi_2 \leq \varepsilon \\ f_1 \leq f_2 & \text{si } \Phi_1 = \Phi_2 \\ \Phi_1 < \Phi_2 & \text{en otro caso} \end{cases} \tag{2.10}$$

Para $\varepsilon = \infty$, $<_\infty$ y $\leq_\infty$ equivalen a las comparaciones ordinarias $<$ y $\leq$ entre los valores de la función objetivo. Con las definiciones (2.9) y (2.10) de los operadores $<_\varepsilon$ y $\leq_\varepsilon$ podemos extender los tres algoritmos (PSO, SA, GA) anteriormente expuestos para optimizar problemas sin ligaduras a problemas de optimización no lineal sujeta a restricciones lineales o no lineales, de igualdad o desigualdad. Los algoritmos extendidos se denominan como εPSO, εSA y εGA respectivamente.

2.1.5. Análisis de sensibilidad de parámetros

Ahora que se han presentado los parámetros e interioridades de los métodos propuestos vamos a realizar un breve análisis para ver la respuesta de cada uno de los métodos ante la variación de los distintos parámetros que lo componen. Para ello, vamos a someter a los tres algoritmos a problemas de prueba:

1. Minimizar $x_1^2 + x_2^2$ (Figura 2.4) sujeta a $x_1 \geq 0$, $x_2 \geq 0$ y $x_1 \cdot x_2 \geq 10$. La solución exacta de este problema es $x_1 = x_2 = \sqrt{10}$ y $f = 20$

2. Minimizar $x_1^2 + x_2^2$ (Figura 2.4) sujeta a $x_1 \geq 0$, $x_2 \geq 0$ y $x_1 \cdot x_2 = 10$. La solución exacta de este problema es $x_1 = x_2 = \sqrt{10}$ y $f = 20$

3. Minimizar la función de Rastrigin (extraída de [21]) (Figura 2.5)

$$\sum_{i=1}^{2} \left(x_i^2 - 10cos\left(2\pi x_i\right) + 10\right)$$

   sujeta a $x_1 \geq 0$ y $x_2 \geq 0$. La solución exacta es $x_1 = x_2 = 0$ y $f = 0$.

PSO

Vamos a someter al algoritmo PSO a los tres problemas de prueba descritos anteriormente, para ver la influencia del número de partículas en el método. Para ello vamos a lanzar cada problema 30 veces, cada vez con condiciones iniciales distintas. Los parámetros toman los valores $c_1 = c_2 = 2$ y $\omega_0 = 0{,}9$ $\omega_T = 0{,}4$ tal y como se recomendó en la introducción de este método (referencia [15]).

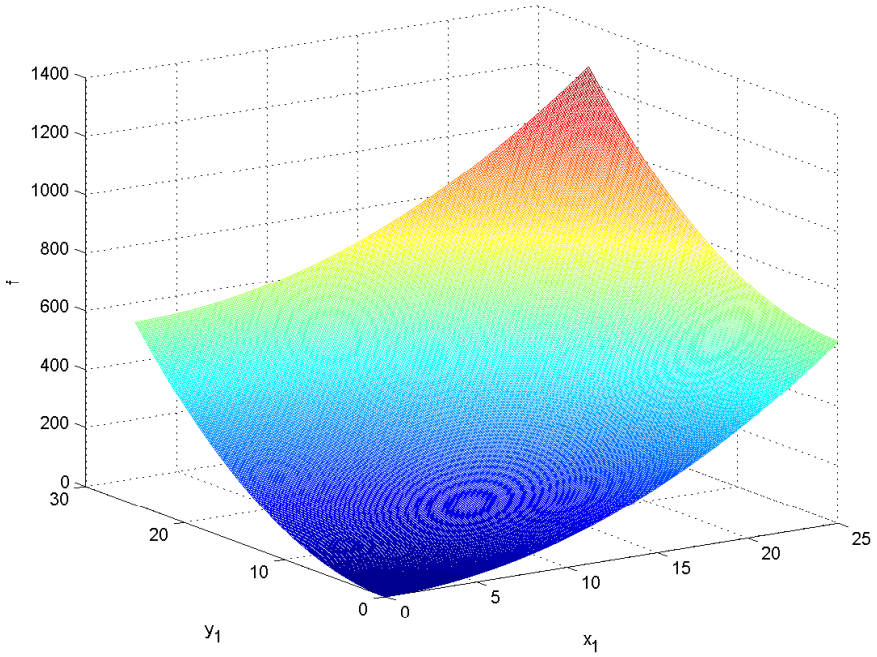


Figura 2.4: Función objetivo de prueba 1 y 2

Como condiciones iniciales, distribuimos aleatoriamente (mediante distribución uniforme) las partículas en el cuadrado $x_1 \in [0,50]$ y $x_2 \in [0,50]$ y velocidad inicial nula. Para los problemas 1 y 2 limitamos el algoritmo a 3.000 evaluaciones de la función objetivo mientras que para el 3 lo limitamos a 10.000 evaluaciones.

En la Figura 2.6 se muestra la evolución media del algoritmo, es decir, la media del valor de la función objetivo frente al número de evaluaciones de ésta. En la Tabla 2.1 se muestran media, desviación típica, mejor y peor valores que devuelve el algoritmo al final del proceso de optimización. Por mejor (peor) valor se entiende el más cercano (lejano) a la solución exacta en valor de la función objetivo. No se tiene en cuenta si la solución está dentro de la región de admisibilidad puesto que por el propio funcionamiento del ε level comparison el algoritmo devolverá, en caso de no estar la solución dentro de la región admisible, la solución más próxima a ésta.

La conclusión que podemos obtener es que el algoritmo PSO funciona mejor en todos los problemas de prueba con un número de partículas intermedio, en torno a 30. Además, vemos que en el problema 2, debido a la existencia de restricciones de igualdad, la solución del problema no es tan buena como en los problemas 1 y 3.

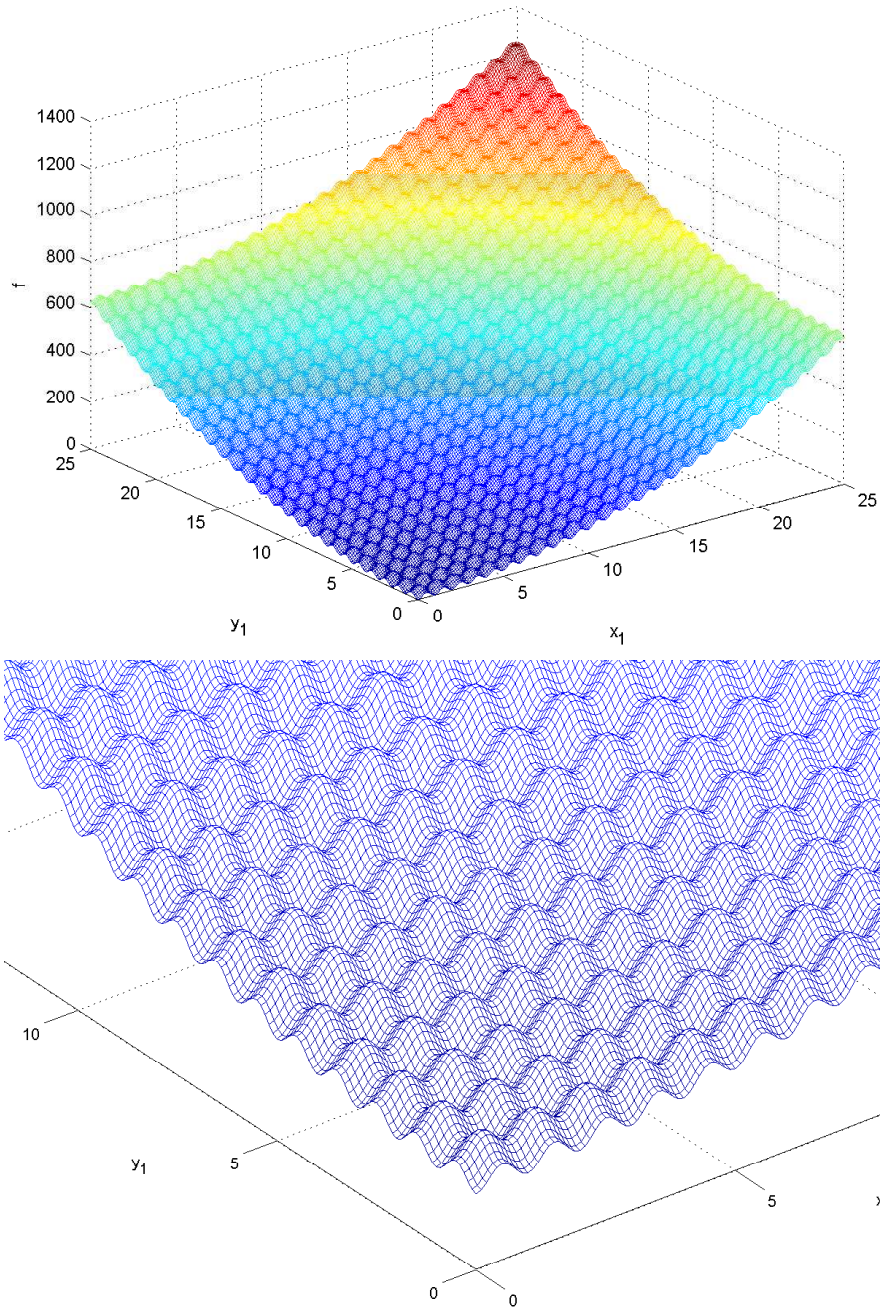


Figura 2.5: Función objetivo de prueba 3, con vista detalle.

Número de partículas	Mejor	Media	Peor	STD
Problema 1				
5	20.0000	22.5406	73.0623	10.2596
15	20.0000	20.0000	20.0000	9.6612e-006
30	20.0000	20.0001	20.0008	1.5575e-004
50	20.0000	20.0008	20.0098	0.0019
75	20.0001	20.0026	20.0283	0.0051
Problema 2				
5	19.0404	97.2332	758.2389	158.0582
15	20.0614	19.4338	11.1802	1.8798
30	19.9631	17.5336	2.5852	4.7736
50	19.9720	16.0465	0.4557	5.8716
75	19.2841	14.8397	0.0627	5.9527
Problema 3				
5	0	3.48233	35.81799	7.46267
15	0	1.16078	15.91924	2.86325
30	0	0.33165	0.99496	0.47704
50	0	0.23216	0.99496	0.42801
75	1.77636e-015	0.46431	3.97983	0.81517

Tabla 2.1: Influencia del número de partículas en el PSO. Resultados de los problemas de prueba.

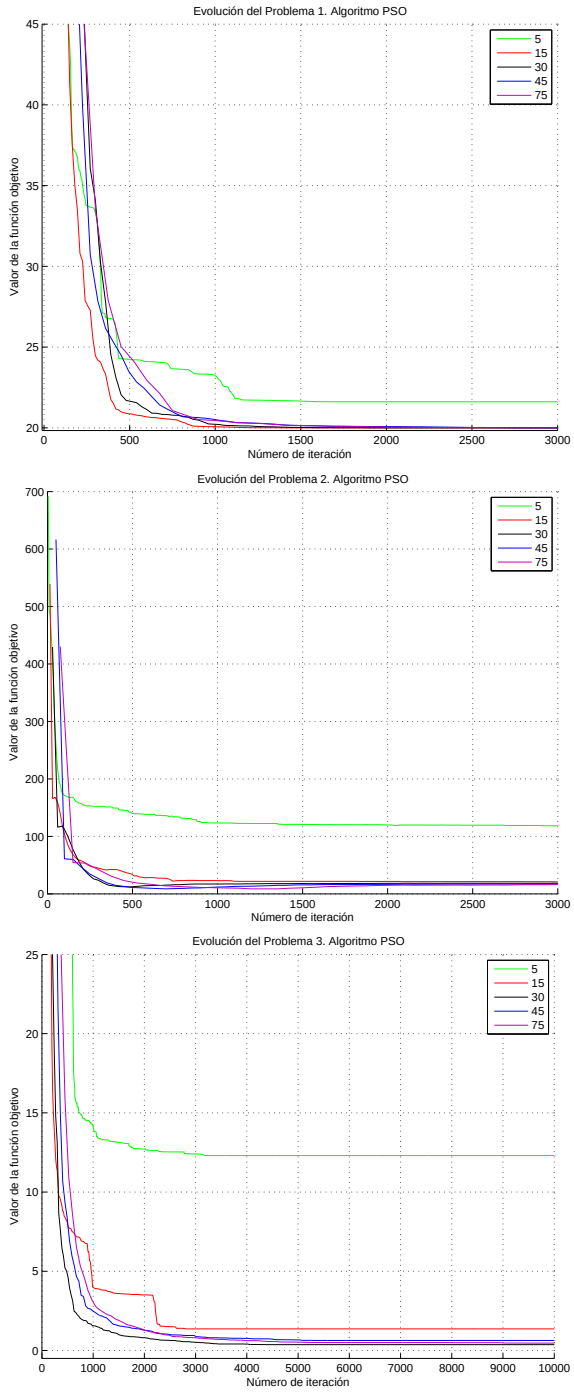


Figura 2.6: Evolución media del óptimo global en función del número de evaluaciones de la función objetivo. Algoritmo PSO.

SA

Repetimos los problemas de prueba del apartado anterior. Lanzamos 30 veces la optimización y obtenemos mejor, peor, media y desviación típica. Partimos desde puntos distribuidos aleatoriamente (mediante distribución uniforme) en la región $x_1 \in [0, 50]$ y $x_2 \in [0, 50]$. Para los problema 1 y 2 detenemos las iteraciones tras 3.000 evaluaciones de la función objetivo mientras que para el 3 lo hacemos tras 10.000 evaluaciones.

Los resultados se puede ver en la Tabla 2.2 (mejor, media, peor y desviación típica, al igual que se hizo con el PSO) y en la Figura 2.7 (evolución media del valor de la función objetivo frente a número de evaluaciones).

Los mejores resultados se obtienen para T baja y paso bajo y fuertemente decreciente, siempre que el problema contenga desigualdades. En el caso de igualdad los resultados del algoritmo son bastante pobres independientemente de los valores que tomen los parámetros.

$paso_0$, $paso_T$, T	Mejor	Media	Peor	STD
Problema 1				
5,5,10	20.0058	20.0819	20.2210	0.0571
5,5,50	20.0097	25.0153	161.9048	25.8690
5,5,90	20.0016	21.9204	51.7310	5.9688
5,.1,10	20.0013	20.0072	20.0207	0.0052
5,.1,50	20.0006	20.6310	35.0790	2.8096
5,.1,90	20.0004	22.1785	36.2857	3.9360
15,5,10	20.0042	20.1808	20.4459	0.1305
15,5,50	20.0420	20.1697	20.3714	0.1000
15,5,90	20.0391	20.2883	23.8644	0.6808
15,1,10	20.0086	20.0538	20.1465	0.0336
15,1,50	20.0088	20.3000	24.1172	0.9304
15,1,90	20.0103	20.0526	20.1374	0.0289
Problema 2				
5,5,10	20.0144	972.0686	2280.0298	641.2314
5,5,50	20.3083	539.4759	1708.9793	511.9135
5,5,90	20.0337	771.8265	2037.5088	559.6721
5,0.1,10	20.3873	940.7767	2520.5469	740.9835
5,0.1,50	19.9452	792.8558	2266.3759	614.0228
5,0.1,90	20.0639	945.1106	1954.8859	702.2009
15,5,10	18.4290	646.4251	2521.4447	785.4998
15,5,50	20.0544	466.8112	1630.8980	540.8710
15,5,90	20.1817	681.6396	2071.0372	624.7546
15,1,10	20.3287	562.1970	2196.9645	629.0544
15,1,50	19.9836	612.4802	2264.9856	660.5483
15,1,90	18.0229	648.6222	2491.1722	741.4453
Problema 3				
5,5,10	0.1586	1.0958	2.1043	0.4935
5,5,50	0.1364	1.5300	11.6226	2.0408
5,5,90	0.2241	3.0651	19.7965	4.2402
5,0.1,10	0.2739	0.9291	1.6086	0.3386
5,0.1,50	0.0080	0.8136	1.6033	0.5001
5,0.1,90	0.0082	2.5045	24.2254	5.5055
15,5,10	0.0164	2.8533	7.7420	1.7191
15,5,50	0.6660	2.8456	10.3123	1.8612
15,5,90	1.4536	3.5063	16.9043	2.7726
15,1,10	0.9586	2.7300	5.2601	1.3299
15,1,50	0.6789	2.6140	5.4629	1.4272
15,1,90	0.1533	3.0029	14.3572	2.5788

Tabla 2.2: Influencia de los parámetros del SA. Resultados de los problemas de prueba.

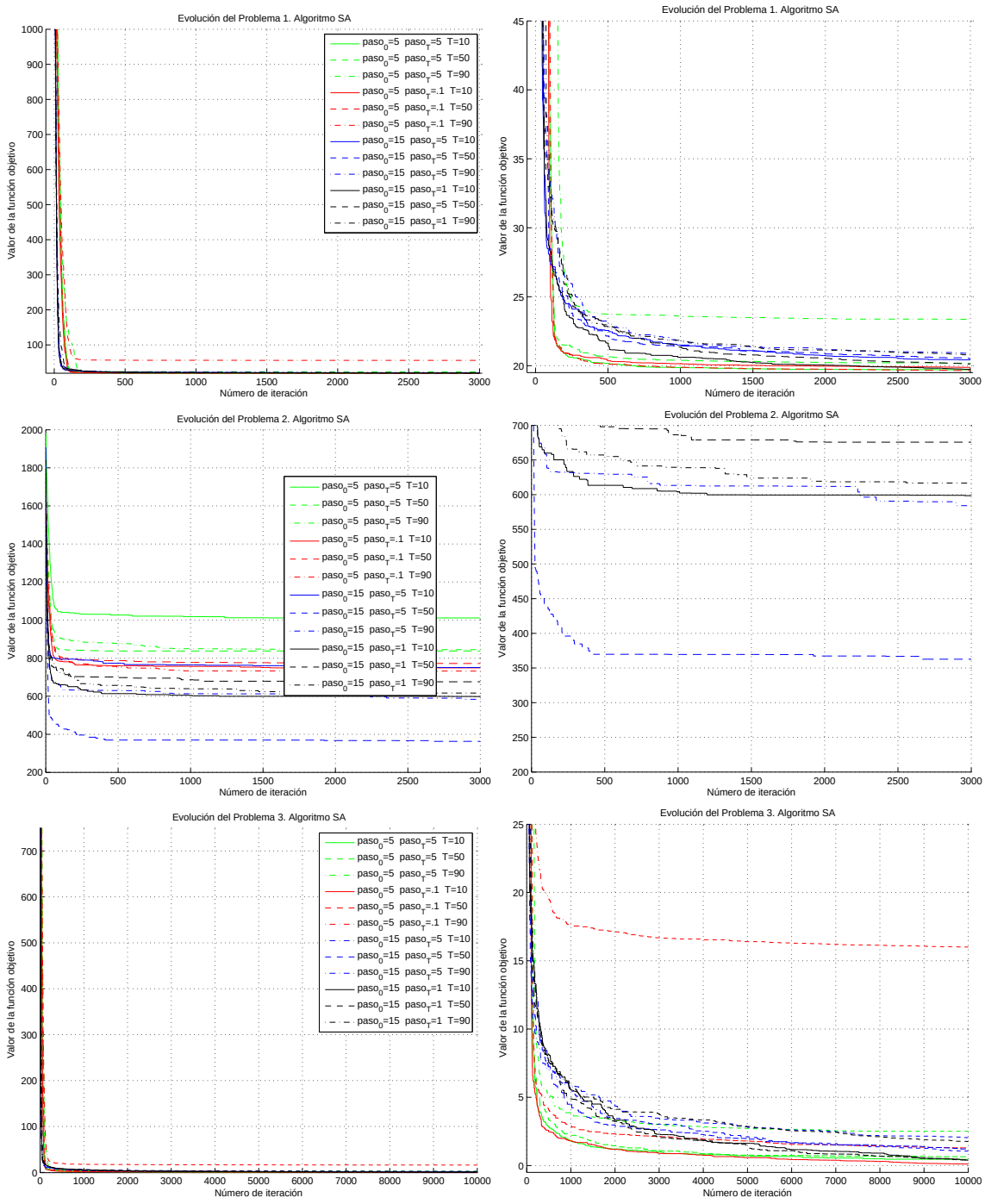


Figura 2.7: Evolución media del óptimo global en función del número de evaluaciones de la función objetivo. Algoritmo SA.

GA

Mismos problemas que en los dos apartados anteriores. Lanzamos 30 veces la optimización y obtenemos mejor aproximación, peor, media y desviación típica. Como condiciones iniciales, situamos los padres distribuidos aleatoriamente (mediante distribución uniforme) en la región $x_1 \in [0,50]$ y $x_2 \in [0,50]$. Para los problemas 1 y 2 limitamos a 3.000 evaluaciones de la función objetivo mientras que para el 3 limitamos a 10.000 evaluaciones.

Los resultados se puede ver en la Tabla 2.3. Vemos en la Figura 2.8 (evolución media del valor de la función objetivo frente al número de evaluaciones de la función objetivo) que obtenemos mejores resultados en todos los problemas con un número bajo de padres, con un número bajo de hijos (5-15). En problemas suaves (problema 1) un coeficiente bajo de mutación acelera la convergencia de la solución, mientras que para problemas no suaves o con restricciones de igualdad (problemas 2 y 3) un coeficiente de mutación bajo hace decrecer la velocidad de convergencia de la solución. Además, en el caso de problemas con restricciones de igualdad este algoritmo no se comporta de manera adecuada.

Nº padres, Nº hijos, Mutación	Mejor	Media	Peor	STD
Problema 1				
2,5,0.1	20.0055	20.0354	20.1436	0.0312
2,5,0.3	20.0198	20.1954	20.4923	0.1153
2,5,0.5	20.0212	20.3594	21.0130	0.2269
2,15,0.1	20.0097	20.0551	20.1559	0.0306
2,15,.3	20.0026	20.2659	20.6136	0.1674
2,15,.5	20.0928	20.8683	21.7489	0.4689
5,15,0.1	20.0076	20.0736	20.2425	0.0570
5,15,0.3	20.1166	20.6089	21.2992	0.3480
5,15,0.5	21.6693	86.5128	365.2632	88.0426
20,50,.1	20.0381	21.1216	23.9664	0.9522
20,50,.3	21.1824	90.3763	200.0203	56.7556
20,50,.5	21.8307	140.7911	549.0738	99.6153
Problema 2				
2,5,0.1	20.0120	28.2652	57.5903	10.2785
2,5,0.3	20.0314	59.01452	935.8561	166.7414
2,5,0.5	20.0161	26.7530	59.4998	10.2015
2,15,0.1	20.0050	23.7449	38.6395	4.5682
2,15,0.3	20.0411	136.2178	1365.1587	321.8682
2,15,0.5	20.3000	51.5909	246.2804	50.9299
5,15,0.1	20.0189	27.7320	112.8744	17.0864
5,15,0.3	19.9634	40.7527	154.7584	27.1166
5,15,0.5	19.9948	86.1250	1094.4465	203.0288
20,50,0.1	20.1158	80.3203	1052.6655	189.1536
20,50,0.3	20.7353	1352.2639	21360.0479	4006.0393
20,50,0.5	17.5968	1188.4263	13964.7464	2975.1658
Problema 3				
2,5,0.1	0.9964	2.3260	4.9752	0.9955
2,5,0.3	0	0	0	0
2,5,0.5	0	0	0	0
2,15,0.1	0.9952	1.7275	2.0012	0.4478
2,15,0.3	0	0	0	0
2,15,0.5	0	0	0	0
5,15,0.1	0	0.7702	1.0457	0.4178
5,15,0.3	0	0.2035	1.0489	0.3050
5,15,0.5	3.6547	103.1497	610.5874	145.8753
20,50,0.1	0.0123	0.6646	1.2767	0.4450
20,50,0.3	6.5066	67.9517	244.5232	51.0181
20,50,0.5	10.1205	100.5329	343.7259	76.2989

Tabla 2.3: Influencia de los parámetros del GA. Resultados de los problemas de prueba.

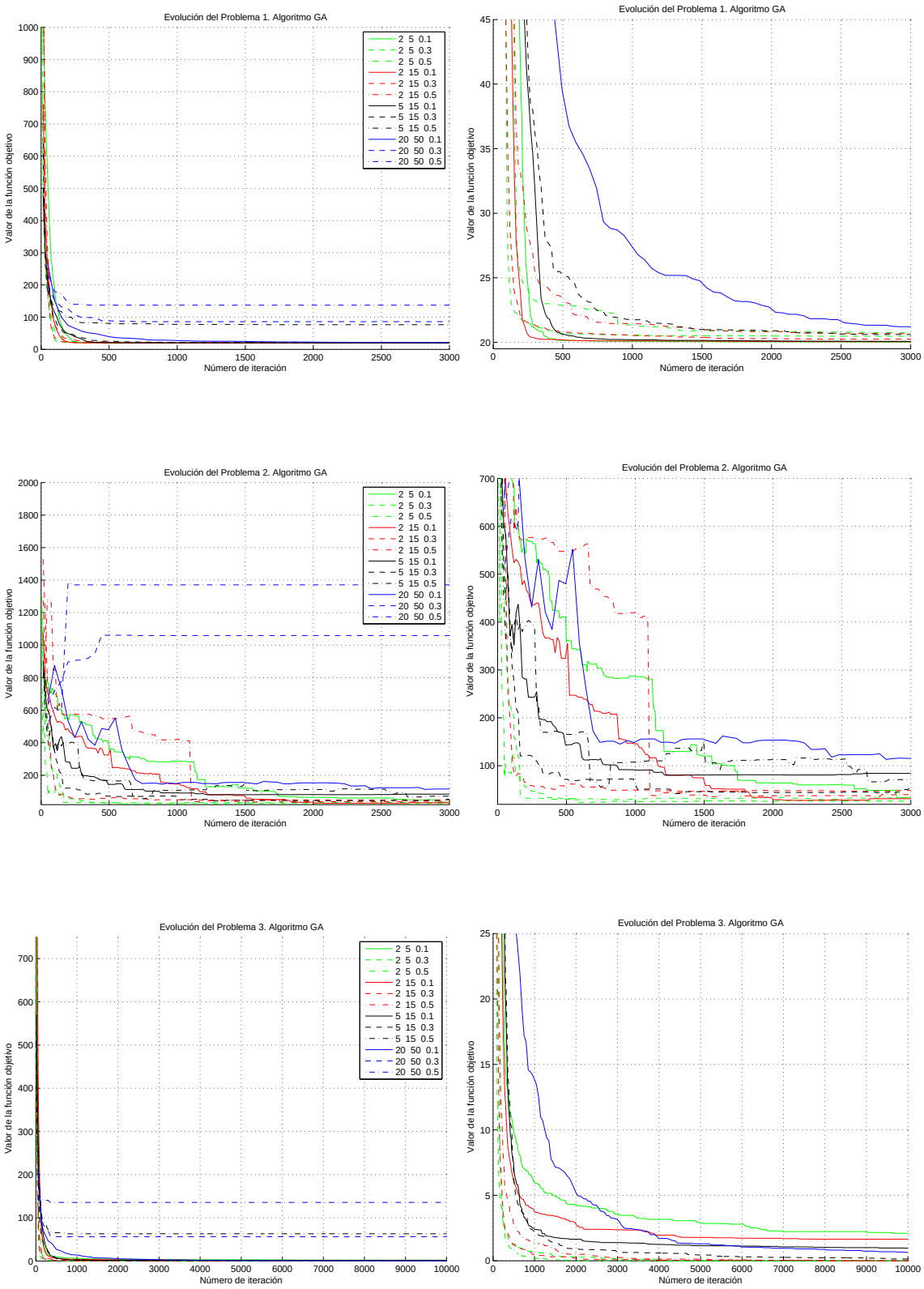


Figura 2.8: Evolución media del óptimo global en función del número de evaluaciones de la función objetivo. Algoritmo GA.

2.2. Algoritmos de optimización SLP

Se conocen como métodos de optimización LP a aquellos que resuelven problemas de optimización donde tanto la función objetivo como las ligaduras son lineales.

2.2.1. Método de Kelley

El método *Sequential Linear Programming* (SLP) consiste en linealizar las ecuaciones del problema de la minimización con el desarrollo de Taylor de las funciones en torno al punto $\mathbf{x}^0$, de forma que obtenemos las ecuaciones (2.11). A las ecuaciones linealizadas se les pueden aplicar los métodos de programación lineal. En cada iteración se linealiza en torno a la solución lineal de la iteración anterior, de forma que se tiene a la solución del problema no lineal. Para más información, puede consultar [22].

$$\begin{aligned} min \quad & f(\mathbf{x}) \simeq f(\mathbf{x}^0) + \nabla f(\mathbf{x}^0) \cdot \delta\mathbf{x} && && (2.11)\\ s.a \quad & g_j(\mathbf{x}) \simeq g_j(\mathbf{x}^0) + \nabla g_j(\mathbf{x}^0) \cdot \delta\mathbf{x} \leq 0, && j = 1,\ldots,q \\ & h_j(\mathbf{x}) \simeq h_j(\mathbf{x}^0) + \nabla h_j(\mathbf{x}^0) \cdot \delta\mathbf{x} = 0, && j = q+1\ldots,m \\ & l_i \leq x_i + \delta x_i \leq u_i, && i = 1,\ldots,n \\ & \delta\mathbf{x} = \mathbf{x} - \mathbf{x}^0 && && (2.12) \end{aligned}$$

Este algoritmo fue implementado por M. A. Herrada (profesor del Departamento de Ingeniería Aeroespacial de la Universidad de Sevilla), por lo que no nos extenderemos sobre su uso. Como parámetros de ajuste tenemos η y `fraction`, donde η es el porcentaje al que se reduce la región de búsqueda en cada iteración y `fraction` es el porcentaje de la región de admisibilidad que el método usa en la primera iteración.

2.2.2. Análisis de sensibilidad de parámetros

Aplicamos el método de Kelley a los tres problemas de prueba. Ahora no podemos limitar a un número determinado de evaluaciones de la función objetivo, sino que ajustamos las tolerancias a 10^{-4}. En las Figuras 2.9 y 2.10 podemos ver la evolución media del método de Kelley para los problemas 1 y 3, respectivamente. Para el problema 2 no se obtuvieron resultados tras 150 minutos. Vemos que la mejor combinación de parámetros es $\eta = 0{,}8$ y fraction = 0,1.

η, fraction	Mejor	Media	Peor	STD
Problema 1				
0,7, 0,1	20.0000	22.5406	73.0623	10.2596
0,7, 10	20.0000	20.0000	20.0000	9.6612e-006
0,8, 0,1	20.0000	20.0001	20.0008	1.5575e-004
0,8, 10	20.0000	20.0008	20.0098	0.0019
Problema 3				
0,7, 0,1	0	2.689e-012	1.159e-011	3.290e-012
0,7, 10	0	552.674	1672.665	578.350
0,8, 0,1	0	68.381	347.195	110.905
0,8, 10	0	754.762	1850.484	585.275

Tabla 2.4: Influencia de los parámetros en el método de Kelley. Resultados de los problemas de prueba.

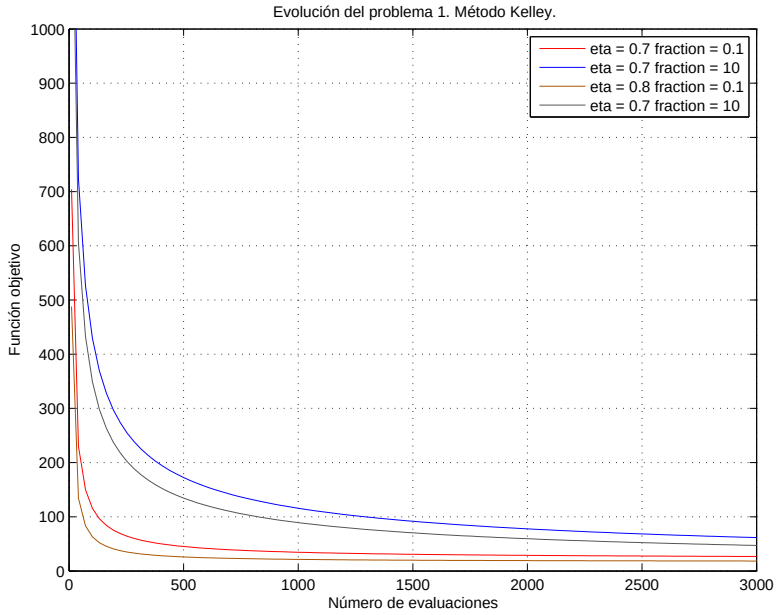


Figura 2.9: Evolución media del óptimo global en función del número de evaluaciones de la función objetivo. Método de Kelley.

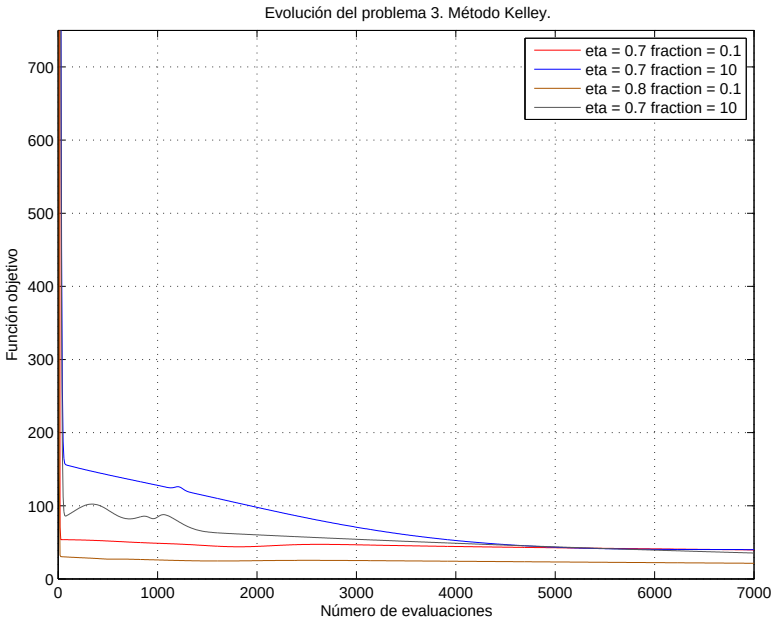


Figura 2.10: Evolución media del óptimo global en función del número de evaluaciones de la función objetivo. Método de Kelley.

2.3. Algoritmos optimización SQP

Se conocen como métodos de optimización QP a aquellos que resuelven problemas de optimización donde la función objetivo es cuadrática y las restricciones son lineales.

2.3.1. `fmincon`

`fmincon` es una función de MATLAB que encuentra el mínimo de problemas de optimización multivariables, no lineales y con ligaduras. En concreto, `fmincon` resuelve el problema definido en la ecuación (2.13).

$$\begin{array}{lll} min & f(\mathbf{x}) & \\ s.a & c(\mathbf{x}) & \leq 0 \\ & ceq(\mathbf{x}) & = 0 \\ & A \cdot \mathbf{x} & \leq \mathbf{b} \\ & Aeq \cdot \mathbf{x} & = \mathbf{beq} \\ & \mathbf{lb} \leq x_i \leq \mathbf{ub} & \end{array} \tag{2.13}$$

donde **x**, **b**, **beq**, **lb** y **ub** son vectores, A y Aeq son matrices, $c(\mathbf{x})$ y $ceq(\mathbf{x})$ son funciones que devuelven vectores y $f(\mathbf{x})$ es una función que devuelve un escalar. $f(\mathbf{x})$, $c(\mathbf{x})$ y $ceq(\mathbf{x})$ pueden ser funciones no lineales.

Definimos

$$L(\mathbf{x},\lambda,\sigma) = f(\mathbf{x}) - \lambda c(\mathbf{x}) - \sigma ceq(\mathbf{x}) \tag{2.14}$$

donde λ y σ son los multiplicadores de Lagrange. En una iteración k, un algoritmo SQP define una dirección apropiada de búsqueda $\mathbf{d}_k$ como solución al subproblema cuadrático

$$\begin{aligned} min \quad & f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T \mathbf{d} + \tfrac{1}{2}\mathbf{d}^T \nabla^2_{xx} L(\mathbf{x}_k,\lambda_k,\sigma_k)\mathbf{d}, && (2.15)\\ s.a \quad & b(\mathbf{x}_k) + \nabla b(\mathbf{x}_k)^T d \geq 0, && (2.16)\\ & c(\mathbf{x}_k) + \nabla c(\mathbf{x}_k)^T \mathbf{d} = 0. && (2.17) \end{aligned}$$

donde el Hessiano no se calcula, sino que se aproxima a partir de los valores del gradiente.

Para más información sobre esta función, puede acudir a la referencia [23]. Al no haber que ajustar parámetros, no se hace aquí el análisis de sensibilidad y el comportamiento de este algoritmo se verá directamente en la comparativa de la sección 2.4.

2.4. Comparativa

Ahora que conocemos con más detalle los algoritmos propuestos, vamos a proceder a una comparación entre ellos, de forma análoga a como hemos hecho anteriormente. Ejecutamos la optimización 30 veces y vamos a representar la evolución media, de la misma forma en que lo hicimos en los apartados anteriores. Seleccionamos la combinación de parámetros de cada uno de los algoritmos que mejores resultados obtuvo en los apartados anteriores, es decir:

- PSO: Número de partículas = 30
- SA: $\text{paso}_0 = 5$, $\text{paso}_T = 0{,}1$, $T = 10$
- GA: Nº padres = 2, Nº hijos = 5, coef. mutación = 0,1
- Kelley: $\eta = 0{,}8$, fraction = 0,1

En las Figuras 2.11-2.13 tenemos la comparación de algoritmos. La función `fmincon` de MATLAB muestra una velocidad de convergencia claramente superior en los problemas 1 y 2. Esto es debido a que la función objetivo es del tipo $\sum x_i^2$, y al tratarse `fmincon` de un algoritmo cuadrático con la segunda derivada se ajusta al comportamiento de la función objetivo. Sin embargo, en el problema 3, `fmincon` no alcanza el mínimo global y devuelve como óptimo uno de los mínimos locales de la función objetivo. En cuanto a los métodos sin cálculo de derivadas, SA se aproxima más rápidamente a la solución que GA y PSO, los cuales no presentan diferencias apreciables entre ellos. Sin embargo, con restricciones de

igualdad, SA no se comporta bien. Esto puede deberse a que este algoritmo no intenta evolucionar en una dirección concreta, sino que simplemente busca entre la vecindad de cada punto. Por su parte, el método de Kelley es el punto intermedio entre `fmincon` y los métodos sin derivadas, su velocidad de convergencia no es tan rápida como `fmincon` pero en el problema 3 no se queda tan lejos del óptimo. Sin embargo, no devuelve ningún resultado para el problema 2, por lo que su uso no presenta ventajas en ninguno de los problemas estudiados.

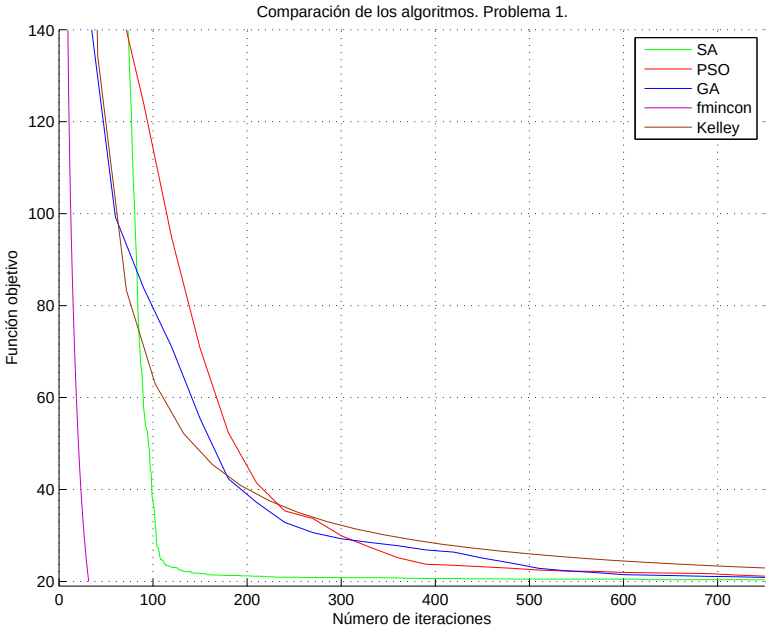


Figura 2.11: Comparación de algoritmos en el problema 1.

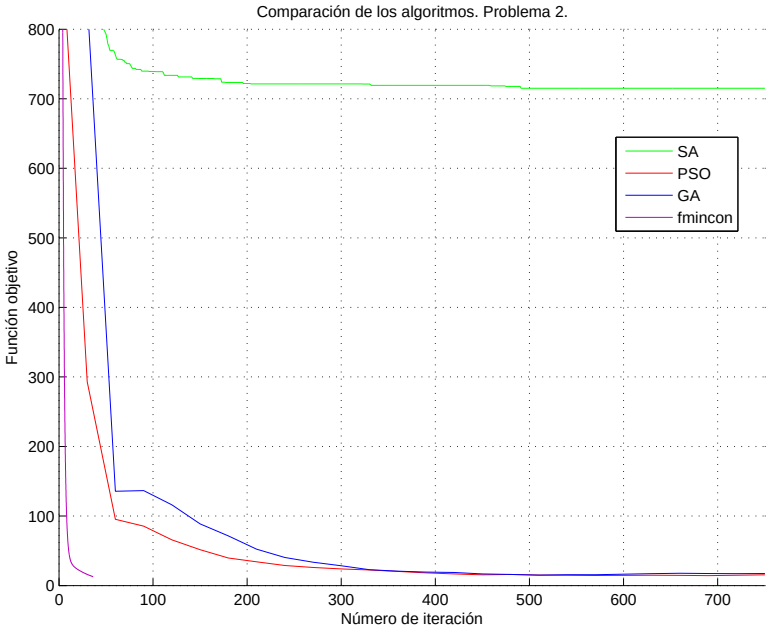


Figura 2.12: Comparación de algoritmos en el problema 2.

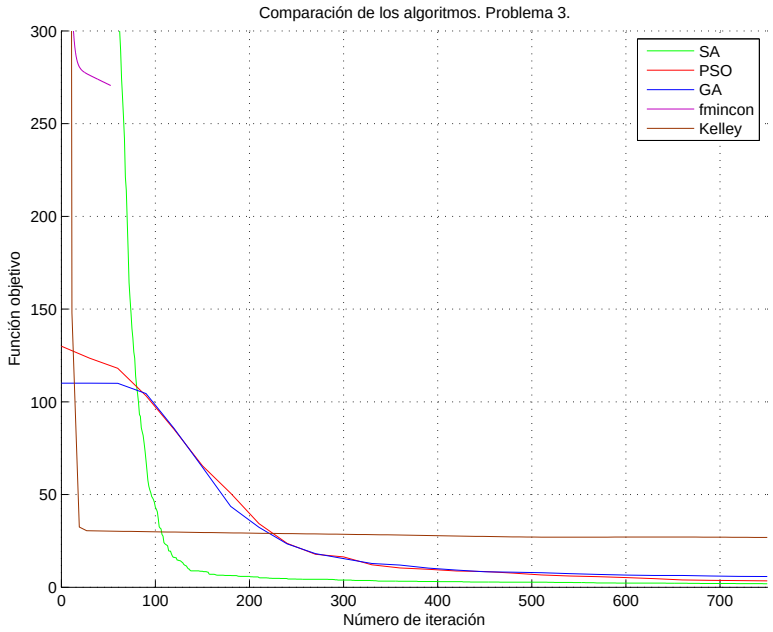


Figura 2.13: Comparación de algoritmos en el problema 3.

Como resumen, podemos dar algunas directrices para el uso de algoritmos:

- `fmincon` es el algoritmo que más rápido converge. Se recomienda su uso siempre que sea posible. Su problema es que puede obtener como solución algún mínimo local, y no conseguir llegar al mínimo global. Por tanto, se debe usar `fmincon` siempre que sea posible y, cambiando el punto de partida o ejecutando otro algoritmo, asegurarse de que el mínimo es el global.
- El algoritmo SA es el más rápido de los que no calculan derivadas. Sin embargo, en problemas con restricciones de igualdad, su comportamiento es el peor de todos, por lo que se debe evitar su uso en este tipo de problemas. En dicho caso, y nuevamente entre los algoritmos sin cálculo de derivadas, el PSO es el más adecuado.
- De acuerdo a los casos estudiados, el método de Kelley no aporta ninguna ventaja sobre los anteriores.

Capítulo 3

Optimización de trayectorias

Una vez que conocemos las características de cada método estamos en disposición de pasar a la siguiente fase: la optimización de trayectorias. Para llevar a cabo la optimización se va a hacer uso de las trayectorias tipo. Una trayectoria tipo es una trayectoria definida por un conjunto de segmentos los cuales dependen de parámetros. Algunos de estos parámetros se establecen como libres (a otros, sin embargo, se les puede fijar su valor), de forma que pueden ser usados para optimizar ciertas propiedades de la trayectoria.

Un ejemplo muy simple de una trayectoria tipo sería una trayectoria formada por un único segmento, por ejemplo, vuelo a altitud, velocidad y rumbo constantes. Se pueden dejar como parámetros libres la velocidad y la altura, mientras que se establecen como fijos el rumbo, las condiciones iniciales (excepto velocidad y altitud) y la condición de parada (por ejemplo recorrer una distancia determinada). Gracias a esta trayectoria tipo se podrían buscar los valores de velocidad y altitud que minimizan el consumo de combustible para el alcance dado.

Es recomendable que el número de segmento de la trayectoria no varíe en función de los valores de los parámetros. Por tanto, si se conocen la condición inicial y final de alguna de las variables, es recomendable usar esta información para discretizar la trayectoria de algún modo, usando dicha discretización como condición de parada de los segmentos (distancia en caso de alcance dado, peso en el caso de carga de combustible dada, etc.).

Un inconveniente que presentan las trayectorias tipo es que desde el momento en que se escogen los segmentos se fija en cierta manera de qué forma se va a volar. Por lo tanto, la elección de los segmentos debe realizarse con cierto criterio.

Como calculador de segmentos se va a usar Trajectory. Trajectory es una herramienta desarrollada por el Departamento de Ingeniería Aeroespacial de la Universidad de Sevilla capaz de calcular las variables que describen una trayectoria, y que basa su funcionamiento en el uso de trayectorias tipo (más información en el Apéndice B).

Aunque algunos de los problemas que vamos a describir a continuación ya han sido resueltos mediante el empleo del control óptimo, aquí vamos a abordarlos, como hemos expuesto, mediante trayectorias tipo. Si bien ésto implica realizar una

aproximación al problema real, con la problemática asociada a las aproximaciones (introducción de nuevas soluciones o eliminación de las existentes, precisión de los resultados...), obtenemos una clara ventaja con respecto a las soluciones obtenidas mediante control óptimo: éstas últimas presentan una evolución continua de las variables del problema (velocidad, altura, rumbo...), lo cual desde los puntos de vista operativo y técnico no es factible en la actualidad, mientras que con el uso de trayectorias tipo podemos adecuarnos a los condicionantes operativos actuales.

Vamos a plantear tres problemas, dos de los cuales ya han sido abordados mediante el uso de control óptimo, lo que nos será muy útil de cara a comparar ambos planteamientos. Estos dos problemas son la optimización de un crucero de máximo alcance a altura constante y la optimización de un crucero a altura constante con tiempo dado, los cuales han sido resueltos mediante la aplicación del control óptimo por D. Rivas y A. Valenzuela [24], y A. Franco, D. Rivas y A. Valenzuela [3] respectivamente. El tercer problema será la optimización de un crucero libre, en el que se parte del punto A una altura $h_A = 10000$ ft a una velocidad CAS de 250 kt, y se llega al punto B a una altura $h_B = 10000$ ft y velocidad CAS de 250 kt. Entre ambos puntos habrá una distancia que variaremos como parámetro para ver distintos comportamientos. Las limitaciones de altitud y velocidad impuestas a los puntos A y B vienen dadas por el control de tráfico que habitualmente se hace: no se permite volar a más de 250 kt por debajo de 10000 ft.

Vamos a usar, en todos los problemas un Boeing 767-300ER con un modelo aerodinámico definido por una polar con coeficientes dependientes del número de Mach. Usaremos dos motores CF6-80C2, representados mediante un modelo que considera el empuje dependiente del número de Mach y de la altura, y el consumo específico dependiente del número de Mach. Tanto el modelo de aerodinámico como el modelo propulsivo de la aeronave están desarrollados en el Apéndice C. En todos los problemas vamos a usar la atmósfera estándar ISA. Igualmente, sólo consideraremos vuelo simétrico, por lo que resolveremos problemas contenidos en el plano vertical.

3.1. Crucero de máximo alcance a altura constante

Supongamos un avión con pesos inicial W_i y final W_f dados, a una altura h_A constante, con ángulo de balance μ nulo. Queremos optimizar el alcance de dicho avión variando el número de *Mach* a lo largo del crucero. Este problema ha sido resuelto mediante control óptimo, obteniendo una variación continua del número de Mach, por D. Rivas y A. Valenzuela en la referencia [24].

Para modelar el problema discretizamos el peso W, de forma que tendremos segmentos en los que se vuela a M constante y segmentos de transición (aceleración o desaceleración) entre ellos. Por tanto, describiendo nuestra trayectoria tipo, tenemos un segmento a Mach, altura y rumbo constante como restricciones, y alcanzar el peso final del segmento como condición de parada del segmento.

A continuación, tenemos un segmento a altura, rumbo y rating de motor (*Idle* o máximo continuo) constante. Este segmento finalizará cuando se alcance el Mach del siguiente segmento. A continuación volveríamos a usar el primer tipo de segmento pero como peso inicial tomamos el peso final del anterior.

En la Figura 3.1 tenemos dos ejemplos de discretización en segmentos. El *crucero ejemplo 1* se compone de un primer segmento entre $W = 1{,}7 \cdot 10^6$ N y $W = 1{,}425 \cdot 10^6$ N a $M = 0{,}75$, un segmento de deceleración a rating de motor *Flight Idle, FI*, y un segmento entre el peso final del segmento anterior y $W = 1{,}15 \cdot 10^6$ N a $M = 0{,}72$. El *crucero ejemplo 2* se compone de un primer segmento entre el peso final del segmento anterior y $W = 1{,}4 \cdot 10^6$ N a $M = 0{,}735$, un segmento de aceleración a rating de motor máximo de crucero, *MCRZ*, y un segmento entre $W = 1{,}4 \cdot 10^6$ N y $W = 1{,}15 \cdot 10^6$ N a $M = 0{,}74$.

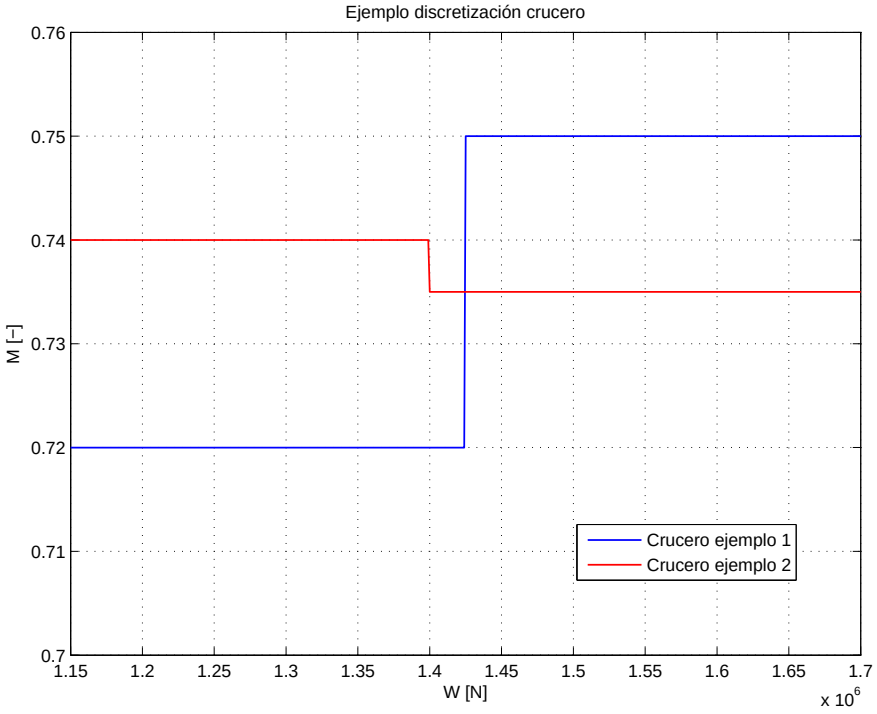


Figura 3.1: Ejemplo de discretización de cruceros.

Una vez explicada la discretización que vamos a realizar y el enunciado del problema, lo definimos formalmente como:

$$\begin{aligned} max \quad & Alcance(M_1, M_2, \ldots, M_q) \\ s.a \quad & 0{,}65 \leq M_j \leq 0{,}82, \qquad j = 1, \ldots, q \\ & h_A = \text{cte} \\ & W_i = 1{,}7\,\text{MN} \\ & W_f = 1{,}15\,\text{MN} \end{aligned} \tag{3.1}$$

es decir, vamos a maximizar el alcance variando para ello el Mach de vuelo en los distintos segmentos sin exceder $M = 0{,}82$ para evitar que el motor no tenga

potencia suficiente para acelerar en los tramos entre segmentos y sin ser menor a $M = 0{,}65$ para estar lejos de la entrada en pérdida. Estando establecido el enunciado del problema, vamos a representar la función objetivo (Figura 3.2) para el modelo de avión Boeing 767-300ER, para un peso inicial $W_i = 1{,}7 \cdot 10^6\,\mathrm{N}$ y un peso final $W = 1{,}15 \cdot 10^6\,\mathrm{N}$, dos segmentos a M constante, con $W = 1{,}4 \cdot 10^6\,\mathrm{N}$ como peso final del primer segmento y con altura de crucero $h_A = 10000\,\mathrm{m}$. La función objetivo resultante es suave lo que nos hace pensar que `fmincon` va a ser el mejor método de optimización, aunque al no haber restricciones de igualdad, el comportamiento del algoritmo SA puede ser equiparable.

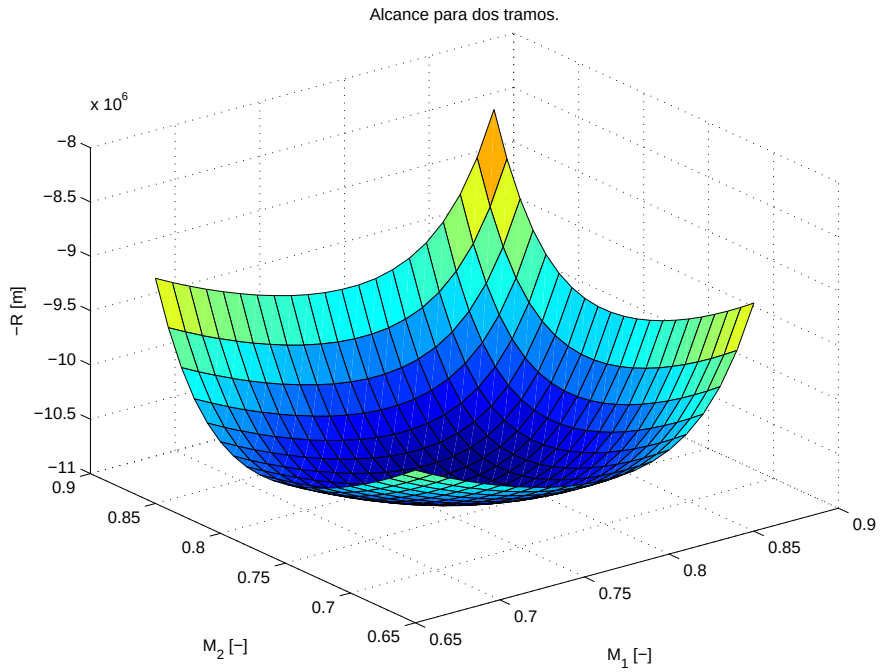


Figura 3.2: Función objetivo del crucero a altura constante $h_A = 10000\,\mathrm{m}$.

Generamos una discretización de 19 segmentos, de los cuales 10 son a Mach constante[1] y finalizan cuando se alcanza el peso final del segmento (los pesos finales de los segmentos son equidistantes entre sí), mientras que los 9 restantes son segmentos de aceleración o deceleración entre los anteriores. Este problema lo resolvemos con `fmincon` y con SA, proporcionando ambos los mismos resultados. Estos resultados a diferentes alturas (9.000, 10.000 y 11.000 m) pueden compararse en la Figura 3.3 y en la Tabla 3.1 con la solución obtenida mediante control óptimo del problema (extraída de [24]). Realmente tenemos que puntualizar que no estamos resolviendo exactamente el mismo problema. En la solución obtenida mediante control óptimo se impuso además que las condiciones inicial

[1] El uso de 10 segmentos a Mach constante equivale a aproximadamente un cambio de M a la hora en un crucero de estas características. Por tanto, y a diferencia de los resultados obtenidos mediante el control óptimo donde el Mach varía de forma continua, este resultado se podría llevar a cabo en un vuelo real.

y final estuvieran sobre el arco singular, mientras que mediante trayectorias tipo éstas pueden tomar los valores que el optimizador decida.

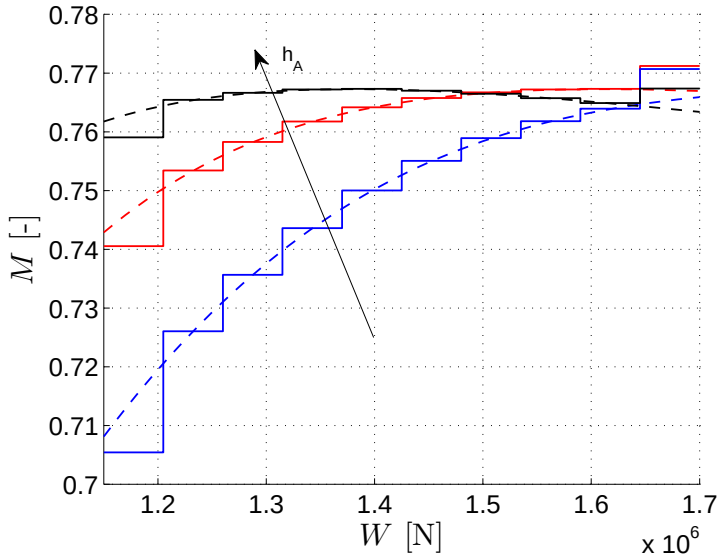


Figura 3.3: Número de *Mach* en función del peso para una discretización con 10 segmentos a Mach constante, para alturas $h_A = 9000$, 10000 y 11000 m.

Altura [m]	9000	10000	11000
Alcance control óptimo [km]	11095	11144	10873
Alcance 10 segmentos [km]	11091	11138	10866

Tabla 3.1: Comparación de los resultados obtenidos mediante control óptimo y con 10 segmentos a Mach constante.

Los resultados obtenidos mediante segmentos frente a los obtenidos mediante control óptimo son muy similares, pues como vemos en la Figura 3.3 las líneas son casi coincidentes y, en términos tanto relativos como absolutos, la diferencia en alcance entre ambos es pequeña. Sin embargo, en la solución mediante segmentos los tramos con mayor y menor peso son a mayor y menor número de Mach, respectivamente, que la solución mediante control óptimo.

En la Figura 3.4 se puede ver el efecto de la discretización sobre los resultados: para ello se ha discretizado con 4, 8, 16 y 32 segmentos a Mach constante, para ver cómo las condiciones iniciales y finales toman valores extremos a medida que aumenta el número de segmentos. Este fenómeno recuerda al *bang-singular-bang*, que son soluciones donde las variables de control toman valores extremos al principio y al final del dominio y que aparecen frecuentemente en problemas de control en los que no se imponen condiciones de contorno. Ésto no sucede en la solución del problema mediante control óptimo porque se impone que los valores extremos estén sobre el arco singular.

Para explicar lo que sucede vamos a recurrir a conceptos energéticos, en concreto a la energía cinética (si bien el razonamiento sería válido para cruceros con altura variable sin más que considerar la energía mecánica en lugar de la cinética). La solución mediante trayectorias tipo predice un Mach superior al analítico al inicio del crucero e inferior al final de éste. La solución mediante trayectorias tipo inicia el crucero con mayor energía cinética (puesto que no está penalizada en la función objetivo la aceleración previa al inicio del crucero) y al final aprovecha la energía cinética para extender el rango volando a un régimen cercano a Fligh Idle. Para ver el efecto sobre el alcance de estos hechos se ha elaborado la Tabla 3.2, en la que se muestra el alcance para distintos números de segmentos a Mach constante para una altura de crucero $h_A = 10000$ m. Conforme aumentamos el número de segmentos, la solución mediante segmentos se aproxima en su zona central (arco singular) a la solución mediante control óptimo, y el alcance predicho mediante segmentos crece.

Segmentos	4	8	10	12	32
Alcance [km]	11137	11138	11138	11140	11141

Tabla 3.2: Influencia en el alcance del número de segmentos a Mach constante para un crucero a $h_A = 10000$ m.

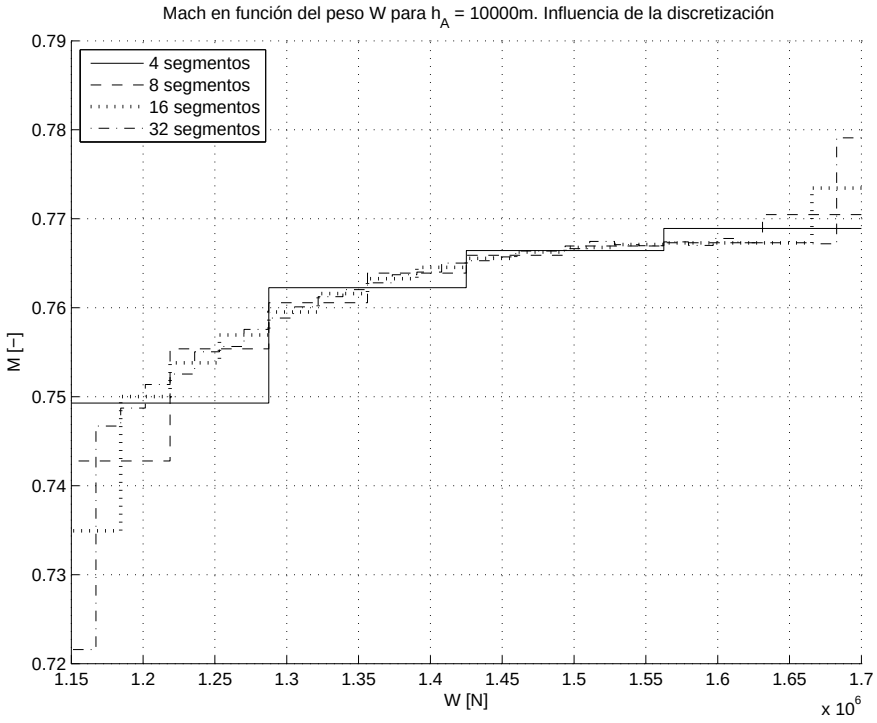


Figura 3.4: Efecto del número de segmentos a Mach constante en el crucero óptimo a altura consntate $h_A = 10000$ m.

Según lo expuesto en el capítulo anterior, `fmincon` puede quedarse atrapado en mínimos locales mientras que SA, pese a ser un poco más lento, evita este problema. Lo que se ha hecho por tanto es ejecutar tanto `fmincon` como SA cinco veces con condiciones iniciales aleatorias (en concreto, las condiciones iniciales seleccionadas son uniformes y contenidas dentro de la región de admisibilidad del número de Mach) persiguiendo un doble objetivo: por un lado se ha querido garantizar que no existan mínimos locales o que si existieran `fmincon` no se quede atrapado en ellos, lo que conseguimos mediante comparación con los resultados de SA. Por otro lado, se ha perseguido extraer una evolución media (de forma análoga a lo que se hizo en el capítulo anterior) para comparar la velocidad de convergencia de ambos algoritmos. La comparación, en la Figura 3.5, muestra que tienen un comportamiento casi idéntico, y por lo tanto se pueden usar indistintamente. El algoritmo SA supera las expectativas que sobre él se tenían depositadas. Una posible explicación a su buen comportamiento puede basarse en que el dominio del número de Mach es pequeño, por lo que algoritmo SA recorre este dominio de forma eficaz e iguala a su velocidad a `fmincon`.

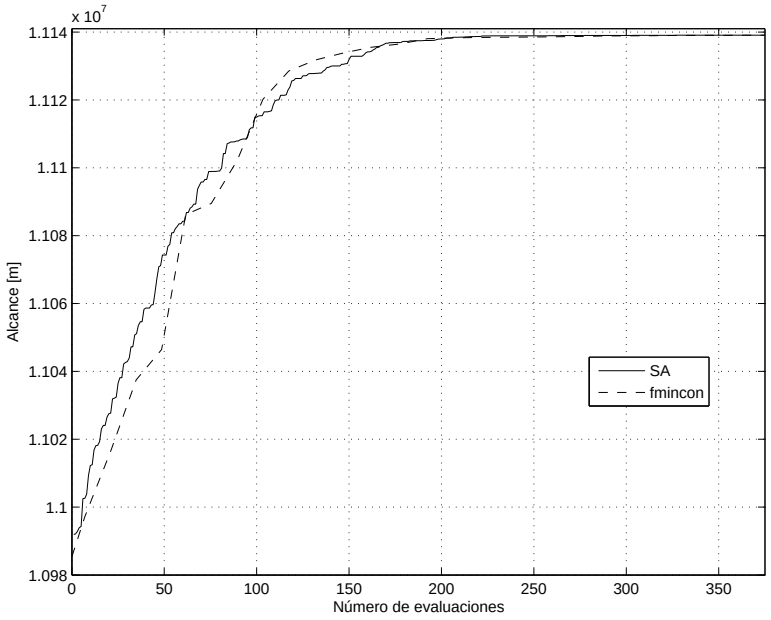


Figura 3.5: Evolución media de los algoritmos SA y `fmincon`.

3.2. Crucero óptimo a altura constante con tiempo dado

Queremos minimizar el consumo de combustible para un crucero con alcance, altura, tiempo y peso inicial dados. De forma simplificada, podemos decir que añadimos sobre el problema anterior la restricción del tiempo de crucero y pa-

samos a minimizar el consumo de combustible, ya que el alcance pasa a estar dado. Por supuesto, al igual que en el problema anterior, obtendremos los resultados para diferentes alturas. A. Franco, D. Rivas y A. Valenzuela han resuelto este problema mediante control óptimo en [3].

Por tanto, vamos a minimizar el consumo de combustible de un crucero con alcance $x_f = 10000\,\text{km}$, con un peso inicial $W_i = 1600\,\text{kN}$ a una altura h_A dada, en un tiempo t_f dado, con la limitación $0{,}65 < M < 0{,}82$. Matemáticamente:

$$\begin{aligned} min \quad & Consumo(M_1, M_2, \ldots, M_q) \\ s.a \quad & 0{,}65 \leq M_j \leq 0{,}82, \qquad j = 1, \ldots, q \\ & h_A = \text{constante} \\ & W_i = 1{,}6\,\text{MN} \\ & tiempo = t_f \\ & alcance = 10000\,\text{km} \end{aligned} \tag{3.2}$$

Ya que tenemos restricciones de igualdad, el algoritmo a usar es `fmincon`. En este caso la discretización se realiza en la distancia recorrida, ya que conocemos la distancia final, y usaremos una trayectoria tipo muy similar al problema anterior. Usaremos dos tipos de segmentos: el primero de ellos será a Mach constante y finalizará cuando se llegue al alcance predeterminado para ese segmento. El segundo tipo de segmento será un segmento de aceleración o deceleración, y finalizará al alcanzar el Mach del siguiente segmento. Las condiciones finales de este segundo tipo de segmento serán las iniciales del primer tipo que se recorra a continuación. La pareja segmento de transición más el siguiente segmento a Mach constante tendrán un alcance igual al alcance total del crucero entre el número de segmentos a Mach constante. Vamos a obtener resultados para $h_A = 9000\,m$ y $h_A = 11000\,m$. En cuanto al tiempo, vamos a obtener en primer lugar el tiempo para el que el consumo de combustible es mínimo, es decir, vamos a resolver un subproblema en el que eliminamos la limitación de tiempo. Variando el tiempo de crucero en el entorno de este *tiempo óptimo* obtendremos las tendencias que sigue el consumo de combustible para este problema.

Los resultados se muestran en la Figura 3.6. En ellos, se representa el incremento de combustible sobre el óptimo (es decir, crucero con tiempo libre) frente al incremento de tiempo de crucero, para las alturas de vuelo $h_A = 9000, 11000\,\text{m}$. En concreto, se muestra el resultado mediante control óptimo (extraído de [3]), el resultado de realizar el crucero a Mach constante, y los resultados para 5 y 2 segmentos a Mach constante, con los respectivos segmentos de aceleración o deceleración a rating de motor MCRZ o Idle entre ellos. Lo más destacable de los resultados es:

- La poca variación de los resultados a las diferentes alturas para discretización con 2 segmentos a Mach constante, comparando los resultados a la discretización con 5 segmentos o a las soluciones mediante control óptimo.

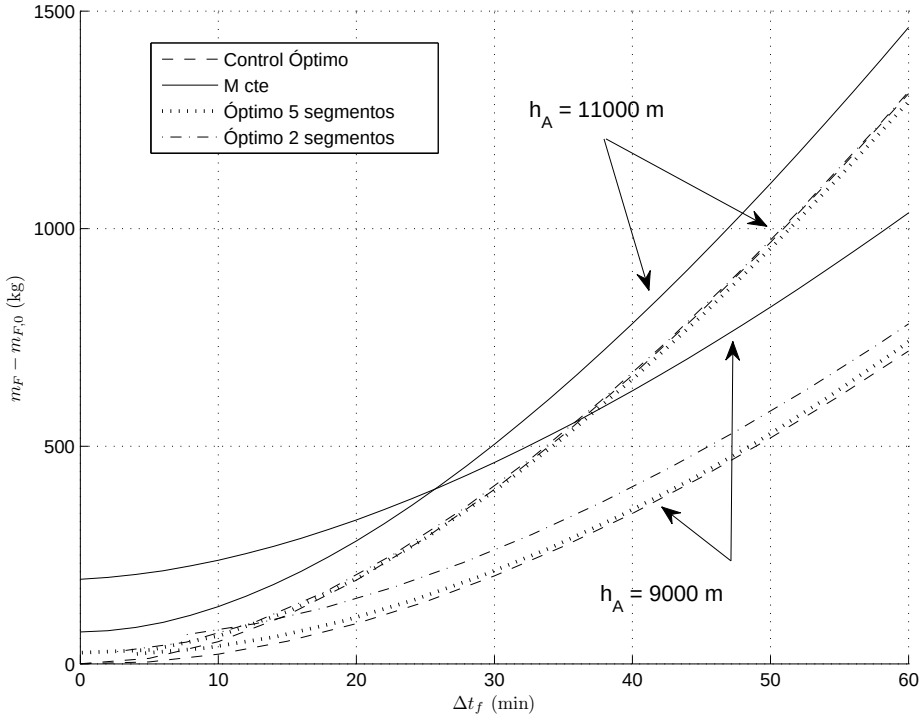


Figura 3.6: Consumo de combustible frente a retraso: resultado mediante control óptimo, crucero a Mach constante, y resultados con 5 y 2 segmentos.

Es decir, si en lugar de volar todo el crucero a Mach constante realizamos un cambio de Mach durante éste, el consumo de combustible se aproxima sensiblemente al óptimo. Además, computacionalmente es muy rápido solucionar el problema para 2 segmentos, por lo que se podría usar como solución inicial para cruceros con más segmentos.

- Para una variación del tiempo de óptimo de 30 minutos (aprox. 4% del tiempo de vuelo), el consumo de combustible se incrementa en aproximadamente 200 kg, lo que equivale a un 0.5% del consumo total.

- La dependencia del consumo de combustible decrece con la altura de vuelo h_A.

Por último, vamos a comparar el consumo de combustible entre el resultado mediante control óptimo y una discretización con 10 segmentos a Mach constante. Como vemos en la Tabla 3.3, resultados numéricos y analíticos son bastante cercanos. A diferencia de lo que ocurría en el problema anterior, en este problema sí obtenemos masas finales mayores mediante el uso de segmentos que mediante el uso de control óptimo. Para cuantificar el efecto que produce sobre el alcance el hecho de que tanto el Mach inicial como el final tiendan a los extremos, se resolvió un problema mediante segmentos en el que se imponía que tanto Mach

inicial como final debían estar sobre el arco singular de la curva, viniendo tal arco definido por la solución mediante control óptimo. La máxima desviación que se obtuvo fue de 44 kg, o lo que es lo mismo un $0,1\,\%$ sobre el total de combustible consumido durante el crucero.

h_A(m)	t_f(s)	m_F (kg) segmentos	m_F (kg) control óptimo
9000	46443	48853	48855
9000	44511	48637	48625
9000	42631	49066	49024
11000	45067	48898	48892
11000	44226	48827	48802
11000	43275	49063	49005

Tabla 3.3: Masa consumida m_F para distintos tiempos y alturas, para una discretización con 10 segmentos a Mach constante.

3.3. Crucero libre

El último problema que vamos a abordar es un crucero libre, en el que las únicas limitaciones serán el punto de partida y el punto de finalización. La posición de partida (punto A) de la aeronave es $h_A = 10000\,\text{ft}$ y $VCAS_A = 250\,\text{kt}$, mientras que se pretende que la aeronave llegue a (punto B) $h_B = 10000\,\text{ft}$ y $VCAS_B = 250\,\text{kt}$. La distancia entre los puntos A y B es de 1000 km. No se establece ninguna limitación sobre el tiempo o altura de vuelo, motivo por el que enunciamos el problema como *Crucero libre*. Los puntos A y B no se definen por casualidad. En la actualidad, por debajo de 10000 ft no se permite volar a más de 250 kt CAS, por lo que necesariamente cualquier vuelo real debería pasar por los puntos A y B. Además, imponer estas condiciones de contorno hace que el sistema no acumule más energía al inicio del crucero, como sucedía en los problemas anteriores y que hacía que tanto el Mach inicial como el final se desviaran del comportamiento normal. El rumbo se considerará constante y resolveremos por tanto el problema en el plano vertical. Por tratarse de un problema sujeto a restricciones de igualdad, el optimizador más indicado a usar es `fmincon`.

La primera discusión que debemos hacer es la referente a las trayectorias tipo. En un primer momento se discretizó el problema de forma similar a los anteriores. Se pensó en segmentos de una distancia determinada, a Mach y ángulo de subida constantes, con segmentos de aceleración/deceleración a altura constante entre ellos, motivado principalmente por la forma de control que se tiene con el piloto automático en una aeronave. Generalmente, como entradas de control al piloto automático se introduce un parámetro referente a la velocidad (Mach o velocidad CAS), y otro referente a la altura (alcanzar una altura determinada o mantener una velocidad vertical). Por tanto, los resultados que se obtuvieran con esta discretización serían coherentes con la operaciones actuales.

Sin embargo, durante la resolución del problema, tal discretización presentó los siguientes problemas:

- La longitud de los segmentos. Si el segmento era menor que unos 50 km podría suceder que el tramo de aceleración/desaceleración ocupase más de esos 50 km, lo que quiere decir que el salto de Mach entre segmentos adyacentes era superior al que podían proporcionar los motores. Habría que limitar el salto de Mach entre segmentos adyacentes.

- El ángulo de subida debía limitarse a valores bajos, en torno a $\pm 4^{o}$. Esta limitación dependía de la distancia de la discretización, para segmentos largos hay que reducir el ángulo máximo y viceversa para los cortos.

- Al limitar el salto de Mach entre segmentos adyacentes se corre el riesgo de estar perdiendo soluciones.

- Había que limitar la altura máxima alcanzada durante el recorrido a 11.000 m. Por encima de esta altura el consumo del motor podía resultar excesivo, con lo que el avión se quedaría sin masa para realizar el resto del crucero.

A la vista de las limitaciones que presentaba el uso de esas trayectorias tipo, se desechó y se modeló mediante otra ligeramente diferente. Como en el caso anterior, el rumbo se considera constante y sólo se resuelve el problema en el plano vertical. Se pasó entonces a usar segmentos en los que se volaba a régimen de motor y ángulo de subida constantes, sin segmentos intermedios. El segmento se detiene al alcanzar la distancia de fin de segmento o si se vuela a menos de una velocidad cercana a la pérdida (VTAS = 50 m/s). En el caso de alcanzar dicha velocidad el crucero queda detenido en ese punto, lo que obliga a introducir el alcance final como una ligadura. Este control de «velocidad mínima» se introduce porque al trabajar directamente con el motor podría suceder que el avión intentase volar fuera de su zona operativa. Las condiciones iniciales de cada segmento son las finales del anterior, a excepción del primer segmento donde las condiciones iniciales son dadas por el enunciado del problema. Observando el comportamiento para obtener los primeros resultados, se decidió usar un método iterativo, en el que en un primer momento se resolvía el problema con pocos segmentos. A continuación, se duplicaba el número de segmentos y la solución anterior se usaba como condición inicial de esta iteración. Este método iterativo ayudaba a mejorar la velocidad de la solución, ya que si se atacaba directamente el problema con un número de segmentos bastante algo, `fmincon` quedaba atrapado en soluciones no factibles.

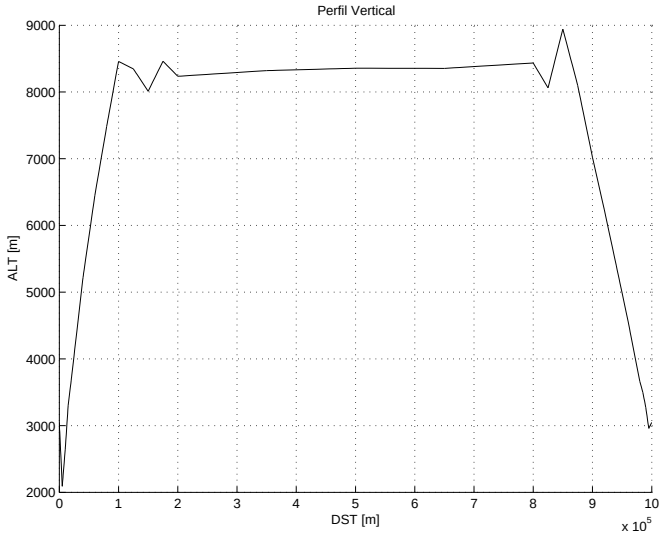
Se introducen dos limitaciones más sobre el modelo. La primera consiste en limitar los regímenes máximo y mínimo de motor, que tomas valores de 0,95 y 0,01, respectivamente, siendo el primero de ellos el valor típico de máximo continuo y el segundo de *Flight Idle*. La segunda limitación es el ángulo de subida, al que se obliga a estar entre $\pm 11^{o}$. Este valor intenta representar el máximo/mínimo

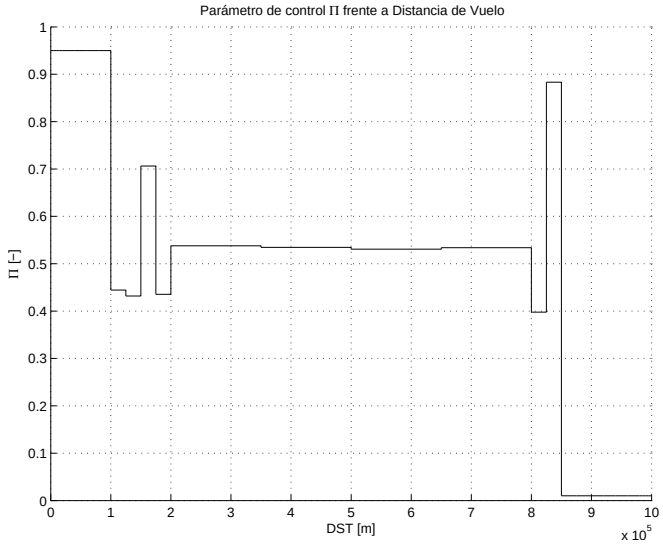
ángulo de subida que sería capaz de seguir de manera confortable una aeronave de estas características, a falta de valores más precisos. Como resumen, el enunciado matemático resulta ser:

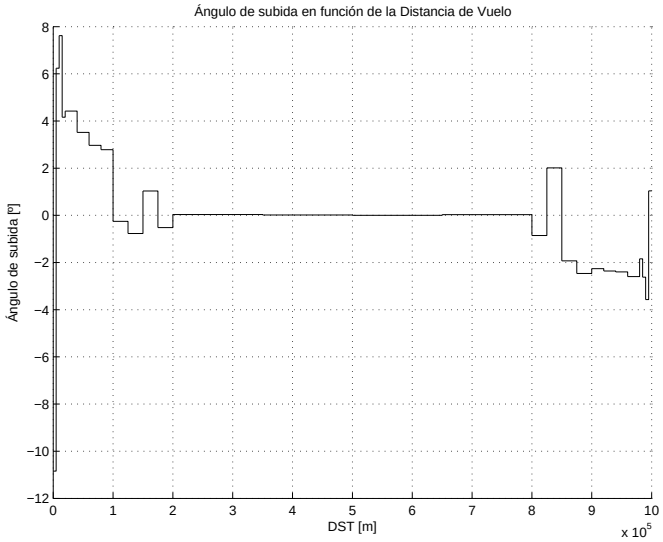
$$
\begin{array}{lcll}
min & Consumo(\pi_1, PTH_1, \pi_2, PTH_2, \ldots, \pi_q, PTH_q), & & (3.3)\\
s.a & 0{,}01 \leq \pi_j \leq 0{,}95, & j = 1, \ldots, q, & \\
 & -11^{\mathrm{o}} \leq PTH_j \leq 11^{\mathrm{o}}, & j = 1, \ldots, q, & \\
 & W_i = 1{,}7658\,\mathrm{kN}, & & \\
 & h_A = h_B = 10000\,\mathrm{ft}, & & \\
 & VCAS_A = VCAS_B = 250\,\mathrm{kt}, & & \\
 & \mathrm{Distancia}\, A \mapsto B = 1000, 5000\,\mathrm{km}. & &
\end{array}
$$

Se ha tomado una mayor densidad de segmentos en los extremos y una denisdad baja en la zona central, donde se espera que aparezca un crucero. Las longitudes en km de los segmentos para alcance 1000 y 5000 km son:

Nº segmento	1000 km	5000 km
1-4	5	5
5-8	20	20
9-12	25	25
13-16	150	1150
17-20	25	25
21-24	20	20
25-28	5	5


Perfil Vertical
ALT [m]
DST [m]
x 10^5


Parámetro de control Π frente a Distancia de Vuelo
Π [–]
DST [m]
x 10^5


Ángulo de subida en función de la Distancia de Vuelo
Ángulo de subida [º]
DST [m]
x 10^5

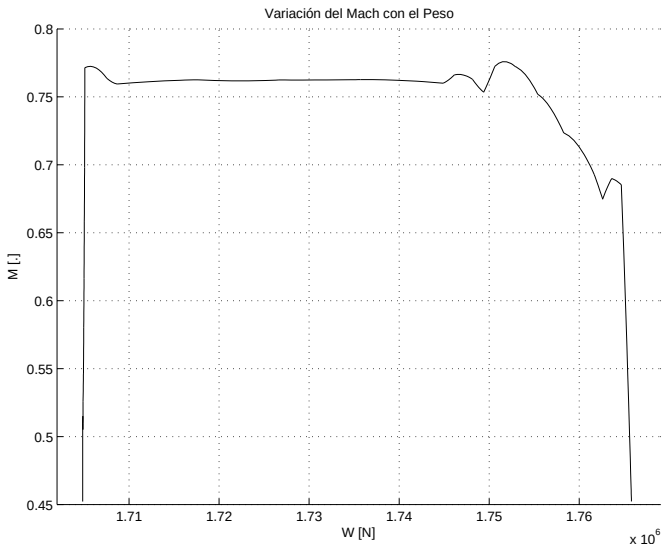
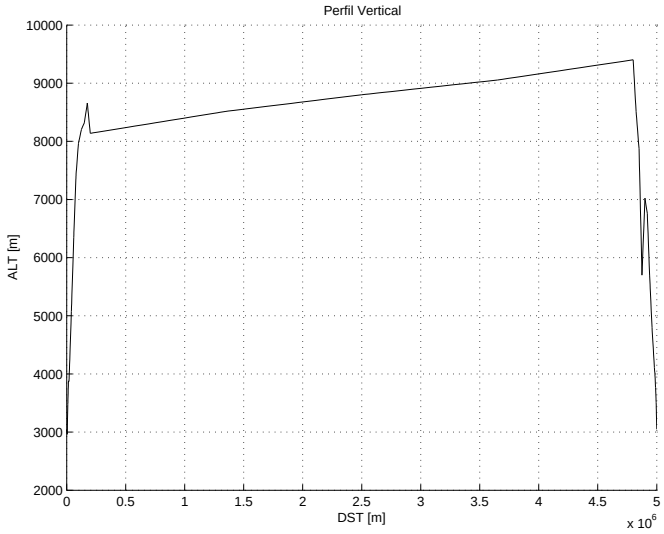
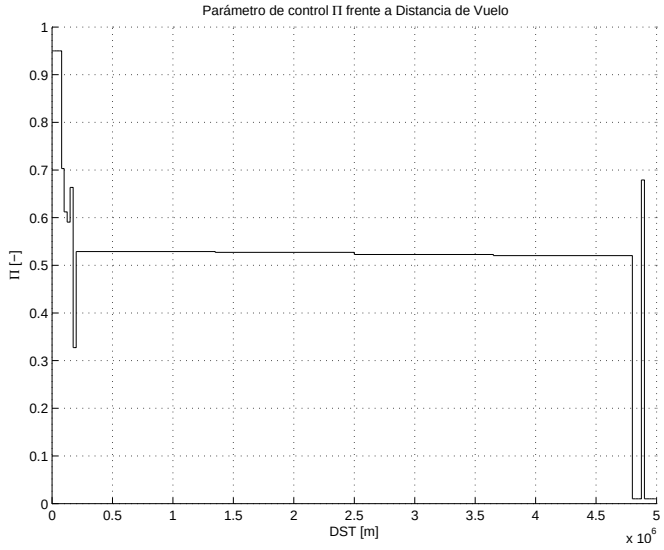
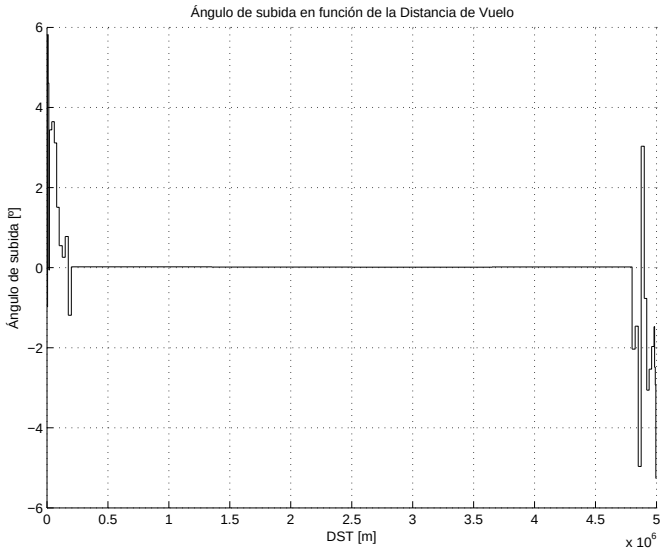


Figura 3.7: Variables significativas para alcance 1000 km y 28 segmentos.


Perfil Vertical
ALT [m]
DST [m]
x 10^6


Parámetro de control Π frente a Distancia de Vuelo
Π [–]
DST [m]
x 10^6


Ángulo de subida en función de la Distancia de Vuelo
Ángulo de subida [º]
DST [m]
x 10^6

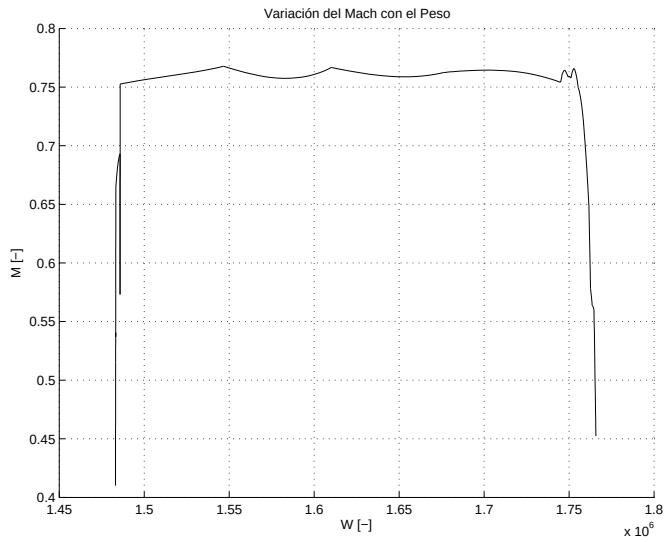


Figura 3.8: Variables significativas para alcance 5000 km y 28 segmentos.

Como podemos ver en las Figuras 3.7 y 3.8 los resultados tienen la apariencia esperada. Se realiza un ascenso hasta llegar a una altura de vuelo óptima, tras ésto se realiza un *Cruise Climb* (aunque en algunas gráficas pueda parecer que el ángulo de subida toma como valor 0^o durante una parte del crucero, su valor real es de aprox. 0.5^o. En las figuras para alcance 5000 km puede apreciarse mejor) y por último se desciende con el motor en Idle. Sin embargo, hay algunos comentarios que debemos hacer:

- Al inicio del crucero, la solución óptima para 1000 km propone realizar un descenso. Vemos cómo al picar el avión se consigue acelerar con mayor rapidez, y así conseguir que la subida sea a mayor velocidad. No es algo que nos deba sorprender ya que en muchos casos se considera la subida con mínimo consumo de combustible como la más rápida. A 5000 km también se observa la caída (el ángulo de subida es negativo al principio) aunque la caída no es tan pronunciada.

- No se aprecia una variación clara del número de Mach conforme disminuye el peso porque al ser la altitud del vuelo creciente el número de Mach se mantiene aproximadamente constante. Por tanto, podemos decir que el cruise climb se hace prácticamente a Mach constante. En el crucero de 5000 km el Mach oscila ligeramente a lo largo de los segmentos centrales.

- Se produce una oscilación en todas las variables del problema en el tramo donde termina la subida, así como al inicio del descenso o durante el propio descenso. Esta oscilación crece conforme aumenta el número de segmentos de la discretización, como vemos en la Figura 3.9, donde se representa el perfil vertical del crucero de 1000 km para 14, 28 y 56 segmentos. Estas

oscilaciones aparecen cuando Π pasa de su valor extremo al «arco singular» y viceversa, cuando acabandona tal «arco singular» y vuelve a tomar un valor extremo. Mientras Π está en valores extremos o en el arco, parece que γ también sigue una evolución suave (excepto en el contorno). Por tanto, las oscilaciones aparecen en las transiciones.

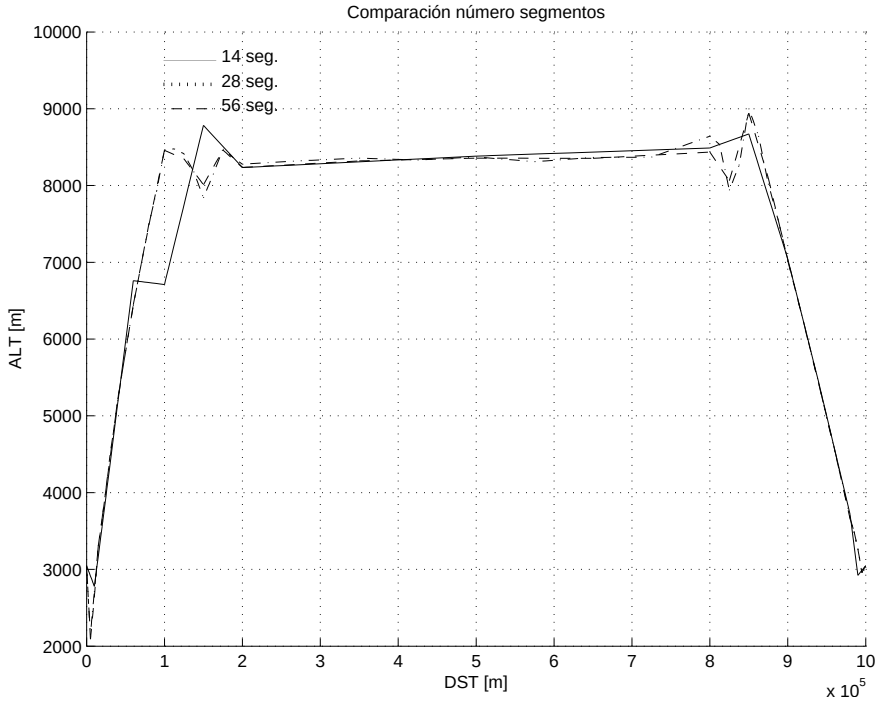


Figura 3.9: Perfil vertical del crucero de 1000 km para 14, 28 y 56 segmentos.

Capítulo 4

Conclusiones

Si nos remontamos ahora al capítulo de Introducción, concretamente a la exposición de objetivos podemos leer:

> En vista de lo anterior, los objetivos del presente proyecto son dos:
>
> 1. La implementación y evaluación de algoritmos para la optimización multivariable sujeta a restricciones.
> 2. La aplicación de dichos algoritmos en la optimización de trayectorias modeladas mediante trayectorias tipo y su comparación con trayectorias óptimas obtenidas mediante otros métodos.

Tras todo lo expuesto a lo largo de los capítulos de este proyecto, podemos concluir que los objetivos marcados inicialmente han sido cubiertos. Se han implementado y evaluado tres algoritmos para optimización multivariable, se han evaluado dos algoritmos más y se ha expuesto un método para extender los algoritmos de optimización libre a optimización sujeta a restricciones sin necesidad de emplear el cálculo de derivadas de las restricciones. Con los conocimientos adquiridos se han aplicado algunos algoritmos a la optimización de trayectorias. Dos de estas trayectorias óptimas se han comparado con las obtenidas mediante otros métodos, y una tercera ha sido estudiada de forma libre.

Del trabajo, la experiencia adquirida y los resultados obtenidos durante el desarrollo de este proyecto podemos extraer las siguientes conclusiones:

- Conclusiones sobre uso de algoritmos. En general, el algoritmo `fmincon` saca a relucir las horas de trabajo que Mathworks ha invertido en él y es más rápido que los algoritmos sin cálculo de derivadas que se han implementado aquí. Aun así, el algoritmo SA puede rivalizar en rendimiento con `fmincon` bajo determinadas circunstancias, tal como apuntamos al final del Capítulo 2.

- Ventajas del uso de trayectorias tipo. Aunque los problemas de optimización pueden enfocarse desde otras técnicas, como por ejemplo el control óptimo

que ha sido nombrada en el desarrollo de este proyecto, el uso de trayectorias tipo presenta algunas ventajas. En el modelo de tráfico aéreo actual no es posible realizar un crucero con una variación continua de las leyes de pilotaje, ya que atenta contra los principios actuales del ordenamiento del tráfico aéreo. Mediante las trayectorias tipo, puede condicionarse la solución y hacer que ésta se mueva dentro de los límites operativos actuales, si así se desea. Pero no tiene por qué limitarse a soluciones de este tipo, pues como hemos visto en el crucero libre también podemos resolver problemas donde sí existe una variación continua de las variables básicas de vuelo.

- Similitud de los resultados. Los resultados obtenidos son, por lo general, muy similares en comparación a los obtenidos por otras técnicas. El fenómeno más destacado se ha producido cuando Mach inicial y final tendía a acercarse a los extremos al aumentar el número de segmentos.

- Aproximación al problema. La optimización mediante trayectorias tipo permite obtener una primera aproximación al problema óptimo, ofreciendo pistas sobre el resultado que se puede obtener mediante otras técnicas.

- Importancia de la trayectoria tipo. La elección de los segmentos (tipo y longitud) es la estapa más crítica del problema. Una buena elección puede acelerar la convergencia de la solución, pero una mala elección puede hacer que el algoritmo optimizador no sea capaz de proporcionar una solución válida.

- Estrecho margen de mejora. El margen de mejora y la importancia relativa de los beneficios que se podrían obtener aplicando estas técnicas son bastante reducidos, habiendo otra serie de factores tales como meteorología o gestión de colas, donde podrían obtener mejoras mayores. Sin embargo, en un sector económico que ha evolucionado tanto en los últimos años, pasando de ser un transporte de élite a uno de masas donde la competencia entre compañias es muy alta, y con los nuevos retos que se están planteando como la reducción de emisiones a la atmósfera, cualquier contribución al incremento de la ecoeficiencia debe ser aceptado, y con más motivo aún si este incremento conlleva pocos o ningún cambio importante.

Apéndice A

Implementación en MATLAB

La implementación de los algoritmos nombrados durante el presente documento se ha llevado a cabo en MATLAB. Los motivos son:

- Es un lenguaje simple, fácil, con un período de adaptación corto y una curva de aprendizaje óptima. En una comparación frente a C o C++ el rendimiento es cercano. Además, dentro de MATLAB vienen incorporadas multitud de librerías que permiten hacer un gran número de operaciones matemáticas avanzadas.
- Es, de facto, uno de los estándares en la investigación.
- Porque Trajectory está también programado en MATLAB, lo que simplifica la interacción entre ambos.

Los métodos εPSO, εSA y εGA se han implementado en una función de MATLAB llamada `optimizador_global2`. Tal función busca el mínimo de problemas multivariable, libres o sujetos a restricciones lineales o no lineales, de igualdad o de desigualdad. Un ejemplo de llamada es:

```
[gbest,gbestvalue,pbest,pbestvalue,x,phip,phig,neval]=...
optimizador_global2(@fobj,x0,options_evo,options_cons);
```

1. Argumentos de entrada:
   - `fobj`: Función objetivo a minimizar. Devuelve dos valores escalares: en primer lugar la función objetivo y en segundo lugar la función violación de restricciones Φ, definida según (2.7) ó (2.8). Por ejemplo, la restricción $x_1 + x_2 \leq 1$ debe escribirse como $-x_1 - x_2 + 1$; mientras que la restricción $x_1 + x_2 = 1$ equivale a $\|x_1 + x_2 - 1\|$
   - `x0`: Solución inicial a partir de la cual los algoritmos iniciarán su evolución. Es una matriz de dimensiones $n \times m$ donde n es la dimensión del vector $\mathbf{x} = (x_1, \ldots, x_n)$ y m es el número de partículas para el algoritmo PSO, 1 para el SA y el número de padres para el GA. Si bien no es necesario, sí es recomendable que la solución inicial sea admisible.

- `options`: Parámetros de configuración de los diferentes algoritmos.

  a) Parámetros comunes a todos los algoritmos: `options.alg` Algoritmo a usar. Puede tomar como valores 'pso', 'sa' o 'ga'.
  `options.max_iter` Es la primera condición de parada. Cuando se alcanzan el número máximo de iteraciones, el algoritmo se detiene. `options.nequal` y `options.tol` forman la segunda condición de parada. Si durante *nequal* iteraciones la función objetivo cambia con respecto a la iteración menos de *tol*, el algoritmo se detiene.

  b) Parámetros de cada algoritmo:

  1) PSO:
  `options.npart`: Número de partículas
  `options.Vmax`: Velocidad máxima v_{max}. Se recomienda que Vmax sea aproximadamente el 20% del mínimo dominio admisible de x_n
  `options.omega0` Valor inicial del parámetro ω, según se definió en la sección 2.1.1.
  `options.omegaT` Valor del parámetro ω para el número máximo de iteraciones. ω decrece linealmente entre ω_0 y ω_T, según se definió en la sección 2.1.1.
  `options.c1` Valor del parámetro c_1 o componente cognitiva. Como regla general, $c_1 \approx 2$, según se definió en la sección 2.1.1
  `options.c2` Valor del parámetro c_2 o componente social. Como regla general, $c_2 \approx 2$, según se definió en la sección 2.1.1

  2) SA:
  `options.paso0` Radio de la circunferencia centrada en s que delimita los posibles s' para la iteración primera.
  `options.pasoT` Radio de la circunferencia centrada en s que delimita los posibles s' para la iteración última. Entre la primera y la última, decrece linealmente.
  `options.T` Parámetro temperatura, según se definió en la sección 2.1.2.

  3) GA:
  `options.nparents`: Número de padres que se usa durante cada iteración.
  `options.childrens`: Número de hijos que se obtienen en cada generación.
  `options.mutation`: Porcentaje de la norma del vector x que

se multiplica por un vector aleatorio con componentes entre -0,5 y 0,5 y se suma a los vectores *x* hijo.

- `options_cons`: Opciones de las restricciones. Tiene un único parámetros, options_cons.tol que es igual al valor de ε en (2.9) y (2.10).

2. Argumentos de salida:

   - `gbest`: Posición del óptimo de todas las partículas en cada iteración. Es una matriz de dimensiones $n \times k$ donde n es el número de variables de la función objetivo y k es número de iteraciones llevadas a cabo.
   - `gbestvalue`: Valor del óptimo de la función objetivo en cada iteración. Es un vector de dimensiones k.
   - `pbest`: Posición del óptimo de cada partícula en cada iteración. Tiene dimensiones $n \times m \times k$ con m el número de padres/partículas del método.
   - `pbestvalue`: Valor del óptimo de la función objetivo para cada partícula/padre en cada iteración. Tiene dimensiones $m \times k$
   - `x`: Matriz que almacena la evolución de la posición de cada partícula o elemento a lo largo de la iteración. Dimensiones $n \times m \times k$
   - `phip`: Valor de la función violación de restricciones (ecuación 2.7) para cada partícula y cada iteración. Dimensiones $m \times k$
   - `phig`: Valor de la función violación de restricciones (ecuacuón 2.7) para el óptimo en cada iteración. Es un vector de dimensiones k.
   - `neval_evo`: Número de evaluaciones finales de la función objetivo.

Apéndice B

Trajectory

B.1. Introducción

Trajectory es una herramienta creada por el Departamento de Ingeniería Aeroespacial de la Universidad de Sevilla que permite el cálculo de las variables que describen la trayectoria. En los algoritmos descritos a lo largo del presente documento, Trajectory es el calculador básico que permite obtener dichas variables en función del tiempo y correspondientes a cada segmento de vuelo que conforma la trayectoria tipo. Para ello Trajectory necesita las condiciones iniciales, las ligaduras que definen el segmento y al menos una condición de parada o finalización del mismo.

B.2. Ecuaciones del movimiento

Para la descripción del movimiento de la aeronave Trajectory emplea un modelo de masa puntual con 3 grados de libertad, apropiado para tareas de predicción de trayectorias y similar a los usados en CTAS, PHARE y ATOMS [25]. De este modo, las ecuaciones describen el movimiento del centro de masas de la aeronave considerado éste como un cuerpo de masa variable. Para un sistema inercial de ejes ligados a Tierra las ecuaciones son:

$$\frac{d\mathbf{x}}{dt} = \mathbf{V}_g \tag{B.1}$$

$$m\frac{d\mathbf{V}_g}{dt} = \mathbf{F}_A + \mathbf{F}_T + m\mathbf{g} \tag{B.2}$$

$$\frac{dm}{dt} = -\dot{m}_F \tag{B.3}$$

donde $\mathbf{x}$ es el vector de posición de la aeronave, $\mathbf{V}_g$ es la velocidad sobre el terreno de la aeronave, $\mathbf{F}_A$, $\mathbf{F}_T$ y $m\mathbf{g}$ son las fuerzas aerodinámicas, propulsivas y gravitacionales, respectivamente, m es la masa de la aeronave, $\dot{m}_F$ es el flujo de combustible consumido y t es el tiempo.

El vuelo de cada segmento se desarrolla en rutas loxodrómicas definidas por rumbo constante, $\chi = \chi_A$, lo cual es equivalente a volar en planos verticales en un modelo de tierra plana.

Para obtener información de las ecuaciones generales se han aplicado las siguientes hipótesis generales [26]:

- Tierra esférica y sin rotación
- Aeronave simétrica y rígida
- Vuelo simétrico en un plano vertical
- Empuje paralelo a la velocidad aerodinámica de la aeronave

Hipótesis que son apropiadas para aeronaves subsónicas de transporte.

Las ecuaciones escalares del movimiento son [26]:

$$m\frac{dV}{dt} = T - D(V,h,L) - mgsin(\gamma), \tag{B.4}$$

$$mV\frac{d\gamma}{dt} = L - mgcos(\gamma), \tag{B.5}$$

$$\frac{dm}{dt} = -c(M,h)T, \tag{B.6}$$

$$(R_E + h)\frac{d\varphi}{dt} = Vcos(\gamma)cos(\chi_A), \tag{B.7}$$

$$(R_E + h)\,cos(\varphi)\frac{d\lambda}{dt} = Vcos(\gamma)sin(\chi_A), \tag{B.8}$$

$$\frac{dh}{dt} = Vsin(\gamma), \tag{B.9}$$

$$\frac{dr}{dt} = Vcos(\gamma)\frac{R_E}{R_E + h}, \tag{B.10}$$

donde r es la distancia horizontal recorrida por la aeronave sobre la superficie terrestre; V, γ, χ son el módulo de la velocidad aerodinámica verdadera, el ángulo de asiento de la trayectoria y el rumbo; φ, λ, h son las coordenadas geodésicas del centro de masas de la aeronave (latitud, longitud, altitud); T es el empuje, L es la sustentación, D es la resistencia aerodinámica; c es el consumo específico de combustible (definido como masa de combustible por unidad de empuje y unidad de tiempo) y R_E el radio terrestre.

Para cerrar el problema es necesario proporcionar las variables de control L y T, o de forma equivalente, 2 condiciones o ligaduras que cierren el problema. En general, son necesarias 3 condiciones, pero en este caso ya se ha impuesto el vuelo a rumbo constante $\chi = \chi_A$. Esta tercera condición va asociada a una tercera variable de control, el ángulo de balance μ; que en este caso resulta ser $\mu = 0$.

Las ligaduras pueden ser de control, como por ejemplo un rating del motor; o ligaduras asociadas al movimiento, como altitud de vuelo constante o Mach de vuelo constante. De forma general las ligaduras son de la forma:

$$g(V,\gamma,\chi,m,\varphi,\lambda,h,T,L,\mu,t)=0$$

Las ligaduras impuestas deben cerrar el problema matemático, es decir, deben ser compatibles entre ellas.

Finalmente, se deben proporcionar condiciones iniciales compatibles con las ligaduras y, al menos, una condición de parada de la integración también compatible con las ligaduras. Esta condición de parada puede ser un tiempo de integración o combinación de las variables. De forma general la condición de parada tendrá la forma:

$$s(V,\gamma,\chi,m,\varphi,\lambda,h,T,L,\mu,t)=0$$

Las ecuaciones del movimiento más las ligaduras forma un sistema de ecuaciones diferenciales algebraicas (DAE). El método empleado en Trajectory para la resolución de las DAE's se basa en la obtención explícita de las variables de control a partir del sistema DAE y su sustitución en las ecuaciones del movimiento, obteniendo por tanto un sistema de ecuaciones diferenciales ordinarias (ODE) que se resuelve numéricamente.

Trajectory también es capaz de de calcular las carreras de despegue y aterrizaje. Para ello, basta añadir a la ecuación que gobierna la evolución de la velocidad la reacción horizontal debida al contacto con el suelo

$$R=-\mu\left(mg-\frac{1}{2}\rho V^2 S C_L\right)$$

que no es más que una resistencia adicional. En esta reacción μ es el coeficiente de fricción (típicamente $\mu=0{,}02$ para el despegue y $\mu=0{,}3$ para el aterrizaje, en este último caso mayor pues contempla la aplicación de frenos) y C_L es conocido.

B.3. Modelos aplicados

Para resolver las ecuaciones es necesario la consideración de 3 modelos: modelo de Tierra, aerodinámico y propulsivo. El modelo de Tierra tiene las siguientes características:

- Tierra esférica de radio $R_E=6356{,}766\,km$
- Gravedad constante $g=9{,}80665\,m/s^2$
- Modelo de atmósfera estándar ISA, que define la densidad ρ, presión p y temperatura Θ como función de la altura h.

El modelo de aeronave proporciona las características de actuaciones requeridas tanto aerodinámicas como propulsivas. El modelo aerodinámico define la resistencia como

$$D = \frac{1}{2}\rho V^2 S C_D$$

donde C_D, coeficiente de resistencia, es conocido como polar del avión

$$C_D = C_D(M, C_L)$$

El coeficiente de sustentación C_L está definido por $L = \frac{1}{2}\rho V^2 S C_L$, donde ρ es la densidad del aire y S es la superficie alar de referencia. El número de Mach viene dado como

$$M = \frac{V}{\sqrt{\kappa R_a \Theta(h)}}$$

donde $\kappa = 1{,}4$ es el cociente de calores específicos, R_a es la constante del aire. La expresión de C_D es necesario modificarla en función de la configuración de la aeronave (flaps y tren de aterrizaje). El modelo detallado de C_D se puede ver en Apéndice C.

El modelo propulsivo define el empuje disponible y el consumo específico. Para el empuje disponible se considera el siguiente modelo

$$T = W_{TO}\delta C_T(M, N_C)$$

donde W_{TO} es el peso de referencia en despegue, $\delta = \frac{p}{p_{SL}}$ es el cociente de presiones (p_{SL} es la presión de referencia a nivel del mar en el modelo de atmósfera ISA) y C_T es el coeficiente de empuje, en general función del número de Mach y del parámetro de control del motor, N_c. Dado un rating de motor (take off, climb o idle), el parámetro de control es función del número de Mach y la altitud $(N_C(M,h))$, y por tanto se puede definir el modelo como

$$T = T_{RATE}(M, h)$$

.

B.4. Interfaz de Trajectory

La comunicación con Trajectory se realiza mediante un conjunto de datos de entrada, a partir de los cuales Trajectory será capaz de calcular la trayectoria y proporcionar las variables que definen la trayectoria como datos de salida. Los datos de entrada necesarios para el funcionamiento de Trajectory son:

- Condiciones iniciales.

```
TAS : velocidad verdadera
PTH : ángulo de asiento de la trayectoria
```

```
HDG : rumbo geográfico
m   : masa de la aeronave
LAT : latitud
LONG: longitud
ALT : altitud
DST : distancia sobre la superficie de la tierra
t   : tiempo
```

- Ligaduras o constraints (CONS). Como se indicó anteriormente, cada segmento de vuelo está definido por dos ligaduras. Los tipos de ligaduras necesarias para generar una trayectoria global tipo son:

  - SPD, mantener velocidad de vuelo constante. Tipos
    - CAS
    - MACH
  - ALT, mantener altitud de vuelo constante. Tipos
    - h_{AGL}. El nivel de referencia para medir alturas es con respecto al nivel de QFE.
    - h_p. La referencia para medir alturas es con respecto al nivel de $1013 HPa$.
  - ENG, mantener rating del motor constante. Tipos
    - TO
    - CLB
    - IDLE
    - MCRZ
  - PTH, mantener ángulo de trayectoria vertical de vuelo constante.
  - HDG, mantener rumbo geográfico constante.

- Eventos (EVENT). Los eventos se utilizan para definir el punto de finalización de un segmento de vuelo unitario correspondiente a la trayectoria tipo. Cuando una variable de vuelo, velocidad, altitud, distancia, way point; alcanza el valor de un evento asociado a esa variable se finaliza el cálculo de ese segmento. Trajectory permite chequear varios eventos a la vez, estando cada uno de ellos definido por 3 campos:

  - Events.Function
  - Events.Options.Type
  - Events.Option.Data

  Por otra parte, los datos de salida generados por Trajectory son:

```
TAS   : velocidad verdadera
PTH   : ángulo de asiento de la trayectoria
HDG   : rumbo geográfico
m     : masa de la aeronave
LAT   : latitud
LONG  : longitud
ALT   : altitud
DST   : distancia sobre la superficie terrestre
BNK   : ángulo de balance
L     : sustentación
T     : empuje
t     : tiempos en los que se evalúa la integración
TE    : tiempo en el que se alcanzó el evento
YE    : valor del evento alcanzado
IE    : evento alcanzado
M     : número de Mach
CAS   : velocidad calibrada
```

que son vectores que expresan la evaluación de dichas variables en el vector de tiempos t.

Con estos datos se podrán obtener las condiciones iniciales del siguiente segmento.

Apéndice C

Modelo aeronave

C.1. Modelo aerodinámico

El modelo de aeronave considerado para las aplicaciones es un Boeing 767-300ER (un avión de transporte típico bimotor de fuselaje ancho) con una superficie alar $S = 283{,}3\,m^2$, peso máximo al despegue $MTOW = 186880\,kg$ y máxima capacidad de combustible de $73635\,kg$ [27]. Se considera la polar definida por Cavcar and Cavcar [28], dada por:

$$\begin{aligned} C_D &= \left(C_{D_{0,i}} + \sum_{j=1}^{5} k_{0j} K^j(M)\right) + \left(C_{D_{1,i}} + \sum_{j=1}^{5} k_{1j} K^j(M)\right) C_L \\ &+ \left(C_{D_{2,i}} + \sum_{j=1}^{5} k_{2j} K^j(M)\right) C_L^2 \end{aligned} \tag{C.1}$$

donde

$$K(M) \frac{(M-0{,}4)^2}{\sqrt{1-M^2}}$$

Los coeficientes incompresibles de la parábola para el modelo de aeronave son $C_{D_{0,i}} = 0{,}01322$, $C_{D_{1,i}} = -0{,}00610$ y $C_{D_{2,i}} = 0{,}06000$ y los coeficientes de compresibilidad se muestran en la Tabla C.1. Esta polar es válida para $M \geq 0{,}4$; para $M < 0{,}4$ los coeficientes de compresibilidad de la parábola no se tienen en cuenta [que se obtiene introduciendo $K = 0$ en la ecuación (C.1)].

j	1	2	3	4	5
k_{0j}	0,0067	−0,1861	2,2420	−6,4350	6,3428
k_{1j}	0,0962	−0,7602	−1,2870	3,7925	−2,7672
k_{2j}	−0,1317	1,3427	−1,2839	5,0164	0,0000

Tabla C.1: Coeficientes de compresibilidad de la polar para el modelo de aeronave.

C.2. Modelo propulsivo

Para el consumo específico de combustible, el modelo lineal definido por Mattingly et al. [29] (y aproximadamente descrita por Miele en [26]) es considerada; viene dada por

$$C = c_{SL}\frac{L_H}{a_{SL}}(1{,}0 + 1{,}2M) \tag{C.2}$$

donde c_{SL} es el consumo específico de combustible al nivel del mar y $M = 0$. Para los motores CF6-80C2 del modelo de avión, tomamos un valo representativo de $c_{SL} = 9{,}0 \times 10^{-6}\, kg/(s \cdot N)$[1]. Para el calor latente del combustible, tomamos $L_H = 43 \times 10^6\, J/kg$.

Con respecto al empuje, el siguiente modelo analítico es considerado para el coeficiente de empuje (está basado en el modelo de Mattingly et al. [29] para la dependencia de M y tiene en cuenta el incremento de C_T con la altura [30]):

$$C_T = \frac{T_{SL}}{W_{TO}}\left(1 + \frac{\gamma - 1}{2}M^2\right)^{\frac{\gamma}{\gamma-1}}\left(1 - 0{,}49\sqrt{M}\right)\frac{1}{\theta} \tag{C.3}$$

donde $\gamma = 1{,}4$ (cociente de calores específicos) y T_{SL} es el máximo empuje al nivel del mar para $M = 0$ para el rating de crucero. Esta expresión define el máximo empuje $T_M = W_{TO}\delta C_T$. Para los dos motores CF6-80C2 del modelo de aeronave se considera $T_{SL} = 5 \times 10^5\, N$ como valor representativo.

[1] Datos disponibles online en http://www.geae.com/engines/commercial/cf6/cf6-80cs.html [dirección visitada en Junio de 2009].

Bibliografía

[1] H. Erzberger and H. Lee. Constrained optimum trajectories with specified range. *Journal of Guidance, Control, and Dynamics, Vol. 3, No. 1*, 1980.

[2] D. M. Pargett and M. D. Ardema. Flight path optimization at constant altitude. *Journal of Guidance, Control, and Dynamics, Vol. 30*, pages 1197–1201, 2007.

[3] A. Franco, D. Rivas, and A. Valenzuela. Minimum-fuel cruise at constant altitude with fixed arrival time. *Aprobado para publicación en Journal of Guidance, Control, and Dynamics.*, 2009.

[4] P.K.A. Menon. Study of aircraft cruise. *Journal of Guidance, Control and Dynamics, Vol. 12, No. 5*, pages 631–639, 1989.

[5] R. Slattery and Y. Zhao. Trajectory synthesis for air traffic automation. *Journal of Guidance, Control, and Dynamics, Vol. 20*, pages 232–238, 1997.

[6] J.N. Barrer. Integrating the flight management system with air traffic control functions- the concept of path objects. *MITRE Technical report MTR 99W000011, MITRE Corporation*, 1999.

[7] M.A. Vilaplana. Co-operative conflict resolution in autonomous aircraft operations using a multi-agent approach. *Ph.D. Thesis, University of Glasgow*, 2002.

[8] D.R. Isaacson and J.E. Robinson III. A knowledge-based conflict resolution algorithm for terminal area air traffic control advisory generation. *AIAA paper 2001-4116, AIAA*, 2001.

[9] J.L. de Augusto. Generador de trayectorias globales de aviones comerciales. *Diploma Thesis, Escuela Superior de Ingenieros de Sevilla*, 2008.

[10] A. Valenzuela and D. Rivas. Conflict detection and resolution in converging air traffic. *AIAA paper 2009-7022, AIAA*, 2009.

[11] K.A. De Jong. Evolutionary computation: a unified approach. *MIT Press, Cambridge MA*, 2006.

[12] A.E. Eiben and J.E. Smith. Introduction to evolutionary computing. *Springer*, 2003.

[13] A.E. Eiben and M. Schoenauer. Evolutionary computing, information processing letters. *Information Processing Letters*, 2002.

[14] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan.*, 1995.

[15] Y.S. Lee, S.M. Shamsuddin, and H.N. Hamed. Bounded pso vmax function in neural network learning. *Bounded PSO Vmax Function in Neural Network Learning*, page 476, 2008.

[16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science. New Series 220*, pages 671–680, 1983.

[17] V. Černý. A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, pages 41–51, 1985.

[18] C.A.C. Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, pages 1245–1287, 2002.

[19] Tetsuyuki Takahama, Setsuko Sakai, and Noriyuki Iwane. Constrained optimization by the ε constrained hybrid algorithm of particle swarm optimization and genetic algorithm.

[20] Tetsuyuki Takahama and Setsuko Sakai. Solving constrained optimization problems by the e constrained particle swarm optimizer with adaptive velocity limit control.

[21] Ximing Liang, Xiang Li, and M. Fikret Ercan. A pso - line search hybrid algorithm. *Springer*, 2009.

[22] Garret N. Vanderplaats. Numerical optimization techniques for engineering design: with applications. *McGraw-Hill*, pages 155–156, 1983.

[23] Inc The MathWorks. MATLAB help `fmincon`. 1984-2009.

[24] D. Rivas and A. Valenzuela. Compressibility effects on maximum range cruise at constant altitude. *Journal of Guidance, Control, and Dynamics, Vol 32, No.5*, pages 1684–1658, 2009.

[25] M. Vilaplana, J. López, E. Valls, and F. Navarro. Courage domain analysis. *Boeing Research and Technology Europe*.

[26] A. Miele. Flight mechanics. theory of flight paths. *Addison-Wesley, Reading, MA*, pages 107,205, 1962.

[27] P. Jackson. Janes all theworlds aircraft. *Janes Information Group*, pages 608–610, 20042005.

[28] A. Cavcar and M. Cavcar. Approximate solutions of range for constant altitude-constant high subsonic speed flight of transport aircraft. *Aerospace Science and Technology, Vol. 8, No. 6*, page 557567, 2004.

[29] J. D. Mattingly. Aircraft engine design, 2nd ed. *AIAA Education Series, AIAA, Reston, VA*, pages 38,71, 2002.

[30] Navarro F. A. Nuic A. Gallo, E. and M. Iagaru. Advanced aircraft performance modeling for atm: Bada 4.0 results. *Proceedings of the 25th Digital Avionics Systems Conference, Inst. of Electrical and Electronics Engineers, Piscataway, NJ*, pages 1–12, 2006.