Universidade do Minho
Escola de Engenharia

Vanesa Fuertes Ibán

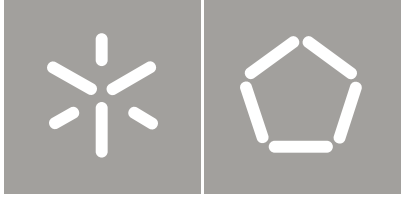
Solución de problemas no lineales con restricciones usando DIRECT y una función Lagrangeana aumentada

Julho de 2011

Estudio de Aprovechamiento Energético
en las Viviendas Mediante Energía Solar

Vanesa Fuertes Ibán

UMinho | 2011



Universidade do Minho
Escola de Engenharia

Vanesa Fuertes Ibán

Solución de problemas no lineales con restricciones usando DIRECT y una función Lagrangeana aumentada

Tese de Mestrado
Engenharia Industrial

Trabalho efectuado sob a orientação da
Professora Doutora Ana Maria Alves Coutinho Rocha

Julho de 2011

Resumen

En este trabajo se realiza un breve estudio sobre la optimización no lineal, en una pequeña clasificación de los diferentes tipos de funciones centrándonos en problemas con y sin restricciones; además de los métodos que se utilizan en la resolución de dichos problemas.

Para la realización de nuestro trabajo llevamos a cabo el estudio y utilización del algoritmo DIRECT [5], que es un método determinista de optimización global. DIRECT comienza la búsqueda del mínimo global dividiendo su espacio (dominio) en partes rectangulares y lo repite continuamente en determinadas iteraciones hasta llegar al punto óptimo.

El uso de la aplicación de DIRECT se desarrolla en primer lugar en problemas de optimización con restricciones del tipo de límites simples (problemas Dixon&Szego [3]), dentro de los cuales representaremos más detalladamente la función Camel.

En segundo lugar utilizamos DIRECT para resolver problemas con restricciones del tipo igualdad y desigualdad (conjunto de problemas g), a través de la utilización de métodos basados en la función Langrangeana aumentada.

La idea central es resolver el problema de optimización con restricciones a través de la resolución de una sucesión de sub-problemas más simples, esos problemas apenas con restricciones de límites en las variables. La utilización de la función Lagrangeana aumentada en un método de penalización tiene como objetivo superar el problema de mal condicionamiento de los sub-problemas sin restricciones. El éxito y la eficiencia de los métodos de penalización basados en la función Lagrangeana aumentada, teniendo como base los métodos de los multiplicadores [1].

Así, fue implementado el método de los multiplicadores basado en la función Lagrangeana aumentada para resolver problemas con restricciones. En cada iteración, es necesaria la resolución de un sub-problema, que fue realizado a través del método DIRECT.

Por último, para evaluar el desempeño del método, este fue aplicado en una colección de problemas g [7] y comparamos con la resolución de los mismos problemas a través del método de los multiplicadores, más con el comando FMINCON (de Matlab [10]) en la resolución de los sub-problemas.

Abstract

This paper provides a brief study on the area of nonlinear optimization, namely in the classification of different types of functions focusing on problems with and without constraints and, in addition, the methods used in solving these problems.

To carry out our work we study and use the DIRECT method [5], that is a deterministic method of global optimization. DIERCT begins the search for global minimum by dividing the space (domain) into rectangular parts and continuously repeat it along the iterations to reach the optimal point.

First using the DIRECT has been applied to solve bound constrained optimization problems (the Dixon&Szego problems [3]), where a more detailed explanation have been carried out to the Camel function.

Second, to solve problems with equality constraints and inequality type (problems g), through the use of methods based on augmented Langrangian functions.

The central idea is to solve the constrained optimization problem via solving a sequence of simpler sub-problems, called simple bound problems. The subproblems are based on an augmented Lagrangian function. The use of an augmented Lagrangian function in a penalty method aims to overcome the problem of ill-conditioning of the sub-problems without constraints. It has been shown through years the success and efficiency of penalty methods based on augmented Lagrangian functions, that are based on the multipliers method [1].

Hence, a multipliers method based on an augmented Lagrangian function has been implemented to solve constrained problems. In each iteration, is necessary to solve a subproblem that will be done by the DIRECT method.

Finally, to evaluate the performance of the implemented method, we solve a subset of the collection of gs problems [7] and compared with the multipliers method but with the FMINCON command (from Matlab [10]) when solving the subproblems.

INDICE

1 INTRODUCCIÓN

1.1 Encuadramiento

Después de la 2ª Guerra Mundial, se desarrollaron varios métodos computacionales para resolver problemas de optimización, siendo la mayoría de estos, eficaces en la resolución de problemas simples –problemas con pocas variables, con funciones lineales o funciones unimodales. Unas décadas después, surgirían problemas en varias áreas, tales como en la ingeniería, logística, industria, ciencia etc. que eran definidos por funciones no convexas y poseían múltiplos óptimos locales [1].

Matemáticamente la optimización consiste en determinar el óptimo de una función objetivo en que las variables de decisión deben de respetar un cierto número de restricciones. En ciertas aplicaciones, la resolución de problemas de optimización no lineal a través de métodos que encuentran el óptimo local puede ser suficiente, sin embargo, en otras situaciones existen la necesidad de encontrar el mejor de los óptimos locales [1, 9].

De este modo, varios investigadores pertenecientes a áreas como Matemática Aplicada, Investigación Operacional, Ingeniería Industrial y Ciencias de Informática, comenzaron a estudiar la optimización global, cuyo objetivo consiste en la búsqueda de la mejor solución dentro de varias soluciones óptimas locales, esto es, la solución óptima global.

Hay dos clases de métodos de optimización global: los métodos estocásticos e los métodos deterministas. Los métodos estocásticos que trabajan con un conjunto de puntos generados aleatoriamente y son generalmente dispendiosos en términos de tiempo de ejecución. Además, no hay garantía teórica de convergencia para soluciones globales, porque generalmente las soluciones quedan presas en óptimos locales. Los métodos deterministas son atractivos visto que no son así dispendiosos en términos de exigencias computacionales y convergen, con garantía teórica, en un número finito de iteraciones para una solución global [8].

DIRECT es un método determinista que fue inicialmente creado para resolver problemas difíciles de optimización global con restricciones de tipo límites simples. El método DIRECT inicia el proceso iterativo dividiendo el espacio de procura en hiper-rectángulos, a través de criterios de decisión (tamaño y valor de la función en el centro) bien definidos. Se pretende analizar con detalle este proceso y también mejorar su eficiencia [5, 6].

Uno de los procedimientos más comunes para tratar restricciones no lineales de igualdad y desigualdad se basa en la utilización de funciones de penalidad. El problema con restricciones es transformado en un problema sin restricciones adicionando un término de penalidad a la función objetivo. El término de penalidad incorpora las restricciones de igualdad y desigualdad, además del parámetro de penalidad. El método de penalidad envuelve la resolución de una secuencia de sub-problemas sin restricciones (función de penalidad), para una sucesión de valores del parámetro de penalidad [1, 4].

No obstante, un caso particular de una función de penalidad es la función Lagrangeana aumentada. La utilización de la función Lagrangeana aumentada en un método de penalización tiene como objetivo exceder el problema de mal condicionamiento de los sub-problemas sin restricciones, a través del vector de los multiplicadores de Lagrange. El éxito y la eficiencia de los métodos de penalización basados en la función Lagrangeana aumentada, teniendo como base los métodos de los multiplicadores, depende de la precisión con que el vector de los multiplicadores es estimado [1].

En este trabajo, se pretende extender la aplicación de DIRECT a problemas de optimización con restricciones de tipo igualdad y desigualdad, a través de la utilización de métodos basados en funciones Langrangeanas aumentadas [2]. La idea central es resolver el problema de optimización con restricciones a través de la resolución de una sucesión de sub-problemas más simples sólo con restricciones de límites simples. La resolución de los sub-problemas será hecha por el algoritmo DIRECT [5].

1.2 Optimización global y local

La optimización es el proceso de encontrar el punto que minimiza (o maximiza) una función. Los algoritmos de optimización más rápidos solo buscan una solución local, un punto en el que la función objetivo es menor que en todos los puntos posibles en sus proximidades. No siempre se encuentra lo mejor de todos esos mínimos (o máximos), es decir, la solución global. Soluciones globales son necesarias (o al menos deseables) en algunas aplicaciones, por lo general son difíciles de identificar y aún más difícil de localizar.

Los problemas de programación lineal entran en la categoría de programación convexa. Un caso especial importante de programación convexa es aquel en el que todas las soluciones locales son también soluciones globales. Sin embargo, en general los problemas no lineales, tanto con o sin restricciones, pueden tener soluciones locales que no soluciones globales.

1.3 Clasificación de máximos y mínimos

Un concepto básico para la comprensión del procedimiento de optimización son los máximos y mínimos de la función, puesto que es el principio fundamental de búsqueda. A continuación, se presentan algunas de las definiciones relativas a la optimización de un problema sin restricciones.

Sea $V(x^*, \delta)$ una vecindad de x^* de radio $\delta (\delta > 0)$

- x^* es un minimizador local (relativo) fuerte (débil) de f si $\exists \delta > 0$

  - $f(x)$ es definida en $V(x^*, \delta)$
  - $f(x^*) < f(x), \quad \forall x \in V(x^*, \delta); \ x \neq x^*$ (fuerte)
  - $f(x^*) \leq f(x), \quad \forall x \in V(x^*, \delta); \ x \neq x^*$ (débil)

- x^* es un maximizador local fuerte (débil) de f si ∃ δ > 0

  - $f(x)$ es definida en $V(x^*, \delta)$
  - $f(x^*) > f(x), \quad \forall x \in V(x^*, \delta); \; x \neq x^*$ (fuerte)
  - $f(x^*) \geq f(x), \quad \forall x \in V(x^*, \delta); \; x \neq x^*$ (débil)

- x^* es un minimizador global (absoluto) fuerte (débil) si $f(x^*) < f(x)$ $((x^*) \leq f(x))$, para todo x que pertenece al dominio de f.

- x^* es un maximizador global fuerte (débil) si $f(x^*) > f(x)$ $(f(x^*) \geq f(x))$, para todo x que pertenece al dominio de f.

Así, el óptimo global tiene el menor (o mayor) valor que la función objetivo puede obtener. Además, tenga en cuenta que todo óptimo global es local, no obstante un óptimo local puede no ser global. En la Figura 1.1 se puede ver una representación gráfica de diferentes tipos de máximos y mínimos.

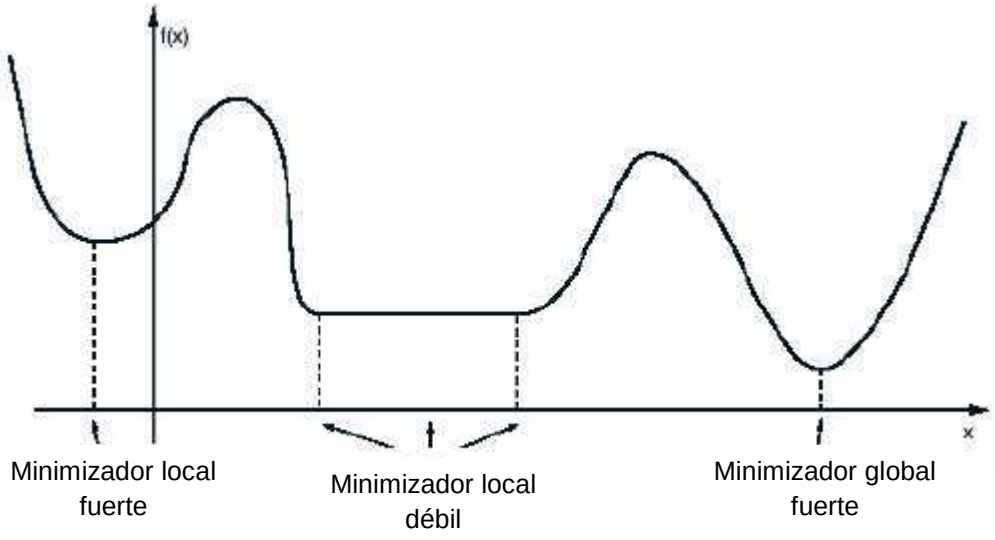


Figura 1.1 - Representación de los diferentes tipos de máximos y mínimos.

Más adelante estudiaremos las condiciones que describen los puntos estacionarios de la función objetivo, es decir, el sistema de ecuaciones que resulta de igualar a cero todas las derivadas parciales de la función. Las condiciones de segundo orden del problema de optimización son aquellas que involucran derivadas parciales de segundo orden, y que, en algunos casos, nos permiten determinar la naturaleza de los puntos estacionarios.

Nótese, como hemos dicho, la propiedad de diferenciabilidad permite caracterizar los extremos locales (mínimos y máximos), proporcionando condiciones necesarias e suficientes para encontrar una solución optima. En este trabajo se centrará la atención en la búsqueda de mínimos, ya que los máximos pueden ser obtenidos a partir de la relación:

$$\text{maximizar } f(x) = \text{- minimizar } (-f(x))$$

1.4 Optimización estocástica y determinista

Existen dos clases de métodos de optimización global. Los métodos estocásticos y los deterministas.

Los métodos estocásticos, basados en la población, trabajan con un conjunto de puntos generados aleatoriamente y son generalmente dispendiosos en términos de tiempo de ejecución. Además de eso, no hay garantía teórica de convergencia para soluciones globales, pues generalmente las soluciones quedan presas en óptimos locales.

En el tipo de problemas de optimización estocástica, el modelo no puede ser completamente especificado, ya que depende de las cantidades que son desconocidas en el momento de la formulación. Esta característica es compartida por muchos modelos de problemas (ejemplo: planificación económica y financiera) que a menudo dependen de los movimientos futuros de las tasas de interés y el comportamiento futuro de la economía. Con frecuencia, sin embargo, los modeladores pueden predecir o estimar las cantidades desconocidas con cierto grado de confianza.

Pueden, por ejemplo, llegar a un número de escenarios posibles para los valores de las incógnitas e incluso asignar una probabilidad en cada escenario. Algoritmos de optimización estocástica utilizan esta cuantificación de la incertidumbre para producir soluciones que optimizan el rendimiento esperado del modelo. Sin embargo, muchos algoritmos de optimización estocástica, se forman mediante la formulación de uno o más sub-problemas deterministas.

Los métodos deterministas son atractivos visto que no son dispendiosos en términos de exigencias computacionales, pero convergen, con garantía teórica, en un número finito de iteraciones para una solución global. Su principal desventaja se debe a que puede ser engañado por los óptimos locales y no llegar al óptimo global.

1.5 Caracterización de los algoritmos de optimización

Generalmente, los algoritmos de optimización son iterativos. Comienzan con una aproximación inicial de los valores óptimos de las variables y generan una secuencia de estimaciones más precisas hasta llegar a una solución. La estrategia utilizada para pasar de una iteración a la siguiente distingue a un algoritmo de otro. La mayoría de las estrategias hacen uso de los valores de la función objetivo, de las restricciones y, posiblemente, la primera y segunda derivada de estas funciones.

Algunos algoritmos acumulan la información recogida en las iteraciones anteriores, mientras que otros utilizan sólo la información local desde el punto actual. Independientemente de estos detalles, todos los buenos algoritmos deben poseer las siguientes propiedades:

- Robustez: Deberían funcionar bien en una amplia variedad de problemas de su clase, para todas las opciones razonables de las variables iniciales.

- Eficiencia: No deben demorar demasiado tiempo en almacenar la información en el ordenador.

- Precisión: Deben ser capaces de identificar una solución con precisión, sin que sea demasiado sensible a errores en los datos o en los errores aritméticos de redondeo que se producen cuando el algoritmo se ejecuta.

Estos objetivos pueden entrar en conflicto. Por ejemplo, un método rápido de convergencia para la programación no lineal puede ser necesario demasiado almacenaje del equipo en problemas grandes. Por otro lado, un método robusto también puede ser el más lento. Ventajas y desventajas entre la tasa de convergencia y los requisitos de almacenamiento, y entre la robustez y velocidad, y así sucesivamente, son temas centrales en la optimización numérica.

La teoría matemática de la optimización se utiliza tanto para caracterizar los puntos óptimos y para proporcionar la base para la mayoría de los algoritmos. No es posible tener una buena comprensión de optimización numérica, sin una sólida comprensión de la teoría de apoyo. En consecuencia, a una sólida comprensión de condiciones de optimización, así como el análisis de convergencia, que revela los puntos fuertes y débiles de algunos de los algoritmos [1].

1.6 Objetivos de la tesis

En este proyecto se pretende hacer a implementación en MatLab [10] de un algoritmo que combine una técnica de penalidad y el DIRECT para resolver problemas no lineales, con restricciones, de optimización global.

DIRECT está implementado en lenguaje y ambiente MatLab [5,6]. Así, se pretende hacer el desarrollo de un algoritmo que implemente una técnica de penalidad, en particular la función Lagrangeana aumentada, en que la resolución de los sub-problemas, en cada iteración, debe ser hecha mediante el DIRECT, de forma que resuelva problemas no lineales de optimización global, con restricciones. La programación del algoritmo debe ser hecha en MatLab y el algoritmo debe ser probado con un conjunto de problemas de optimización.

El ambiente MatLab se ha convertido en una de las herramientas más importantes para la computación científica debido a su lenguaje de alto nivel basado en matrices que resuelven los problemas computacionales de manera más rápida y fácil. De hecho, ya existen muchos investigadores/utilizadores que utilizan MatLab como ambiente de trabajo.

Este trabajo tiene como objetivo resolver problemas no lineales con restricciones a través del método de los multiplicadores, basado en la función Lagrangeana aumentada, y del método DIRECT. Tenga en cuenta que, la implementación de una técnica de estas requiere la selección de valores para el parámetro de penalidad, que es una decisión problemática. En general, un valor apropiado para el parámetro de penalidad depende de la solución. Además de eso, el método implica también, en

cada iteración externa, la actualización del vector de los multiplicadores. El método de los multiplicadores, basados en la función Lagrangeana aumentada, tiene gran interés debido a su éxito que es verificado en la resolución de problemas de optimización global [2].

2 OPTIMIZACIÓN DE PROBLEMAS

2.1 Introducción

En la optimización, los problemas generalmente se pueden clasificar de acuerdo a la naturaleza de la función objetivo y restricciones (lineal, no lineal, convexa y no convexa), del número de variables (grande o pequeño), de las funciones (diferenciable o no diferenciable) etc. Posiblemente la diferencia más importante sea la distinción entre los problemas que tienen restricciones y los que no las tienen.

Los problemas sin restricciones surgen directamente en muchas aplicaciones prácticas. Si existen limitaciones naturales de las variables, a veces, es mejor el desconocimiento y asumir que no tienen ningún efecto sobre la solución óptima. Los problemas sin restricciones surgen también como la reformulación de los problemas de optimización con restricciones en los que las restricciones se sustituyen por los términos de penalidad en la función objetivo y tienen efecto de violaciones.

Por otro lado, los problemas con restricciones poden surgir de los problemas que incluyen restricciones implícitas sobre las variables. Estas restricciones pueden ser simples, tales como límites, restricciones lineales más generales (desigualdades ≤), o no lineales que representan relaciones complejas entre las variables. Cuando tanto la función objetivo y todas las restricciones son funciones lineales de x , el problema será de programación lineal. Los problemas de programación no lineal, en los que al menos algunas de las restricciones o el objetivo son funciones no lineales tienden a surgir en el campo de ciencias e ingenierías [1].

Además, el desenvolvimiento de algoritmos para resolver el problema general de la programación no lineal (con una función objetivo no lineal y restricciones no lineales) es una tarea difícil, pero existen algunos métodos que tienen dado buenos resultados. Más adelante serán presentados algunos de ellos.

2.2 Problemas sin restricciones

La forma general de un problema sin restricciones es la siguiente:

$$\min_{x \epsilon \mathbb{R}^n} f(x) \qquad (1)$$

en que

- n representa el número de variables del problema,
  - si $n = 1$ problema unidimensional y x es un escalar
  - si $n > 1$ problema multidimensional y x es un vector de dimensión n

2.2.1 Condiciones de optimalidad

En esta sección vamos a presentar las condiciones que deben ser satisfechas cuando un $x^* \in \mathbb{R}^n$ es minimizador del problema (1).

Asumimos que $f(x)$ es continuamente diferenciable hasta 2º orden.

- Condición necesaria y suficiente de 1º orden:

Los puntos estacionarios de la función f son los que el vector gradiente, que es el vector compuesto por las derivadas parciales de la función f, es cero, o sea verifican $\nabla f(x) = 0$, donde se pueden distinguir mínimizador, maximizador o punto de inflexión de f.

Si x^* es una solución del problema sin restricciones entonces el vector gradiente evaluadas en el punto x^* se anula, o sea $\nabla f(x^*) = 0$. Si $\nabla f(x^*) = 0$ entonces x^* es candidato a minimizador de f.

- Condición necesaria de 2º orden:

Si x^* es una solución del problema que satisface la condición de 1º orden y si la matriz hessiana de f, la $\nabla^2 f(x^*)$, es semi-definida positiva entonces x^* puede ser un punto minimizador o un punto de inflexión de f.

Tenga en cuenta que una matriz es semi-definida positiva si todos los determinantes de las submatrices principales son no negativos (≥ 0).

- Condición suficiente de 2º orden:

Si x^* es un punto que verifica la condición de 1º orden y si la matriz hessiana calculada en x^*, $\nabla^2 f(x^*)$, es definida positiva entonces x^* es un mínimizador local fuerte de dicho problema.

Tomando $\nabla f(x^*) = 0$:

- Las condiciones necesarias y suficientes de 2º orden para un máximizador son respectivamente

$\nabla^2 f(x^*)$ es semi-definida negativa;

$\nabla^2 f(x^*)$ es definida negativa;

si $\nabla^2 f(x^*)$ es indefinida, entonces x^* será un punto de inflexión.

Una matriz se dice definida negativa si todos los determinantes das submatrices principales tienen signos opuestos alternativamente negativos e positivos, siendo el primer negativo. Si uno es nulo entonces la matriz es semi-definida negativa.

Para identificar una matriz semi-definida positiva o semi-definida negativa, la secuencia de los determinantes das submatrices principales tienen que tener por lo menos un elemento no nulo.

Si una matriz no es definida positiva, ni es semi-definida positiva, ni definida negativa, ni semi-definida negativa entoces es una matriz indefinida.

RESUMEN

Sea x^* un punto para el cual $\nabla f(x^*) = 0$ y $\nabla^2 f(x^*)$ ≠ matriz nula:

- Si $\nabla^2 f(x^*)$ es definida positiva, x^* es un punto minimizador de f.
- Si $\nabla^2 f(x^*)$ es definida negativa, x^* es un punto maximizador de f.
- Si $\nabla^2 f(x^*)$ es semi-definida positiva, x^* es un punto minimizador o un punto de inflexión de f.
- Si $\nabla^2 f(x^*)$ es semi-definida negativa, x^* es un punto maximizador o un punto de inflexión.
- Si $\nabla^2 f(x^*)$ es indefinida, x^* es un punto de inflexión.

2.2.2 Métodos de resolución de problemas sin restricciones

Dentro de los métodos de resolución de problemas sin restricciones podemos distinguir dos tipos: los métodos de búsqueda directa y los métodos de gradiente.

Los métodos de búsqueda directa sólo usan información de la función objetivo f, son apropiados para los problemas no diferenciables, no obstante también pueden ser utilizados en problemas diferenciables. Uno de estos métodos más conocido es el método de Nelder-Mead.

Los métodos del gradiente usan información de la función y de las derivadas y sólo pueden ser utilizados para resolver problemas diferenciables. Como principal diferencia con los métodos de búsqueda directa es que estos convergen más rápidamente. Dentro de esta clase de métodos podemos citar el Método de Newton y Quasi-Newton.

Seguidamente haremos una breve presentación de algunos métodos usados en la resolución de problemas sin restricciones.

Método de Nelder-Mead

El método de Nelder-Mead es utilizado en optimización multidimensional y no necesita de información de las derivadas. En cada iteración define un ‘simplex’ que es un poliedro definido en $\mathbb{R}^n$. Los vértices del ‘simplex’ de dimensión n serán $n+1$ vértices.

La forma de utilización será introduciendo la función objetivo que interesa hacer mínima. Como cada vez que se ejecuta este modelo, que tiene en cuenta factores externos se consume mucho tiempo de cálculo es importante variar los valores con idea para no malgastar recursos.

El método Nelder-Mead genera una nueva posición de prueba (valor buscado) extrapolando el comportamiento de la función en los vértices de un simplex. Así no es necesario el cálculo probando con todos los valores posibles de la función, sino que el algoritmo va reemplazando cada vez uno de los puntos de prueba ajustando con idea para encontrar la solución que minimiza la función más rápidamente.

El modo más sencillo de hacerlo es reemplazar el peor punto con un punto reflejado en el resto de $n-1$ puntos considerados como un plano (de ahí la extrapolación). Si este punto da mejor resultado, el algoritmo prueba a ‘’estirarse’’ tomando los valores exponencialmente en un línea que contenga este punto.

Por otra parte, si este nuevo punto no es mucho mejor que el valor previo, entonces estamos en un valle (buscamos un mínimo) y el algoritmo encoge el simplex hacia el mejor punto. Como otros algoritmos de optimización, Nelder-Mead a veces se queda bloqueado en un mínimo local, por tanto, el algoritmo se da cuenta y se reinicia con un nuevo simplex que empiece en el mejor valor encontrado. Existen muchas variaciones dependiendo de la naturaleza del problema que se quiera resolver. La más usual es, quizá, usar un simplex pequeño de tamaño constante que salte de gradientes locales a máximos local.

El criterio de parada consiste en verificar que el tamaño relativo del simplex es igual o inferior a una cantidad pequeña, entonces el proceso iterativo termina (el vértice del simplex con menor valor de la función objetivo es considerado como la mejor aproximación calculada a la solución), sino, el proceso iterativo continua.

Método Newton

El método de Newton es un algoritmo eficiente para encontrar aproximaciones el máximo o mínimo de una función, encontrando los ceros de su primera derivada. El método de Newton es un método abierto, en el sentido de que su convergencia global no está garantizada. La única manera de alcanzar la convergencia es seleccionar un valor inicial lo suficientemente cercano a la raíz buscada. Así, se ha de comenzar la iteración con un valor razonablemente cercano al cero.

La relativa cercanía del punto inicial a la raíz depende mucho de la naturaleza de la propia función; si ésta presenta múltiples puntos de inflexión o pendientes grandes en el entorno de la raíz, entonces las probabilidades de que el algoritmo diverja aumentan, lo cual, exige seleccionar un valor supuesto cercano a la raíz. Una vez se ha hecho esto, el método linealiza la función por la recta tangente en ese valor supuesto. La abscisa en el origen de dicha recta será, según el método, una mejor aproximación de la raíz que el valor anterior. Se realizarán sucesivas iteraciones hasta que el método haya convergido lo suficiente [11].

Método de Newton básico: está basado en una aproximación local de $f(x)$ por una función cuadrática. Sea $x^{(k)}$ una aproximación a x^*, y usando la expansión de Taylor de f se origina una nueva aproximación del mínimo de $f(x)$. La solución del sistema de Newton es el vector de dirección de Newton d. El proceso se debe repetir para $x^{(k+1)}$, no obstante la dirección podrá ser ascendente, ortogonal o descendente.

También el sistema de Newton puede no tener soluciones, o al contrario, infinidad de ellas. Además de eso existen una serie de limitaciones debidas a la dirección, por lo tanto en cada iteración se implementaran diferentes soluciones que dan origen al método de seguridad de Newton [11].

El método de Newton tiene convergencia local (la convergencia para la solución sólo es garantizada si la aproximación inicial x estuviera cercana a la solución) y cuadrática si verifica:

$$\left\|x^{(k+1)} - x^*\right\| \leq y\left\|x^{(k)} - x^*\right\|^2 \qquad \text{para } y > 0$$

Además de este método posee la propiedad de la terminación cuadrática, por ejemplo si $f(x)$ $(x \in \mathbb{R}^n)$ fuera una función cuadrática y convexa el método de Newton necesita un máximo de *n* iteraciones para encontrar la solución [3].

De entre las desventajas se cita el respecto al cálculo de las segundas derivadas, si la expresión de f es complicada, estas son difíciles de calcular y además exigen una gran cantidad de cálculos para valores de n grandes. La convergencia con este método es local por lo tanto para llevarlo hacia una convergencia global debe implementarse una técnica de globalización para garantizar que el método converge hacia la solución, a partir de cualquier aproximación inicial.

Métodos Quasi-Newton

El método Quasi-Newton es una variante del método Newton. Haciendo una aproximación tanto en la inversa de la matriz Hessiana, como en el sistema de Newton obtenemos una nueva dirección. Esta será el producto de la inversa de la matriz Hessiana por el vector llamada dirección de Quasi-Newton. De esta forma evitamos el cálculo de las 2as derivadas y la resolución del sistema lineal en cada iteración, sustituyendo este por el producto de una matriz por un vector. Como la inversa de una matriz puede ser un proceso moroso, también se usan fórmulas para substituir la inversa de la matriz Hessiana para una aproximación a ella..

Las propiedades de los métodos Quasi-Newton es que tiene convergencia local cuando la convergencia hacia la solución solo es garantizada si la aproximación inicial x estuviera cerca de la solución.

Además este método satisface la propiedad de terminación cuadrática (o sea, el mínimizador de una función cuadrática que se obtiene en n o menos iteraciones). También contemplamos algunas limitaciones del método debido a que los errores de redondeo que se cometen en los cálculos pueden hacer que la matriz Hessiana deje de ser positiva y la dirección Quasi-Newton deja de ser descendente para la función en x.

2.3 Problemas con restricciones

Los problemas con restricciones podemos dividirlos en:

a) Problemas con restricciones de límites simples.

$x^* = \min_{x \epsilon \mathbb{R}^n} f(x)$ sujeto a: $lb_i \leq x_i \leq ub_i\ ,\ i = 1, \dots, n$

b) Problemas con restricciones de igualdad.

$x^* = \min_{x \epsilon \mathbb{R}^n} f(x)$ sujeto a: $c_i(x) = 0,\ i = 1, \dots, m$

c) Problemas con restricciones de desigualdad.

$x^* = \min_{x \epsilon \mathbb{R}^n} f(x)$ sujeto a: $g_i(x) \leq 0,\ j = 1, \dots, p$

siendo x^* una solución del problema con restricciones y donde se supone que a función objetivo f y las funciones de las restricciones $c: \mathbb{R}^n \to \mathbb{R}^n$ y $g: \mathbb{R}^n \to \mathbb{R}^n$, son funciones no lineales dos veces continuamente diferenciables en un subconjunto de $\mathbb{R}^n$.

2.3.1 Condiciones de optimalidad del problema de optimización con restricciones de igualdad

En esta sección von ser presentadas las condiciones de optimalidad en términos de la función Lagrangeana para el problema de optimización con restricciones de igualdad. Así, sea la función Lagrangeana asociada al problema b) es dada por

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i \cdot c_i(x) \qquad (3)$$

en que $\lambda \in \mathbb{R}^m$ es el vector de los multiplicadores de Lagrange asociado a las restricciones de igualdad que son dadas por $c(x)$.

La interpretación intuitiva del Lagrangiana es que se trata de una nueva función objetivo, que nos permite olvidarnos de que las variables han de obedecer las restricciones. Construimos la función Lagrangeana añadiendo a la función objetivo original un término de penalización para cada restricción, de tal forma que los términos adicionales de penalización sólo tienen efecto cuando las respectivas restricciones no se cumplen. Para aplicar la regla de los multiplicadores de Lagrange, definimos en primer lugar la función Lagrangeana y después igualamos a cero las derivadas parciales de la misma con respecto a cada una de las variables originales. Finalmente, añadimos las restricciones para obtener un sistema con tantas ecuaciones como incógnitas.

Para aplicar el método de los multiplicadores de Lagrange, necesitamos que los gradientes de las restricciones evaluados en el maximizador o minimizador sean linealmente independientes. Cuando esta condición no se cumple, los gradientes de las restricciones no describen correctamente el plano tangente al conjunto factible, y las cosas fallan: pueden existir soluciones al problema de optimización que no aparecen cuando aplicamos el método de los multiplicadores de Lagrange.

La condición de regularidad dice que: Sea x^* una solución del problema, si los vectores gradientes de las de las restricciones calculados en la solución fueran linealmente independientes, entonces x^* es un punto regular. Esta condición hace notar que el método de los multiplicadores de Lagrange detecta cualquier maximizador o minimizador local en un problema con restricciones de igualdad, *siempre que éstos sean puntos regulares*. Notemos que la regularidad es una condición suficiente, pero no necesaria, para la validez de la regla de los multiplicadores de Lagrange.

- Condiciones necesarias y suficientes de 1º orden:

Para localizar los puntos estacionarios de un problema con restricciones, las primeras derivadas parciales de la función Lagrangiana con respecto a las variables x e a los multiplicadores λ son igualadas a cero (condiciones necesarias).

Así, sea x^* una solución del problema con restricciones, si x^* es un punto regular entonces existe un λ^* que cumple

$$\begin{cases} \nabla_x L(x^*, \lambda^*) = \nabla f(x^*) + \lambda^* \nabla c(x^*) = 0 & \text{para } n \text{ ecuaciones} \\ \nabla_\lambda L(x^*, \lambda^*) = -c(x^*) = 0 & \text{para } m \text{ ecuaciones} \end{cases}$$

Este sistema de $n + m$ ecuaciones no lineal en x define las condiciones KKT (Karush-Kuhn-Tucker). La solución será (x^*, λ^*) que es punto estacionario de la Lagrangeana y se llama punto KKT.

- Condiciones necesarias de 2º orden:

Sea x^* una solución del problema con restricciones, si x^* es un punto regular que satisface las condiciones KKT, entonces para todos los vectores $s \in \mathbb{R}^n$ que verifican $\nabla c(x^*)^T s = 0$ (vectores tangentes a las curvas admisibles) se tiene $s^T \nabla^2_{xx} L(x^*, \lambda^*) s \geq 0$.

- Condiciones suficientes de 2º orden:

Si (x^*, λ^*) es un punto KKT, es decir

$$\begin{cases} \nabla_x L(x^*, \lambda^*) = 0 \\ \nabla_\lambda L(x^*, \lambda^*) = 0 \end{cases}$$

y si $s^T \nabla^2_{xx} L(x^*, \lambda^*) s > 0$ para todo $s (s \neq 0)$ tal que $\nabla c(x^*)^T s = 0$, entonces x^* es punto minimizador local fuerte.

No obstante, si $s^T \nabla^2_{xx} L(x^*, \lambda^*) s < 0$, todo $s(s \neq 0)$ tal que $\nabla c(x^*)^T s = 0$, entonces x^* es punto maximizador local fuerte.

2.3.2 Métodos de resolución

En esta sección se presenta algunos de los métodos más comunes para la optimización de problemas con restricciones, entre ellos;

- Programación secuencial cuadrática (*Sequential quadratic programming)*
- Método de penalidad
- Método primal dual de puntos interiores

A continuación, presentamos las ideas básicas que caracterizan cada uno de los métodos.

Método de programación cuadrática secuencial (*SQP*)

Este método es tal vez el más conocido para resolver problemas de optimización no lineal con todo tipo de restricciones (igualdad, desigualdad o límites simples).

SQP es un método iterativo que resuelve en cada iteración un problema de programación cuadrática (QP). La solución de este problema cuadrático proporciona la dirección de búsqueda con la cual se calcula la nueva aproximación. Se estima la Hessiana de la función Lagrangeana en cada iteración mediante la fórmula BFGS. Basado en el trabajo de Biggs , Han y Powell, el método permite imitar el método de Newton para la optimización con restricciones como se hace para la optimización sin restricciones [8].

En cada iteración, se hace una aproximación de la función Hessiana o de Lagrange utilizando el método Quasi-Newton actualizado. Esto se utiliza para generar un sub-problema QP cuya solución se utiliza para formar una dirección de la búsqueda de un procedimiento en línea. Teniendo en cuenta una descripción del problema, la idea principal es la formulación de un sub-problema QP basado en una aproximación cuadrática de la función de Lagrange.

Aquí se simplifica la función objetivo, suponiendo que los límites se han expresado como restricciones de desigualdad. Se puede obtener el sub-problema QP linealizando las restricciones no lineales. El proceso termina cuando la aproximación esté cerca de la solución del problema. Esta proximidad es verificada con las condiciones KKT

Método de penalidad

Los métodos de penalidad han sido los primeros que surgirán en la tentativa de trabajar con restricciones no lineales. Así, uno de los procedimientos más comunes para tratar restricciones no lineales de igualdad y desigualdad se basa en la utilización de funciones de penalidad.

El problema original con restricciones (tanto de igualdad como desigualdad,) va a ser transformado en un problema sin restricciones (de más fácil resolución),

adicionando un término de penalidad a la función objetivo. El término de penalidad es tanto mayor cuanto más alejado está de la región admisible y es nulo si el punto estuviese en la región admisible. Entonces, la función objetivo del problema sin restricciones será igual a la función objetivo del problema original más los términos que penalizan la violación de las restricciones.

$$\Phi(x,\mu) = f(x) + P(x,\mu)$$

en que $\mu > 0$ es el parámetro de penalidad.

El término que penaliza la violación de las restricciones $\mathrm{P}(x,\mu)$ se llama término de penalidad, mide la violación de las restricciones y depende de μ. El método de penalidad envuelve la resolución de una secuencia de sub-problemas sin restricciones, para una sucesión de valores del parámetro de penalidad [4].

La idea fundamental es que si μ crece indefinidamente, la solución de $\Phi(x,\mu)$ que es designada por

$$x^*(\mu) = \arg\min_{x\epsilon\mathbb{R}^n} \Phi(x,\mu).$$

será cada vez más próxima de la solución del problema original con restricciones, x^*. Por otro lado, la relación entre $x^*(\mu)$ y x^* dependerá de la función de penalidad usada.

Dentro de los métodos de penalidad podemos hacer la siguiente clasificación de funciones de penalidad para la resolución de los diferentes problemas:

- Función de penalidad cuadrática.
- Función de penalidad de valor absoluto.
- Función de penalidad exacta.
- Función Lagrangeana aumentada.
- Función barrera para las restricciones de desigualdad.
- Función barrera logarítmica.

De entre todos ellos, centraremos nuestra atención en el método que utiliza la función Lagrangeana aumentada, que desarrollaremos más adelante.

Método primal-dual de puntos interiores:

El método que a continuación se describe es una extensión del método de los puntos exteriores desarrollado para programación lineal y cuadrática en problemas no lineales y no convexos.

Este método puede ser aplicado a la resolución de problemas de optimización no lineales con cualquier tipo de restricción (desigualdades, igualdades y de límites simples).

La estrategia usada por este método primal-dual de puntos interiores comienza por introducir variables de holgura no negativas transformando el problema en un problema con restricciones de igualdades y desigualdades simples.

Las desigualdades simples son eliminadas, incorporándolas a un término barrera en la función objetivo, originando un problema nuevo (problema barrera), obteniendo así un nuevo problema que no es más que una sucesión de problemas parametrizados (llevan un nuevo parámetro en la función objetivo).

3 MÉTODO DE LA LAGRANGEANA AUMENTADA

Como ya se mencionó anteriormente uno de los procedimientos más comunes para tratar restricciones no lineales de igualdad y desigualdad se basa en la utilización de funciones de penalidad.

La utilización de la función Langrangeana aumentada es un método de penalización que tiene como objetivo pasar un problema mal condicionado a sub-problemas sin restricciones, a través del vector de los multiplicadores de Lagrange.

El éxito y la eficiencia de los métodos de penalización basados en la función Lagrangeana aumentada, depende de la precisión con la que el vector de los multiplicadores es estimado, teniendo como base el método de resolución de los multiplicadores [1, 8].

3.1 Funcion Lagrangeana aumentada

En este trabajo se pretende resolver problemas no lineales con restricciones a través del método de la función Lagrangeana aumentada, que puede ser considerado como un caso particular de una técnica de penalidad. Los problemas a considerar en este trabajo son de tipo:

$$\begin{aligned} \min \quad & f(x) \\ \text{s.a} \quad & h_i(x) = 0 \quad \text{para } i = 1,\ldots,p \\ & g_j(x) \le 0 \quad \text{para } j = 1,\ldots,m \\ & lb \le x \le ub \end{aligned}$$

en que $f(x): \mathbb{R}^n \to \mathbb{R}$ es la función objetivo a optimizar, $h(x): \mathbb{R}^n \to \mathbb{R}^p$ es el conjunto de p restricciones de igualdad y $g(x): \mathbb{R}^n \to \mathbb{R}^m$ es el conjunto de m restricciones de desigualdad. Téngase en cuenta, que por lo menos una de estas funciones es no lineal. x son las variables de decisión de dimensión n que están limitadas entre un límite inferior (lb) y un límite superior (ub).

Este problema va a ser transformado en un problema solamente con restricciones de limites simples (de más fácil resolución), adicionando un término de penalidad a la función objetivo. El término de penalidad es tanto mayor cuanto más alejado está de la región admisible y es nulo si el punto estuviera en la región admisible. La función Lagrangeana aumentada puede ser escrita por la siguiente función.

$$L_\mu(x,\lambda,\delta) = f(x) + \sum_{i=1}^{p} \lambda_i h_i(x) + \frac{\mu}{2} \sum_{i=1}^{p} h_i(x)^2 + \frac{1}{2\mu} \sum_{j=1}^{m} \left\{ \left(\max(0, \delta_j + \mu g_j(x))^2 - \delta_j^{\,2} \right) \right\}$$

en que μ es el parámetro de penalidad, λ es el vector de los multiplicadores de Lagrange asociados a las restricciones de igualdad y δ es el vector de los multiplicadores de Lagrange asociados a las restricciones de desigualdad.

El método basado en la Lagrangeana aumentada envuelve la resolución de una secuencia de sub-problemas con límites simples, para una sucesión de valores del parámetro de penalidad [2]. Así, el sub-problema tiene que ser resuelto por el método de penalidad, que es el siguiente:

$$\begin{aligned} &\min_x L_\mu^k(x, \lambda^k, \delta^k) \\ &s.a \quad lb \leq x \leq ub \end{aligned} \tag{2}$$

Por eso, las soluciones de la secuencia de sub-problemas con límites simples convergirán hacia la solución del problema original con restricciones, ya que la función de penalidad incluye términos que aseguran la admisibilidad en el límite [1, 2].

Una de las formas de desarrollar el método de la función Langrangeana aumentada es tener como base el método de los multiplicadores, el cual podemos simplificar de la siguiente forma.

En el proceso de iteraciones tenemos:

- La sucesión $\{\lambda^{(k)}\}$ y $\{\delta^{(k)}\}$ deben de ser limitadas (k es el índice de iteración)
- La sucesión de valores del parámetro de penalidad debe satisfacer:
  $0 < \mu^{(k)} < \mu^{(k+1)}$, para todo k y $\mu^{(k)} \to \infty$

Además de esto mostramos una serie de sugerencias para superar algunas dificultades prácticas:

- La convergencia es más rápida cuando $\lambda^{(k)}$ y $\delta^{(k)}$ son actualizados en cada iteración, que cuando son constantes a lo largo del proceso iterativo (si $\mu^{(k)} \to \infty$ o no).
- Usar métodos del tipo Newton para resolver el problema (2).
- La minimización de $L_\mu^{(k)}(x, \lambda^{(k)}, \delta^{(k)})$ no necesita de ser exacta en algunas variantes solo se implementa una iteración del método de Newton para calcular una aproximación a la solución.
- Usar buenas aproximaciones iniciales.

En referencia al parámetro de penalidad μ marcamos también las siguientes consideraciones a tener en cuenta:

- $\mu^{(1)}$ (inicial) no debe ser muy grande para no originar un problema difícil de resolver o mal condicionado en la primera iteración.
- Aumentar $\mu^{(k)}$ de forma moderada.
  $\mu^{(k)}$ no debe de ser aumentado muy rápido para no originar problemas difíciles de resolver rápidamente y no debe ser aumentado lentamente para no tener una convergencia del multiplicador muy lenta.
- Probablemente $\mu^{(k)}$ deberá quedar $> K$ $(\exists K > 0)$
- La razón de convergencia aumenta a medida que $\mu^{(k)}$ se vuelve mayor.

3.2 Actualización del vector de los multiplicadores

A través de los multiplicadores de Lagrange transformamos un problema con restricciones de igualad, desigualdad o ambos en un problema sin restricciones.

Para tener un buen funcionamiento dentro de nuestro problema con la función Lagrangeana necesitamos definir bien todos los parámetros que intervienen en la formulación del problema, para ello actualizamos el vector de los multiplicadores y el parámetro de penalidad.

Siendo k un índice de iteración, entonces la actualización de los multiplicadores será hecha de la forma siguiente:

- para restricciones de igualdad (*i=1,…,p)*

$$\lambda_i^{(k+1)} = \lambda_i^{(k)} + \mu^{(k)} h_i\left(x^{(k)}\right)$$

- para restricciones de desigualdad (*j=1,…,m)*

$$\delta_j^{(k+1)} = \max\left(0, \delta_j^{(k)} + \mu^{(k)} g_j\left(x^{(k)}\right)\right)$$

- ✓ no es necesario que $\mu^{(k)} \to \infty$. Basta con garantizar$\mu^{(k)} > K$ (para que converja) con
  - $K = \sum_{t=1}^{p} \lambda_t^*$ para restricciones de igualdad
  - $K = \sum_{j=1}^{m} \delta_t^*$ para restricciones de desigualdad

3.3 Actualización del parámetro de penalidad

El algoritmo de la actualización del parámetro de penalidad es el siguiente:

Si $\sum_{i=1}^{p} |h_i(x)| + \sum_{j=1}^{m} \max\left(0, g_j(x)\right) > \varepsilon$

$\mu^{(k+1)} = \pi \mu^{(k)}$

Si no $\mu^{(k+1)} = \mu^{(k)}$,

para $\varepsilon \approx 0$ y los valores de $\pi = 2$ o 10;

3.4 Algoritmo del método de la Lagrangeana aumentada

Ahora declaramos formalmente el algoritmo genérico del método de los multiplicadores basado en la función Lagrangeana aumentada.

Algoritmo

1: dados $x^{(1)}$, $\lambda^{(1)}$, $\delta^{(1)}$, $\mu^{(1)}$, π, $k \leftarrow 1$, $\varepsilon^{(1)}>0$, $\varepsilon^{*}>0$ (≈ 0), μ_{max}, k_{max}

2: calcular $\text{viol}^{(1)} = \sum_{i=1}^{p}\left|h_i\left(x^{(1)}\right)\right| + \sum_{j=1}^{m}\max\left(0, g_j\left(x^{(1)}\right)\right)$

3: **mientras** ($(\text{viol}^{(k)} > \varepsilon^{*}$ y $\mu^{(k)} < \mu_{max})$ y $k < k_{max}$) **hacer**

4: $x^{(k+1)} \approx \min_x L_{\mu^{(k)}}\left(x, \lambda^{(k)}, \delta^{(k)}\right)$ (mediante DIRECT o FMINCON)

5: calcular $\text{viol}^{(k+1)} = \sum_{i=1}^{p}\left|h_i\left(x^{(k+1)}\right)\right| + \sum_{j=1}^{m}\max\left(0, g_j\left(x^{(k+1)}\right)\right)$

6: actualización de los multiplicadores

$$\lambda_i^{(k+1)} = \lambda_i^{(k)} + \mu^{(k)} h_i\left(x^{(k)}\right)$$

$$\delta_j^{(k+1)} = \max\left(0, \delta_j^{(k)} + \mu^{(k)} g_j\left(x^{(k)}\right)\right)$$

7: actualización del parámetro de penalidad

si $\text{viol}^{(k+1)} > \varepsilon^{(k)}$ **entonces**

$\mu^{(k+1)} = \pi\mu^{(k)}$

sino

$\mu^{(k+1)} = \mu^{(k)}$

fin de si

8: $\varepsilon^{(k+1)} = 0.1\ \varepsilon^{(k)}$

9: $k = k + 1$

10: **final mientras**

4 DIRECT

Introducimos en este capítulo algunos conceptos teóricos sobre el algoritmo DIRECT.

DIRECT es un método determinista de optimización global .El algoritmo fue diseñado principalmente para superar algunos de los problemas encontrados en la Optimización Lipschitziana [5, 6, 12].

Comenzamos por discutir los métodos de optimización Lipschtziana, y donde se encuentran estos problemas.

4.1 Optimización de Lipschitz

Recordemos la definición de la continuidad Lipschitz en $\mathbb{R}$'.

Definición 1: Sea $M \subseteq \mathbb{R}$ y $f: M \to \mathbb{R}'$. La función f es llamada Lipschitz continua en M con constante de Lipschitz α si

$$| f(x) - f(x') | \leq \alpha | x - x' | \; \forall x, x' \in M \qquad (3)$$

Si la función f es Lipschitz es continua con constante α, entonces podemos utilizar esta información para la construcción de un algoritmo iterativo para buscar el mínimo de f. El algoritmo de Shubert es una de las aplicaciones más directas de esta idea [10]. Por el momento, supongamos que $\mathrm{M} = [\mathrm{a}, \mathrm{b}] \subset \mathbb{R}'$. De la ecuación (3), es fácil ver que f debe satisfacer las dos desigualdades para cualquier $\mathrm{x} \in \mathrm{M}$.

$$f(x) \geq f(a) - \alpha(x - a)$$

$$f(x) \geq f(b) + \alpha(x - b)$$

Las líneas que corresponden a estas dos desigualdades forman una forma de V por debajo de f, como muestra la Figura 4.1. El punto de intersección de las dos líneas es fácil de calcular, y proporciona la estimación del primer mínimo de f. El algoritmo sigue realizando la misma operación en las regiones $[\mathrm{a}, \mathrm{x}_1]$ y $[\mathrm{x}_1, \mathrm{b}]$ (Figura 4.2a), siguiente divide al lado con el menor valor de la función (Figura 4.2b).

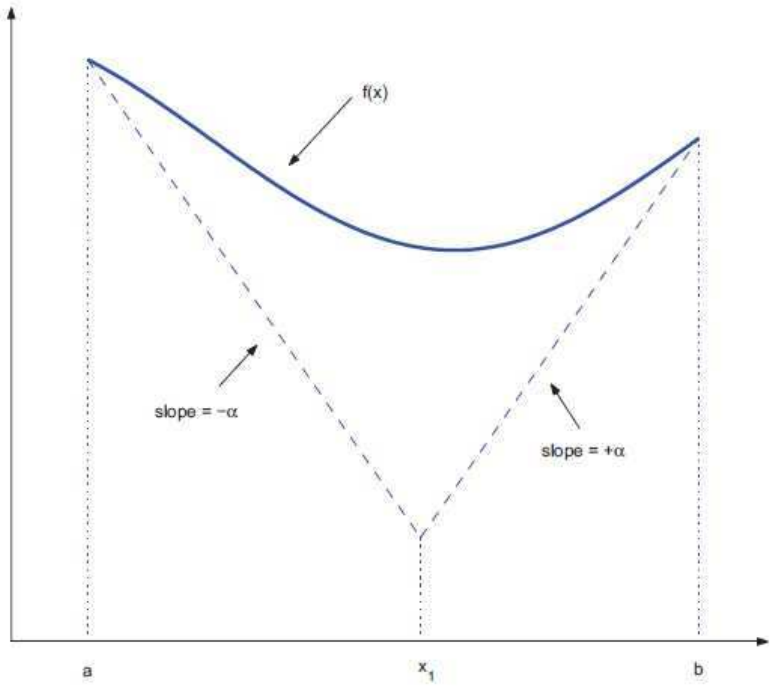


Figura 4.1**: Estimación del primer límite de la función.**

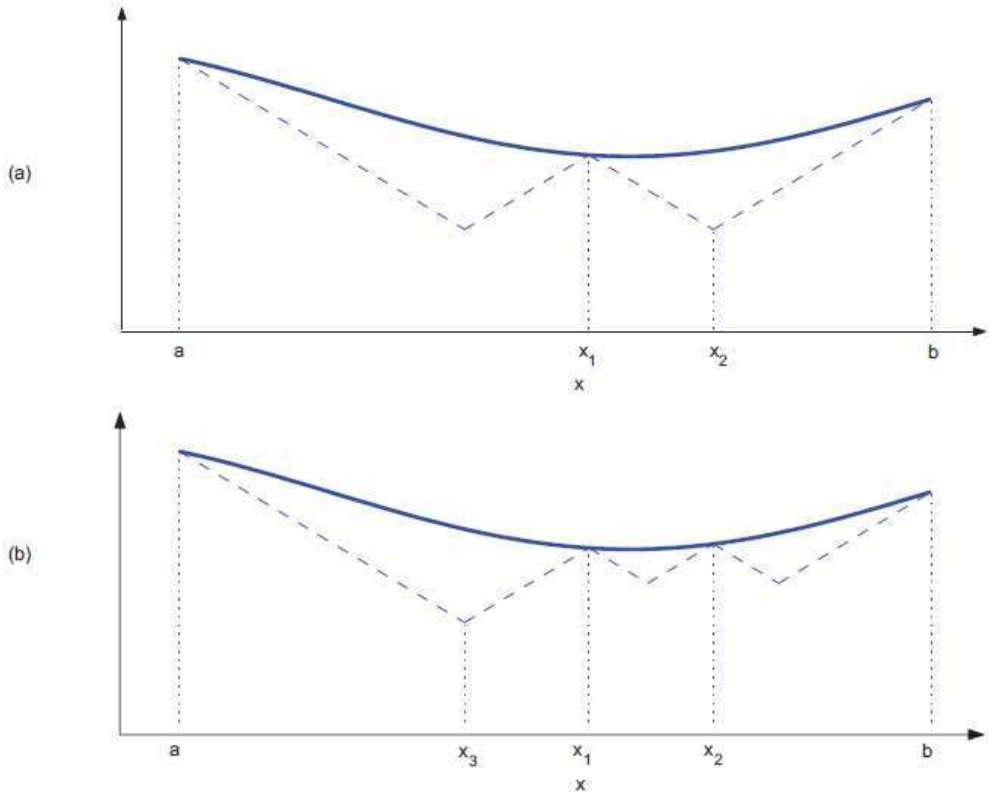


Figura 4.2: Muestra el proceso durante un par de iteraciones.

Existen varios problemas con la utilización de este tipo de algoritmo. Dado que la idea de los extremos no se traduce tan bien para grandes dimensiones, el algoritmo no tiene una interfaz intuitiva generalizada para N> 1. En segundo lugar, la constante de Lipschitz con frecuencia no se puede determinar o estimar de una forma razonable. Muchas simulaciones con aplicaciones industriales pueden incluso no ser constante Lipschitz continua a través de sus dominios. Incluso si la constante de Lipschitz se puede estimar, una mala elección puede llevar a malos resultados.

Si el presupuesto o límite (de la constante de Lipschitz) es demasiado bajo, el resultado no puede tener un mínimo de la función, y si la elección es demasiado grande, la convergencia del algoritmo de será lenta.

La aparición del algoritmo DIRECT fue motivada por estas dos principales desventajas de la optimización Lipschitziana. DIRECT hace muestras directas en los puntos medios de los espacios de búsqueda, eliminando así cualquier confusión de las dimensiones superiores. DIRECT no requiere ningún conocimiento de la constante Lipschitz para la función objetivo. En su lugar, utiliza todos los valores posibles para determinar si una región del dominio debe ser dividida en sub-regiones durante la iteración actual [12].

4.2 Inicialización a DIRECT

DIRECT comienza la optimización mediante la transformación del dominio del problema en la unidad “hiper-cubo”. Es decir:

$$\Omega = \{x \in \mathbb{R}^n : 0 \leq x_i \leq 1\}$$

El algoritmo funciona en este espacio normalizado, y hace referencia al espacio original sólo cuando hacemos referencia (llamadas) a la función. El centro de este espacio es c_1, y comenzamos hallando $f(c_1)$.

Nuestro siguiente paso es dividir este hiper-cubo. Hacemos esto mediante la evaluación de la función en el punto $c \pm \delta e_i, i = 1, \dots, n$ donde δ es un tercio del lado de mayor longitud del hiper-cubo, y e_i es el vector unitario i_{th} (es decir, un vector con un 1 en la posición i y ceros en las demás posiciones).

El algoritmo DIRECT opta por dejar los mejores valores de la función en el espacio más grande; por lo tanto, define:

$$w_i = \min\big(f(c_1 + \delta e_1), f(c_1 - \delta e_i)\big), \qquad 1 \leq i \leq N$$

La dimensión con el menor w_i se divide en tres partes, de modo que $c \pm \delta e_i$ son los centros del nuevo “hiper-rectángulo”. Este patrón se repite para todas las dimensiones "centro hiper-rectángulo ", eligiendo la nueva dimensión mediante la determinación de w_i inmediatamente inferior. La Figura 4.3 muestra este proceso realizado en la función GP (problema Goldstein Price).

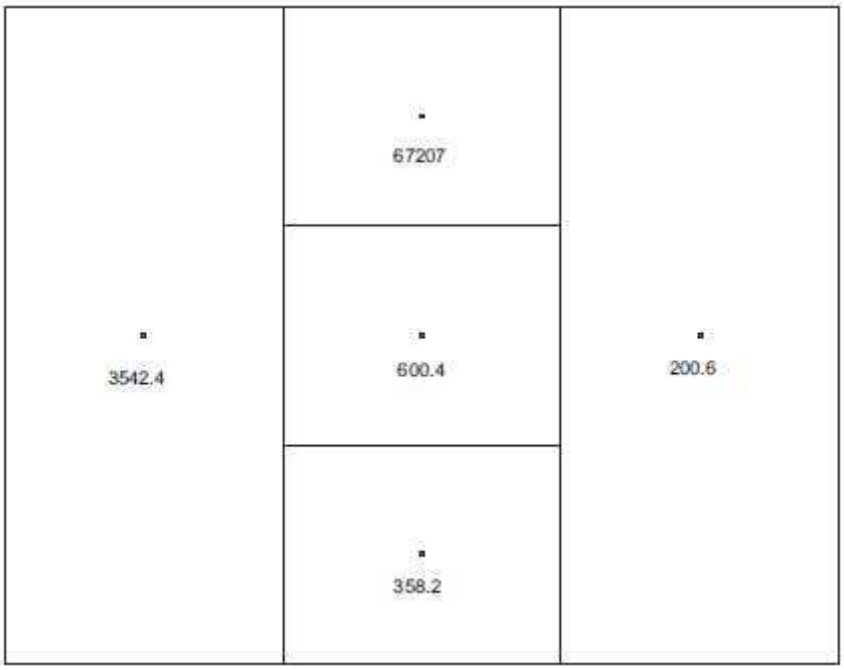


Figura 4.3: Espacio del dominio de la función GP después de la inicialización.

Después, el algoritmo comienza su ciclo de identificar los hiper-rectángulos potencialmente óptimos, dividiendo estos rectángulos adecuadamente, y calcula f en los centros de los nuevos hiper-rectángulos generados.

4.3 Hiper-rectángulos potencialmente óptimos.

En esta sección, se describe lo que el método DIRECT utiliza para determinar qué son los rectángulos potencialmente óptimos, y cuáles deben ser divididos en esta iteración.

DIRECT realiza búsquedas a nivel local y global dividiendo todos los hiper-rectángulos que cumplen los criterios de la siguiente definición.

Definición 2: Sea ε constante positiva y f_{min} sea el corriente mejor valor de la función. Un hiper-rectángulo j se dice que es potencialmente óptima si existe alguna $K > 0$ tal que:

$$f(c_j) - Kd_j \leq f(c_i) - Kd_i, \forall i, \quad \text{y}$$

$$f(c_j) - Kd_j \leq f_{min} - \varepsilon|f_{min}|$$

En esta definición c_j es el centro del hiper-rectángulo j, y d_j es una medida de este hiper-rectángulo. Jones (el creador del algoritmo) optó por utilizar la distancia desde c_j a sus vértices como medida [5].

El parámetro ε se utiliza para que $f(c_j)$ sea superior a nuestra mejor solución actual por una cantidad no trivial. Los datos experimentales se ha demostrado que la condición de $1 \times 10^{-2} \leq \varepsilon \leq 1 \times 10^{-7}$ para el valor tiene un insignificante efecto en los cálculos. Un buen valor de ε es 1×10^{-4}. Este parámetro es uno de los argumentos opcionales que controla el proceso de optimización o también llamado factor Jones.

Podemos hacer algunas observaciones de la Definición 2:

- Si el hiper-rectángulo i es potencialmente óptimo, entonces $f(c_j) \leq f(c_i)$ para todos los hiper-rectángulos que son del mismo tamaño que i (es decir, $d_i = d_j$).

-Si $d_i \geq d_k$ para todos los k hiper-rectángulos, y $f(c_j) \leq f(c_i)$ para todos los hiper-rectángulos tales que $d_i = d_j$, entonces el hiper-rectángulo i es potencialmente óptimo.

- Si $d_i \leq d_k$ para todos los k hiper-rectángulos, e i es potencialmente óptimo, entonces $f(c_i) = f_{min}$.

Para el ejemplo que nos ocupa, sólo un rectángulo es potencialmente óptimo en la primera iteración. La región sombreada de la Figura 4.4a lo identifica. En general, puede haber más de un rectángulo potencialmente óptimo encontrado durante una iteración. Una vez que estos hiper-rectángulos potencialmente óptimos se han identificado, se completa la iteración dividiéndolos [5].

4.4 División de hiper-rectángulos potencialmente óptimos

Una vez que un híper-rectángulo ha sido identificado como potencialmente óptimo, DIRECT divide este hiper-rectángulo en pequeños hiper-rectángulos. Las divisiones están restringidas a hacerlo solamente a lo largo del lado de mayor dimensión del hiper-rectángulo. Esta restricción asegura que los rectángulos se reducirán en cada una de sus dimensiones.

Si el hiper-rectángulo es un hiper-cubo, las divisiones se llevará a cabo a lo largo de todas las partes, como fue el caso de la etapa inicial (inicialización al DIRECT), (comienza la división a lo largo de un lado elegido al azar, puesto que ambos lados tiene la misma dimensión y después lo hace con el rectángulo interior tomando como referencia el lado de mayor dimensión).

La jerarquía para dividir un rectángulo potencialmente óptimo i se determina mediante la evaluación de la función en el punto $c_i \pm \delta_i e_j$, donde e_j es el vector unitario j, y δ_i es un tercio de la longitud del lado máximo de hiper-rectángulo i. La variable j toma en todas las dimensiones la longitud máxima para el hiper-rectángulo i.

Como fue el caso en la fase de inicialización, se define:

$$w_i = \min\{f(c_i + \delta_i e_j), f(c_i - \delta_i e_j)\}, j \in I$$

En la definición anterior, I es el conjunto de todas las dimensiones de la longitud máxima de hiper-rectángulo i. Nuestra primera división se hace en la dimensión con la menor w_i. Direct divide el hiper-rectángulo en 3 hiper-rectángulos a lo largo de dimensión j, de modo que c_i, $c_i + \delta_i e_j$ y $c_i - \delta_i e_j$ son los centros de los nuevos hiper-rectángulos, y son más pequeños. Este proceso se realiza de nuevo en la dimensión de los más pequeños w_i en el nuevo hiper-rectángulo que tiene centro en c_i, y se repite en todas las dimensiones en I [5].

La Figura 4.4 muestra varias iteraciones del algoritmo DIRECT. Cada fila representa una nueva iteración. La transición de la primera columna a la segunda representa el proceso de identificación del hiper-rectángulo potencialmente óptimo. Los rectángulos sombreados en la columna 2 son hiper-rectángulos potencialmente óptimos identificados por DIRECT. La tercera columna muestra el dominio después de que estos rectángulos potencialmente óptimos se han dividido.

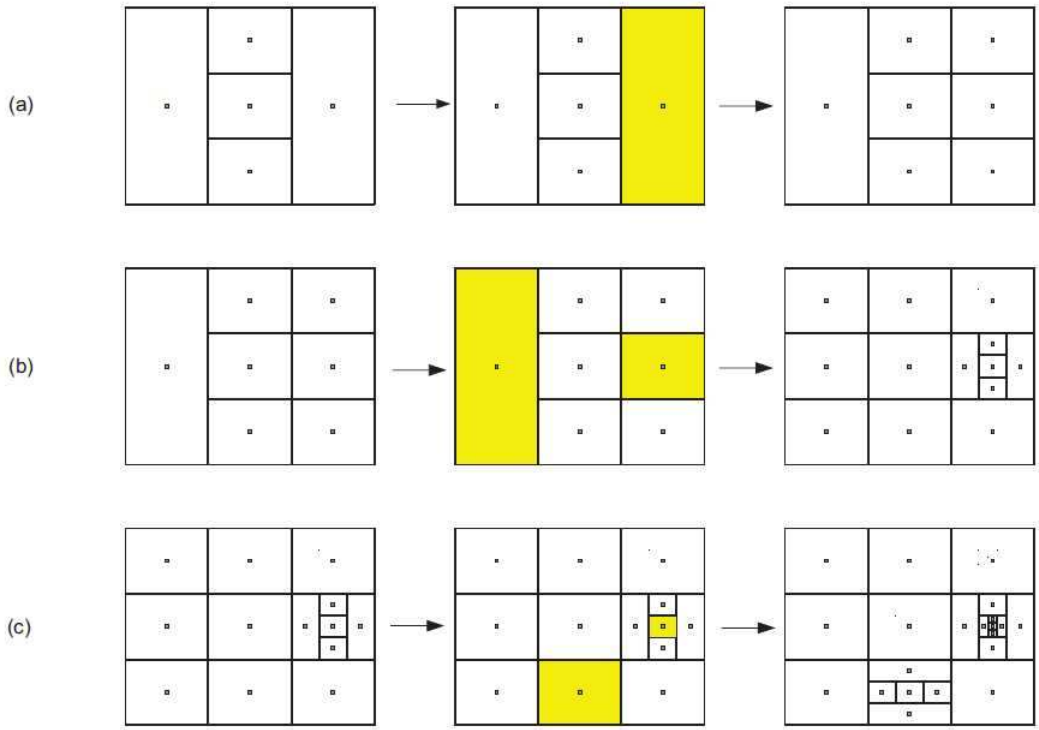


Figura 4.4: Ejemplo de varias iteraciones del DIRECT.

La Figura 4.5 muestra el dominio de la función de GP después de terminado el algoritmo DIRECT. La parada se produce cuando DIRECT llega al 0.01% del valor del mínimo global (valor de la tolerancia de parada, por defecto 0.01 (`options.tol`). DIRECT utiliza 191 evaluaciones (cálculos) de la función.

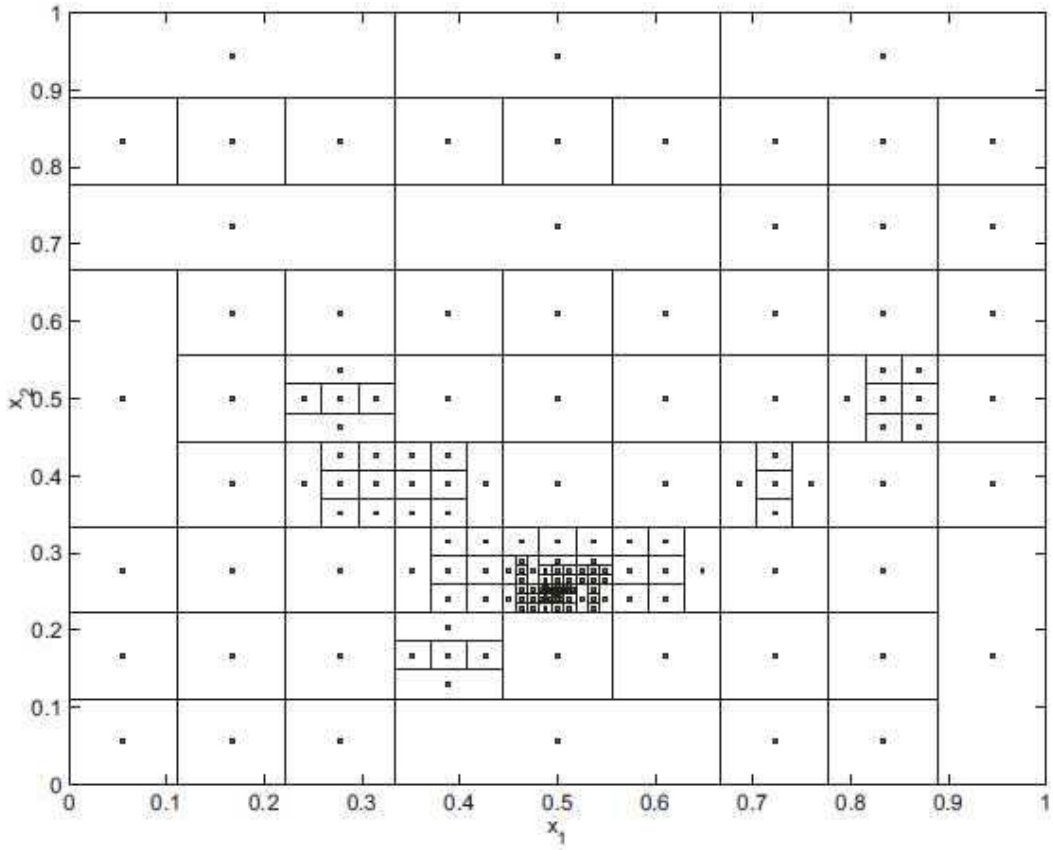


Figura 4.5: Espacio del dominio de la función GP después de 191 iteraciones para llegar al mínimo.

4.5 El Algoritmo DIRECT

Ahora declaramos formalmente el algoritmo DIRECT.

Algoritmo DIRECT ('myfcn', límites, options)

1: Normalizar el dominio en la unidad de hiper-cubo con el centro en c_1
2: Encontrar $f(c_1),\ f(c_1) = f_{min},\ x_{atmin,} = c_1,\ i = 0,$ num_evals $= 1$
3: Evaluar $f(c_1 \pm \delta e_i),\ 1 \leq i \leq n$ y dividir el hiper-cubo
4: **mientras** $i \leq$ max _its y num_evals $\leq$ max _evals) **hacer**
5: Identificar el conjunto S de todos los hiper-rectángulos/cubos potencialmente óptimos
6: **para** todo $j \in S$
7: Identificar el lado más largo del rectángulo j
8: Evaluar myfcn en los centros de los nuevos hiper-rectángulos, y dividir j en rectángulos más pequeños.
9: Actualizar $f_{min,}\ x_{atmin,}\ y$ num_evals
10: **final para**
11: $i = i + 1$
12**: final mientras**

4.6 Invocación del Algoritmo DIRECT

A continuación se muestra como utilizar o comando direct.m, código de MatLab [10], por forma a resolver problemas sin restricciones, solamente con restricciones de limites simples (en el Anexo I, se puede ver una breve explicación de cómo usar DIRECT). Además, DIRECT ha fue extendido para resolver problemas con restricciones de desigualdad a través de la función de penalidad l1, pero aún no fue suficientemente testado [5].

La sintaxis de utilización de DIRECT es:

```
[fmin,xmin,hist] =direct(problem,bounds,options,varargin)
```

en que el argumento de entrada `problem` es una estructura que contiene el problema, que pode ser definida por:

- `Problem.f` - Función objetivo para minimizar (definir en fichero '.m') Si el problema no tiene restricciones este es el único campo que hay que rellenar.
- `Problem.numconstraints` – identifica o Nº de restricciones
- `Problem.constraint(i).func` - restricción i a manejar (definir en fichero '.m')
- `Problem.constraint(i).penalty` – valor inicial del parámetro de penalidad para la restricción i.

`bounds` – es un vector $n \times 2$ vector que describe el dominio (limites das variables). La 1ª columna contiene el límite inferior y la 2ª columna el límite superior.

`options` - argumentos opcionales para controlar el proceso de optimización. Es una estructura que contiene las opciones

`Options` - definición de parámetros	
`Options.ep`	factor Jones. Valor por defecto 0.0001
`Options.maxevals`	nº máximo de evaluaciones de la función. Por defecto 20
`Options.maxist`	nº máximo de iteraciones. Por defecto 10
`Options.maxdeep`	nº máximo de divisiones del rectángulo. Por defecto 100
`Options.testflag`	1 si el mínimo global es conocido, 0 de otro modo. por defecto 0
`Options.globalmin`	valor del mínimo global si se conoce, este parámetro se ignora si `Options.tsetflag` = 0. Por defecto 0
`Options.showits`	1 muestra todas las estadísticas, 0 caso contrario. por defecto 1
`Options.tol`	tolerancia de parado, si `Options.testflag` = 1. Por defecto 0.01
`Options.impcons`	convierte la capacidad de restricción implícita. Por defecto 0, 1 la función objetivo se espera que regrese a la marca que representa la viabilidad del punto de muestreo

`varargin` - (opcional) argumentos adicionales que se pasarán al de la función objetivo.

El comando DIRECT tiene como argumentos de salida:

`fmin` – que es el valor mínimo de la función objetivo que DIRECT puede encontrar

`Xmin` – es la localización del valor mínimo en el dominio, es un argumento de salida opcional.

`hist`- es una matriz de historial de iteraciones que es útil para los gráficos y tablas. La matriz contiene tres columnas: 1. Iteraciones (iter), 2. Valor mínimo encontrado na iteración (f_min), 3. Nº de evaluaciones de la función en la iteración (fn evals).

En el Anexo I, puede encontrar una descripción más detallada del uso del comando DIRECT.

5 EXPERIENCIAS COMPUTACIONALES CON PROBLEMAS CON LÍMITES SIMPLES.

Después de un completo estudio de la teoría y el funcionamiento del algoritmo DIRECT, presentamos una serie de experiencias con dicho algoritmo para dos casos diferentes: problemas con límites simples en las variables (problema Dixon&Szego [3]) y problemas con todo el tipo de restricciones (límites en las variables, igualdad e desigualdad - colección dos problemas g) [7].

Para llevar a cabo todas las experiencias computacionales lo hacemos a través de la implementación en MatLab. El programa MatLab se ha convertido en una de las herramientas más importantes dentro de la computación científica debido a su lenguaje de alto nivel basado en matrices que resuelven los problemas computacionales de una forma más rápida y fácil [10].

A continuación presentamos en un ejemplo cómo resolver un problema con límites simples con DIRECT. Para ello utilizaremos la función CAMEL como función de prueba.

Al final presentamos los resultados obtenidos en la aplicación de DIRECT a los otros problemas de la colección Dixo&Szego [3].

5.1 Ejemplo de aplicación – función Camel

La función CAMEL viene dada por la ecuación:

$$f(x) = \left(4 - 2.1x_1^2 + \frac{1}{3}x_1^4\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$$

El dominio de la función Camel es $-3 \leq x_1 \leq 2$ y $-3 \leq x_2 \leq 2$

Tiene un mínimo global conocido el valor f_{min} = -1.0316284535

Podemos encontrar el mínimo global según las siguientes localizaciones

$$x^*_{min} = \begin{pmatrix} 0.08984201 \\ -0.71265640 \end{pmatrix} \quad \text{y} \quad x^*_{min} = \begin{pmatrix} -0.08984201 \\ 0.71265640 \end{pmatrix}$$

5.1.1 Archivos de resolución

La función es introducida en MATLAB con dos archivos (m.files). En otras palabras, el archivo de la función está presente en la Figura 5.1 y el archivo que permite la invocación del comando DIRECT (script) se puede ver en la Figura 5.2.

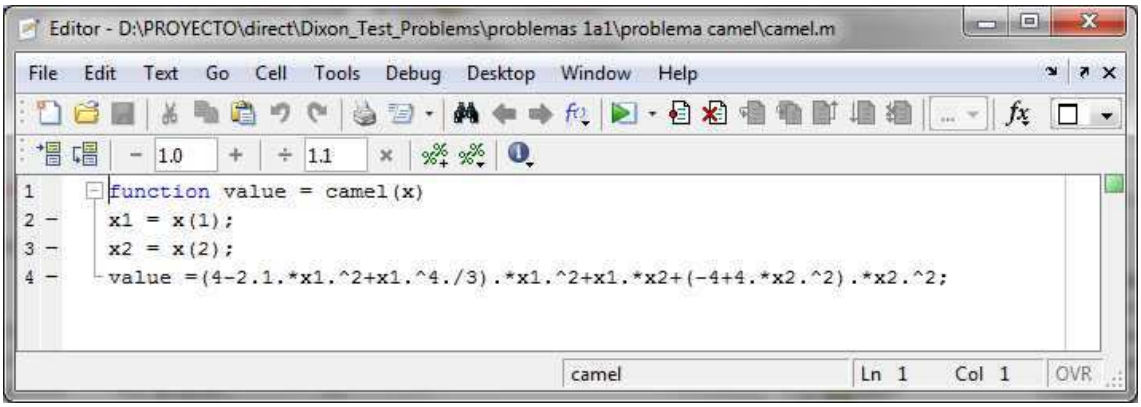


Figura 5.1: Archivo de la función Camel.

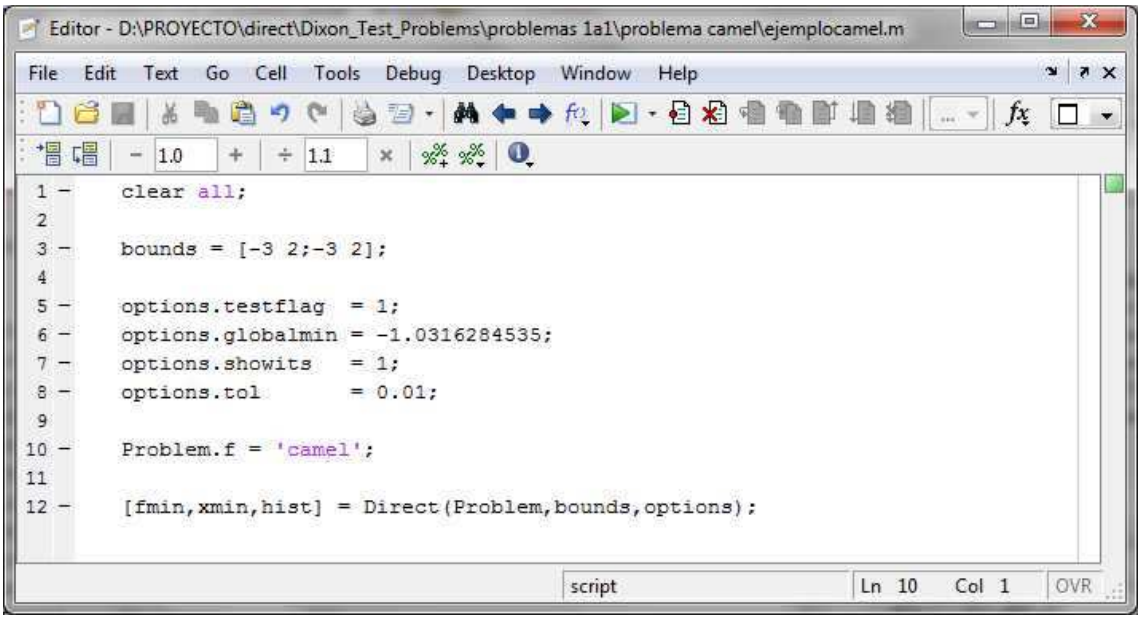


Figura 5.2: Script de invocación.

Dado que `options.testflag` se establece en 1 (que quiere decir que el mínimo global es conocido), DIRECT ignora los límites de evaluaciones de la función e iteraciones con el valor que tienen por defecto. Además completamos con la opción `options.globalmin` que es mínimo real al que deseamos llegar. La opción de `options.showits` con el valor 1 indica que va a mostrar los valores obtenidos en cada iteración.

En su lugar, terminará cuando el error absoluto cometido sea menor que las tolerancias (`options.tol`). En este caso, como `options.tol=0.01`, es decir, terminará una vez se haya llegado a 0.01 del valor del mínimo global indicado en `options.globalmin` ($|f_{min} - f^*| < 0.01$).

5.1.2 Resultados obtenidos

Como resultado de nuestra función obtenemos un historial de iteraciones, el cual veríamos de la siguiente forma:

```
Iter:     1        f_min:     0.3739583333        fn evals:        5
Iter:     2        f_min:    -0.7654520935        fn evals:       11
Iter:     3        f_min:    -0.7654520935        fn evals:       15
Iter:     4        f_min:    -1.0220139696        fn evals:       23
Iter:     5        f_min:    -1.0220139696        fn evals:       31
Iter:     6        f_min:    -1.0220139696        fn evals:       41
Iter:     7        f_min:    -1.0220139696        fn evals:       53
Iter:     8        f_min:    -1.0268688232        fn evals:       65
Iter:     9        f_min:    -1.0306026925        fn evals:       87
Iter:    10        f_min:    -1.0311068407        fn evals:      101
Iter:    11        f_min:    -1.0314454358        fn evals:      121
Iter:    12        f_min:    -1.0316047390        fn evals:      145
Iter:    13        f_min:    -1.0316047390        fn evals:      165
Iter:    14        f_min:    -1.0316047390        fn evals:      185
Iter:    15        f_min:    -1.0316284422        fn evals:      211

fmin =  -1.031628442200385
xmin =    0.089849108367628
         -0.712620027434842
```

Además de la matriz de iteraciones obtenemos también de forma opcional una serie de gráficos que nos muestran diferentes visiones de la función.

Por ejemplo, la Figura 5.3 es un gráfico útil para la evaluación de algoritmos de optimización. En ella podemos ver el valor de la función objetivo decrecer a lo largo del número de evaluaciones de la función. En el eje *x* se representa el número de veces que la función ha sido calculada y en el eje y se representa el valor más pequeño que DIRECT ha encontrado en cada iteración.

Para obtener dicho gráfico tendremos que aumentar nuestro script con la siguiente información:

```
plot(hist(:,2),hist(:,3))
xlabel('Fcn Evals');
ylabel('f_{min}');
title('Iteration Statistics for CAMEL test Function');
```

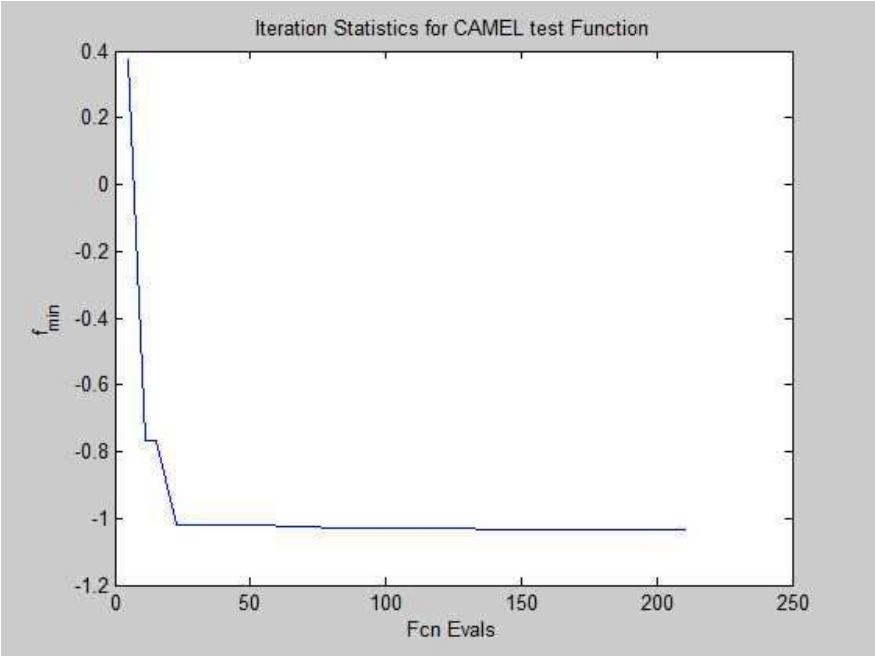


Figura 5.3: Grafico de número funciones evaluadas y valor de la función encontrado.

Además del gráfico anterior completamos la exposición del ejercicio con otro gráfico (Figura 5.4). En él se puede ver de una forma más clara, donde se encuentran los mínimos globales, y se diferencia perfectamente las dos localizaciones de los mínimos globales que posee la función Camel.

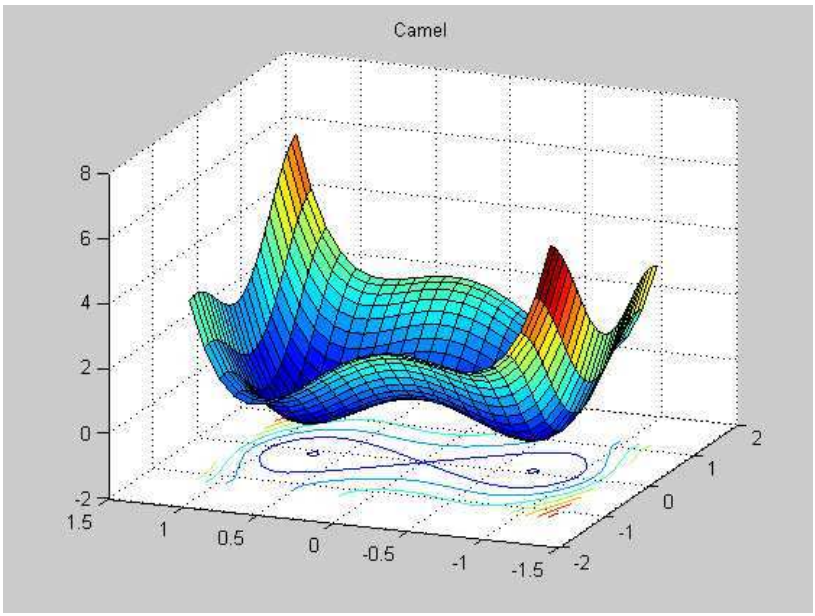


Figura 5.4: Representación grafica de la función Camel.

5.2 Resultados para los problemas Dixon&Szego

Dentro de este apartado mostraremos un resumen de toda la experimentación con DIRECT para problemas. A continuación, se muestra una tabla comparativa de los resultados obtenidos en los diferentes problemas con los que hemos trabajado (problemas Dixon&Szego) [3).

En ella podemos ver como hay problemas que tienen más de una localización de los mínimos globales, lo que DIRECT hace es encontrar una de ellas. También podemos ver como el parámetro options.tol (tolerancia de parado) es el error absoluto cometido en la búsqueda del mínimo global. Con diferentes valores para options.tol comprobamos que el valor del mínimo global se acerca más o menos al valor real del mínimo global a buscar.

En primer lugar vemos los resultados derivados de los problemas Dixon&Szego con una variación en la opción `options.tol` entre ellos. Vamos disminuyendo el valor de dicha opción y obteniendo diferentes resultados.

En la primera fila aparece el nombre del problema Dixon&Szego, la primera sección se muestra los datos de partida del problema y sus características y la segunda los resultados obtenidos.

Opts.tol: Tolerancia para encontrar el valor del mínimo global.

F min global: Valor del mínimo global.

X min global: Localización del mínimizador global.

Nº min globales: Número de mínimos globales que existen en la función.

F min: Valor del mínimo global que encuentra DIRECT.

X min: Localización donde DIRECT encuentra el mínimo global.

Iteraciones: Número de iteraciones realizadas en el proceso de búsqueda.

Nº func evaluadas: Número total de cálculos de la función.

Error relativo: Error relativo existente entre el mínimo encontrado y el real.

	Opts.tol	F min global	X min global	Nº min globales	f min	X min	iteraciones	Nº func evaluadas	Error relativo
GP	$1x10^{-3}$	3	0 -1	1	3.00001003	0 -0.9998475	16	241	$3.343x10^{-6}$
	$1x10^{-4}$				3.00000111	0 -1.0000508	18	305	$3.7\ x10^{-7}$
	$1x10^{-5}$				3.00000012	0 -0.9999830	47	1479	$4\ x10^{-8}$
BRANIN	$1x10^{-3}$	0.397887357	3.14159265 2.27500000	3	0.397891210	3.1424325 2.2736625	15	195	$9.68\ x10^{-6}$
	$1x10^{-4}$		9.42477796 2.47499998		0.397887738	9.4250114 2.4748513	23	377	$9.57\ x10^{-7}$
	$1x10^{-5}$		-3.14159265 12.2750000		0.397887386	-3.1416704 12.275186	52	1295	$7.288\ x10^{-8}$
CAMEL	$1x10^{-3}$	-1.03162845	0.08984201 -0.71265640	2	-1.031628442	0.08984910 -0.71262002	15	211	$-7.754\ x10^{-9}$
	$1x10^{-4}$								
	$1x10^{-6}$				-1.031628450	-0.08983217 0.71263696	1249	44537	0
SHUBERT	$1x10^{-3}$	-186.7309088	-1.42512845 -0.80032121	18	-186.730097	-1.4255956 -0.8000352	140	3143	$-4.34x10^{-6}$
	$1x10^{-4}$				-186.730866	5.4829040 -1.4252569	162	3867	$-2.29\ x10^{-7}$
	$1x10^{-5}$				-186.730898	4.8579992 -0.8003522	500	15915	$-5.78\ x10^{-8}$

	Opts.tol	F min global	X min global	Nº min globales	f min	X min	iteraciones	Nº func evaluadas	Error relativo
SHEKEL5	$1x10^{-3}$	-10.15319967	4 4 4 4	1	-10.1531969	4.0001524157 4.0001524157 4.0001524157 4.0001524157	22	255	$-2.728 x10^{-7}$
	$1x10^{-4}$								
	$1x10^{-5}$				-10.15319902	3.99998306491 4.00015241579 3.99998306491 4.00015241579	655	53525	$-6.402 x10^{-8}$
SHEKEL7	$1x10^{-3}$	-10.40294056	4 4 4 4	1	-10.40286191	4.00015241579 4.00015241579 3.99964436315 4.00015241579	59	1061	$-7.560 x10^{-6}$
	$1x10^{-4}$				-10.40293717	4.00066046842 4.00066046842 3.99964436315 3.99964436315	169	4879	$-3.258 x10^{-7}$
	$1x10^{-5}$				-10.40293954	4.00066046842 4.00066046842 3.99947501227 3.99964436315	1180	38167	$-9.805 x10^{-8}$

	Opts.tol	F min global	X min global	Nº min globales	f min	X min	iteraciones	Nº func evaluadas	Error relativo
SHEKEL 10	$1x10^{-3}$	-10.536409816	4 4 4 4	1	-10.5363294	4.00015241579 4.00015241579 4.00015241579 3.99964436315	61	1131	$-7.631 x10^{-6}$
	$1x10^{-4}$				-10.5364067	4.0006604684 4.0006604684 3.9996443631 3.9996443631	169	4939	$-2.951 x10^{-7}$
	$1x10^{-5}$	No para			-	-	-	-	-
HART 3	$1x10^{-3}$	-3.8627821478	0.1 0.55592003 0.85218259	1	-3.86277067	0.10905349794 0.55624142661 0.85253772290	33	695	$-2.971 x10^{-6}$
	$1x10^{-4}$				-3.86277871	0.11042524005 0.55624142661 0.85253772290	35	751	$-8.897 x10^{-7}$
	$1x10^{-5}$				-3.86278793	0.11316872427 0.55624142661 0.85253772290	36	775	$1.497 x10^{-6}$
HART 6	$1x10^{-3}$	-3.3223680114	0.20168952 0.150010 0.47687398 0.27533243 0.31165162 0.65730053	1	-3.32234500	0.202331961591 0.150205761316 0.476680384087 0.275034293552 0.312071330589 0.657750342935	32	1031	$-6.926 x10^{-6}$
	$1x10^{-4}$	No para			-	-	-	-	-
	$1x10^{-5}$	No para			-	-	-	-	-

En la tabla podemos apreciar que en el apartado option.tol existen unos valores en rojo. Esto se debe a que en ese punto la solución es totalmente igual que la anterior. Con la primera tolerancia hace una búsqueda tan exhaustiva que rebasa el límite de la segunda, por lo tanto aunque dicha tolerancia sea decrecida en el siguiente paso el resultado al que llega es igual al primero. Apreciamos también que obviamente cuanto menor es la tolerancia aplicada, mejor es la solución obtenida y por lo tanto menor es nuestro error relativo

Además de esto vemos la frase “no para” también en rojo en alguno de los procesos. Debido a que para ese determinado caso la tolerancia es mucho reducida, DIRECT no llega a una solución en un tiempo de espera prudencial. .

6 EXPERIENCIAS COMPUTACIONALES CON PROBLEMAS CON TODAS LAS RESTRICCIONES

Dentro de las experiencias realizadas en los problemas con restricciones se ofrece una comparativa de aplicación del método de la Lagrangeana aumentada a través del algoritmo DIRECT y también del comando FMINCON de MatLab para resolver los sub-problemas.

6.1 Comando FMINCON

FMINCON es un comando de MatLab perteneciente al "toolbox" de optimización y el cual tiene implementado cuatro diferentes algoritmos como métodos de resolución: *trust-region-reflective*, *active-set*, *interior point* y *sequential quadratic programming* (SQP) [10].

Para escoger un algoritmo, se procede de la siguiente forma, modificando las opciones de FMINCON: por ejemplo, para escoger SQP, las opciones son: `opciones=optimset('Algorithm','sqp').`

FMINCON es el comando que vamos a utilizar para la resolución de los problemas g además de utilizar también DIRECT. La principal diferencia entre los dos algoritmos es que DIRECT es un método e optimización global y FMINCON es un método e optimización local.

FMINCON encuentra el mínimo valor de una función de varias variables y permite resolver problemas con todo el tipo de restricciones. Podemos ver cómo resolverlos a través de una breve explicación presentada en el Anexo II.

6.2 Colección de problemas g

Fue seleccionada dentro de la colección de problemas g [7] algunos de ellos para mostrar más detalladamente, abarcando así las diferentes posibilidades de problemas con restricciones.

A continuación presentamos una lista de problemas g cuyas dimensiones varían entre 2 y 10, así como el número de restricciones de igualdad y desigualdad.

Nombre	Tipo de problema	Número de variables	Nº Restricciones desigualdad	Nº Restricciones igualdad
g04	Cuadrático	5	6	0
g05	Cúbico	4	2	3
g06	Cúbico	2	2	0
g07	Cuadrático	10	8	0
g09	Polinomial	7	4	0
g11	Cuadrático	2	0	1
g12	Cuadrático	3	1	0
g15	Cuadrático	3	0	2

Para demonstrar cómo fue implementado el método de los multiplicadores basado en la función Lagrangeana aumentada, tomaremos como ejemplo un problema con restricciones de igualdad (g11), otro con restricciones de desigualdad (g06), y un tercero con restricciones tanto de igualdad como desigualdad (g05). La resolución de los problemas lo haremos a través de MatLab, escribiendo todo el problema en archivos m.files.

Para definir el problema completo será necesario crear una serie de archivos en los cuales desarrollaremos todo el problema, para de esta manera, poder resolverlo tanto con el algoritmo DIRECT, como con FMINCON ayudándonos además de la función Lagrangeana. Los archivos necesarios serán los siguientes:

- Archivo función objetivo: En él se escribirá la función objetivo, es decir, la función que queremos minimizar.
- Archivo call_FUN_direct (ver Figura 5.5): este archivo contiene una función adicional que necesita el algoritmo DIRECT para pasar el nombre de la función objetivo y sus parámetros, que son los multiplicadores, porque se trata de un problema con restricciones. Este archivo será común para todos los problemas g.
- Archivo de las restricciones: El contenido de este archivo serán las restricciones a las que está sujeta nuestra función objetivo.
- Archivo de la violación: En este archivo incluimos la violación necesaria para convertir nuestro problema de restricciones en uno sin ellas, como anteriormente explicamos en el capítulo 2, dentro del apartado método de penalidad
- Archivo script: En él, definiremos el algoritmo por completo.

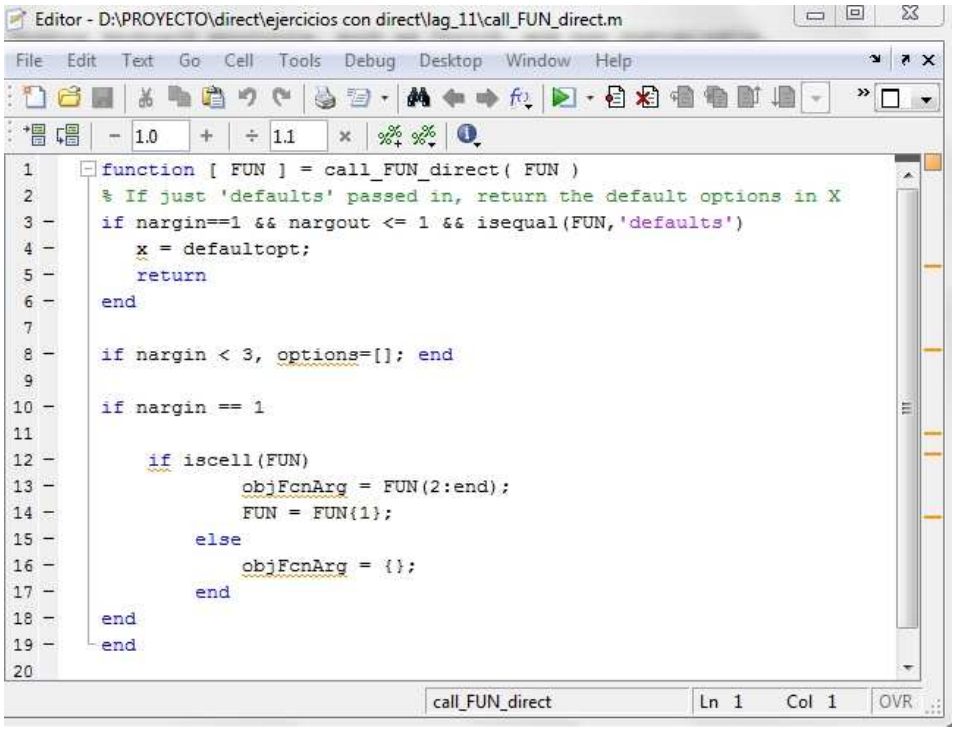


Figura 6.1: Archivo para pasar el nombre de la función objetivo y parámetros.

6.3 Ejemplo de problema con restricciones de igualdad

Tal como fue dicho anteriormente, fue seleccionado el problema g11 para presentar el desarrollo del algoritmo cuando es aplicado a un problema con apenas restricciones de igualdad.

- Definición del Problema g11

Minimizar la función: $f(x) = x_1^2 + (x_2 - 1)^2$

Sujeto a: $h(x) = x_2 - x_1^2 = 0$

Límites: $-1 \leq x_1 \leq 1$ y $-1 \leq x_2 \leq 1$

Solución: $f(x^*) = 0.7499$

- Introducción del problema en MatLab.

El problema g11 es introducido en MatLab con los archivos (m.files) presentados en las Figuras 6.2-6.4

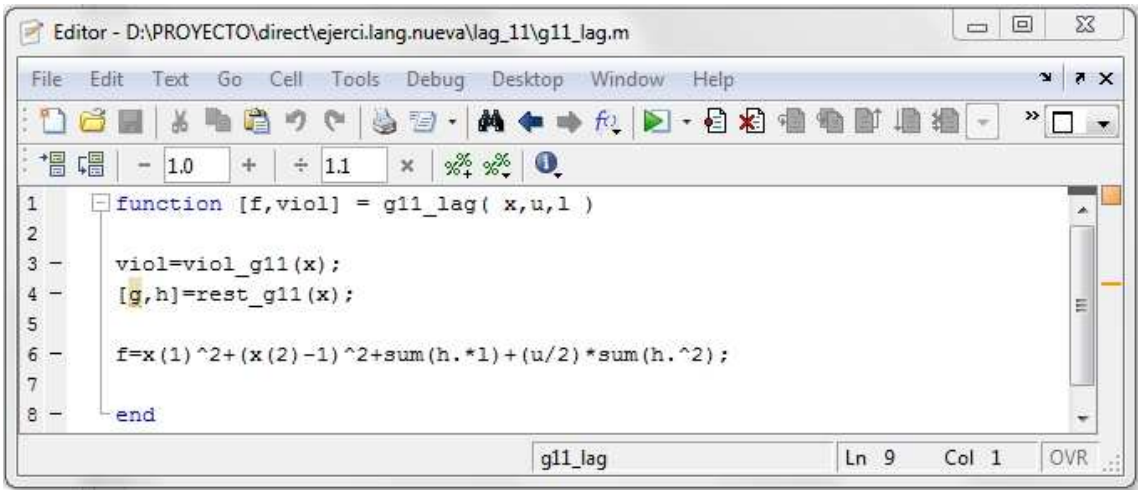


Figura 6.2: Archivo de la función Lagrangeana aumentada del problema g11.

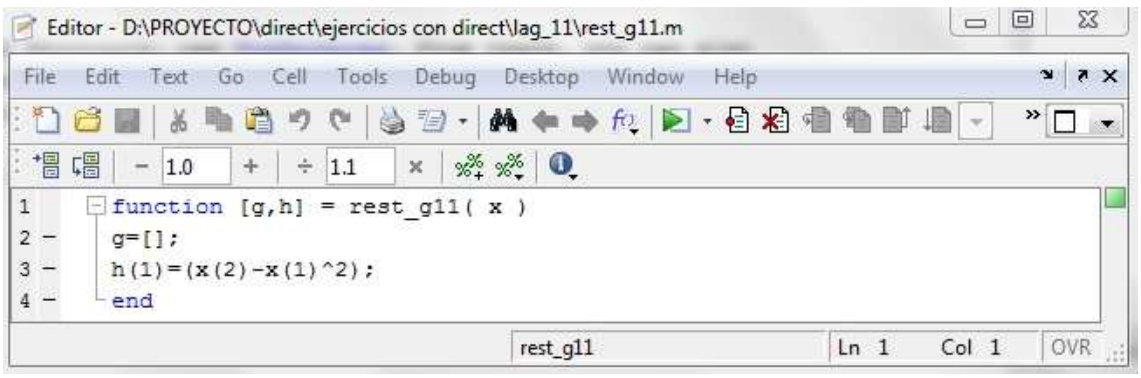


Figura 6.3: Archivo de las restricciones del problema g11.

Figura 6.4: Archivo que mide la violación de las restricciones del problema g11.

La script que implementa el método para el problema g11 contiene los siguientes comandos:

```
clear all
clc

%definir problema
n=2;
m=1;
j=(1:m);
i=(1:n);

x(i)=0.1;
bounds=[-1 1;-1 1];
LB(i)=bounds(:,1);
UB(i)=bounds(:,2);


it=0;
fun_evals=0;
total_it=0;
l(j)=1;
u=1;
viol=viol_g11(x);

%valores do algoritmo
max_u=1e10;
eta_ast=1e-8;
eta_k=1;
it_max=100;
aumenta=2;

%OPTIONS DIRECT
options.testflag  = 0;   options.tol       = 1e-5; options.showits=0;
options.maxevals = 1e4; options.maxits = 1e4;

%OPTIONS FMINCON
op=optimset('algorithm','sqp');

while (( viol > eta_ast   &&    u < max_u) && it < it_max)

%    CALLING FMINCON
     [x,fval,exitflag,output]=…
     fmincon(@(x)g11_lag(x,u,l),x,[],[],[],[],LB,UB,[],op);
     fun_evals=fun_evals+output.funcCount;
     total_it=total_it+output.iterations;

%    CALLING DIRECT%
%    FUN=call_FUN_direct(@(x) g11_lag(x,u,l));
```

```
%    Problem.f = FUN;
%    [fval, x, hist] = Direct(Problem, bounds,options);
%    total_it=total_it+hist(length(hist),1);
%    fun_evals=fun_evals+hist(length(hist),2);

     viol=sum(viol_g11(x));
     [desig,ig]=rest_g11(x);


%    UPDATE MULTIPLIERS

     l(j)=l(j)+u.*ig(j);
%    d(j)=max(0,d(j)+u.*desig(j));

%    UPDATE PENALTY PARAMETER
     if viol > eta_k
         u=aumenta*u;
     end

     eta_k=0.1*eta_k;
     it=it+1;
end
fval, it, total_it, fun_evals
violacion_final = viol
```

Resultados obtenidos para el problema g11

- Para DIRECT

  Valor final de la función encontrado (mínimo) fval = 0.7500

  Número de iteraciones realizadas hasta llegar al mínimo it = 23.

  Valor de la violación violacion_final = 3.330 e-016.

  Número total de cálculos de la función fun_evals = 230731

  Número total de iteraciones realizadas hasta llegar al mínimo total_it =4653

- Para FMINCON

  Valor final de la función encontrado (mínimo) fval = 0.75

  Número de iteraciones realizadas hasta llegar al mínimo it = 42

  Valor de la violación violacion_final = 1.43122 e-008

  Número total de cálculos de la función fun_evals = 914

  Número total de iteraciones realizadas hasta llegar al mínimo total_it = 41

6.4 Ejemplo de problema con restricciones de desigualdad

Definimos el Problema g06

Minimizar la función: $f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$

Sujeto a: $g_1(x) = -(x_2 - 5)^3 + (x_2 - 5)^2 + 100 \leq 0$
$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$

Límites: $13 \leq x_1 \leq 100$ y $0 \leq x_2 \leq 100$

Solución: $f(x^*) = -6961.81387558015$

- Introducción del problema en MatLab.

Tal como para el problema g11, el problema g06 es introducido en MatLab con los archivos en las Figuras 6.5-6.7.

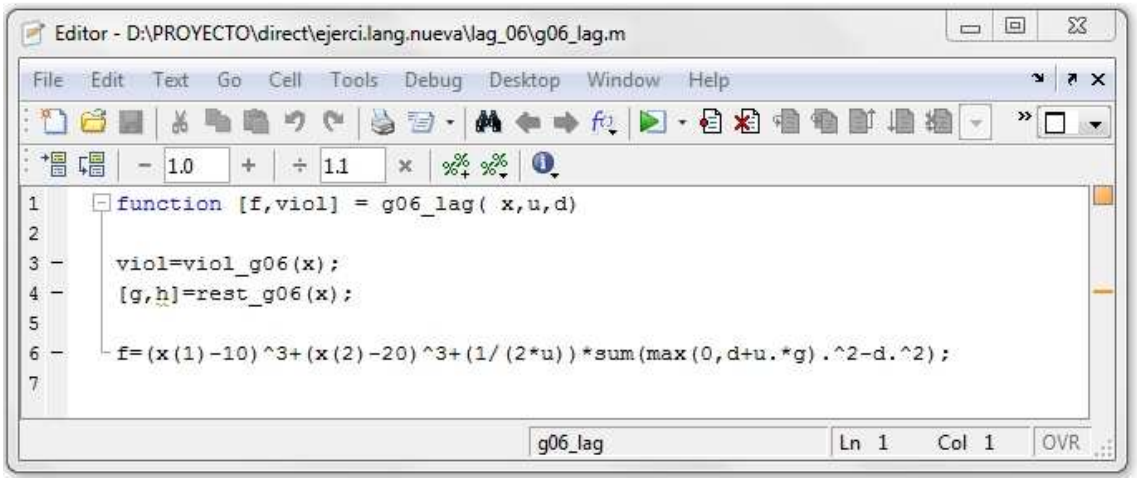


Figura 6.5: Archivo de la función Lagrangeana aumentada del problema g06.

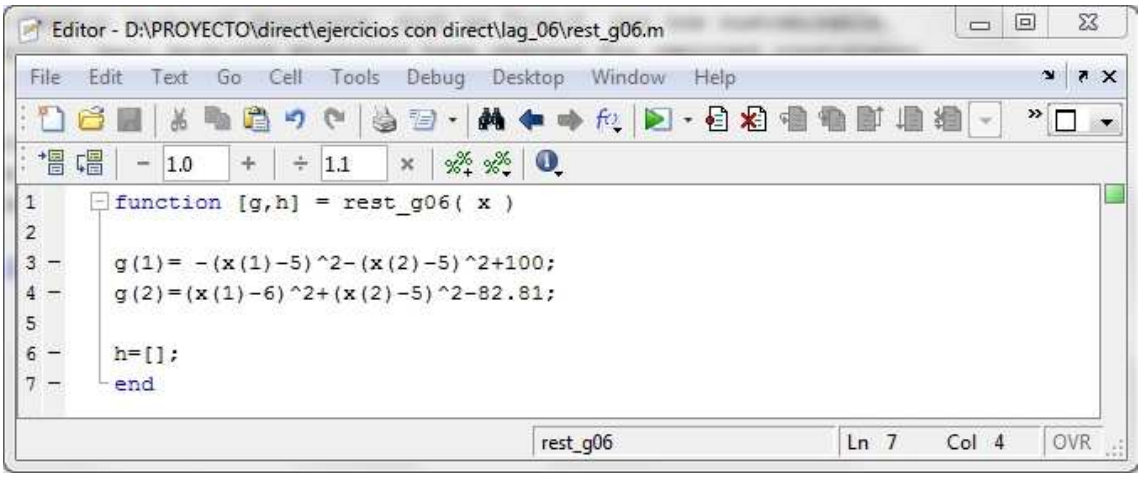


Figura 6.6: Archivo de las restricciones del problema g06.

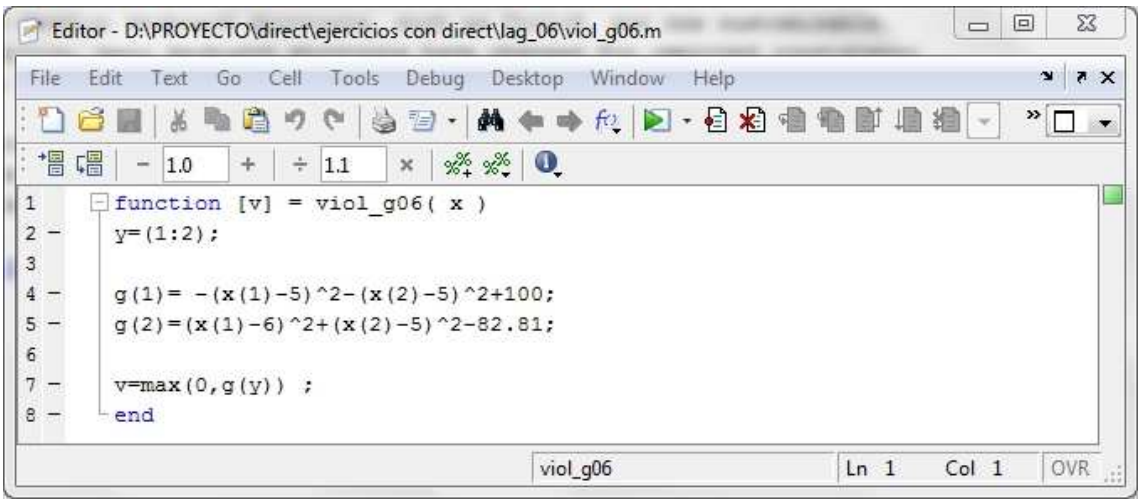


Figura 6.7: Archivo de la violación de las restricciones del problema g06.

Archivo del script:

```
clear all
clc

%definir problema
n=2;
m=2;
j=(1:m);
i=(1:n);

x(i)=1;
bounds=[13 100;0 100];
LB(i)=bounds(:,1);
UB(i)=bounds(:,2);

fun_evals=0;
total_it=0;
it=0;
d(j)=0;
u=1;
viol=sum(viol_g06(x));

%valores do algoritmo
max_u=1e10;
eta_ast=1e-8;
eta_k=1;
it_max=100;
aumenta=2;

%options direct
options.testflag  = 0;  options.showits=0; options.tol=1e-5;
options.maxevals=1e4;options.maxits=1e3;

%options fmincon
op=optimset('algorithm','sqp');

while (( viol > eta_ast   &&    u < max_u) && it < it_max)

%     CALLING FMINCON
%    [x,fval,exitflag,output]=…
%    fmincon(@(x)g06_lag(x,u,d),x,[],[],[],[],LB,UB,[],op);
%    fun_evals=fun_evals+output.funcCount;
%    total_it=total_it+output.iterations;
%
%  CALLING DIRECT
 FUN=call_FUN_direct(@(x) g06_lag(x,u,d));
 Problem.f = FUN;
 [fval, x,hist] = Direct(Problem, bounds,options);
 total_it=total_it+hist(length(hist),1);
 fun_evals=fun_evals+hist(length(hist),2);

   viol=sum(viol_g06(x));
   [desig,ig]=rest_g06(x);

%  UPDATE MULTIPLIERS
% l(j)=l(j)+u.*ig(j);
  d(j)=max(0,d(j)+u.*desig(j));
  %    UPDATE PENALTY PARAMETER

   if viol>eta_k
       u=aumenta*u;
   end
   eta_k=0.1*eta_k;
   it=it+1;
end

fval, it, total_it, fun_evals
violacao_final=viol
```

Resultados:

- Para DIRECT

  Valor final de la función encontrado (mínimo) fval = -6882.98

  Número de iteraciones realizadas hasta llegar al mínimo it = 21

  Valor de la violación violacion_final = 0

  Número total de cálculos de la función fun_evals = 210563

  Número total de iteraciones realizadas hasta llegar al mínimo total_it =4746

- Para FMINCON

  Valor final de la función encontrado (mínimo) fval = -6940.51

  Número de iteraciones realizadas hasta llegar al mínimo it = 34

  Valor de la violación violacion_final = 5.7507 e-5

  Número total de cálculos de la función fun_evals = 1470

  Número total de iteraciones realizadas hasta llegar al mínimo total_it =71

6.5 Problemas con restricciones de igualdad y desigualdad:

Definimos el Problema g05:

Minimizar la función: $f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$

Sujeto a: $h_1(x) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$

$h_2(x) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$

$h_3(x) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$

$g_4(x) = -x_4 + x_3 - 0.55 \leq 0$

$g_5(x) = -x_3 + x_4 - 0.55 \leq 0$

Límites: $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$, $-0.55 \leq x_4 \leq 0.55$

Solución: $f(x^*) = 5126.4967140071$

- Introducción del problema en MatLab.

El problema g05 es introducido en MatLab con los archivos presentados en las Figuras 6.8-6.10.

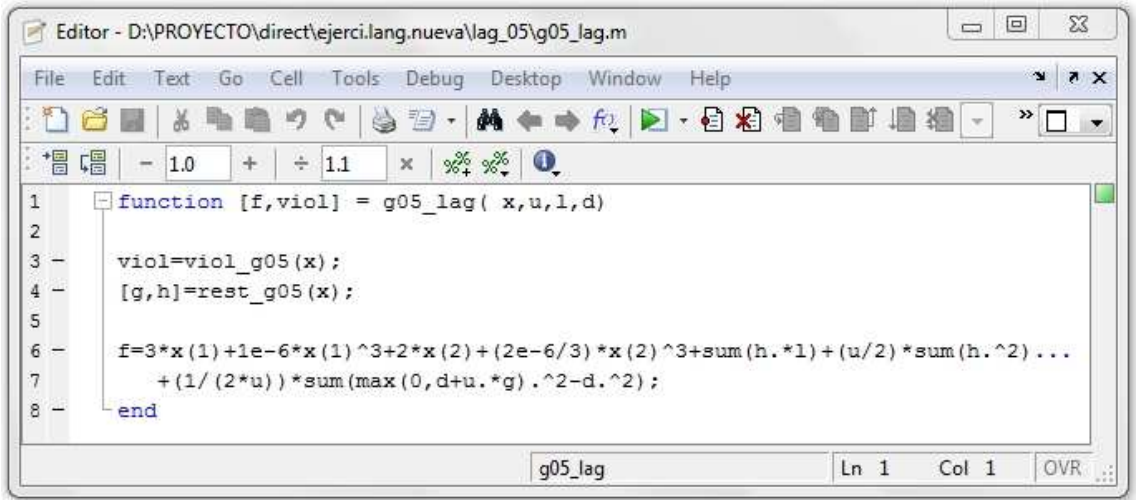


Figura 6.8: Archivo de la función Lagrangeana aumentada del problema g05.

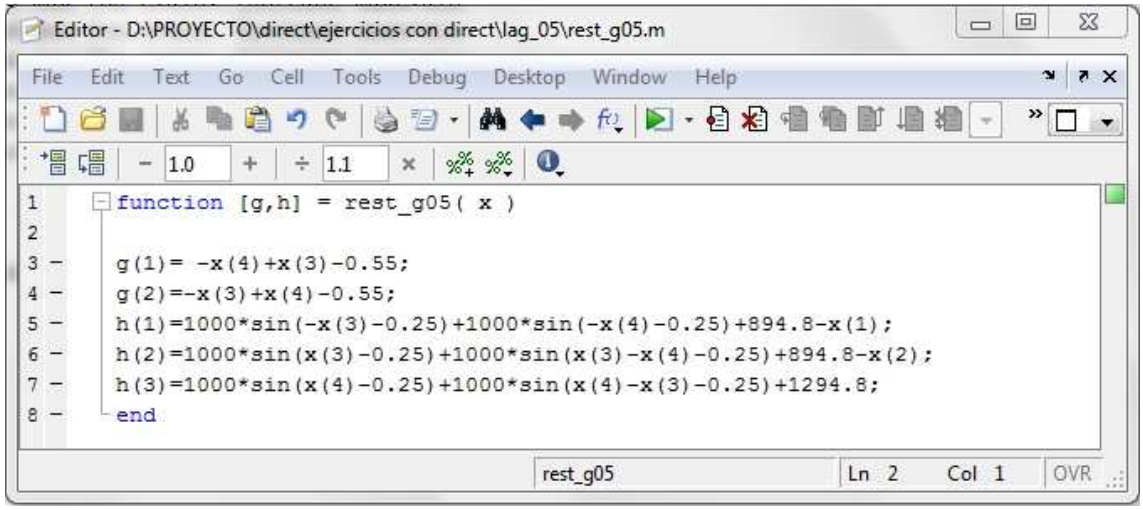


Figura 6.9: Archivo de las restricciones del problema g05.

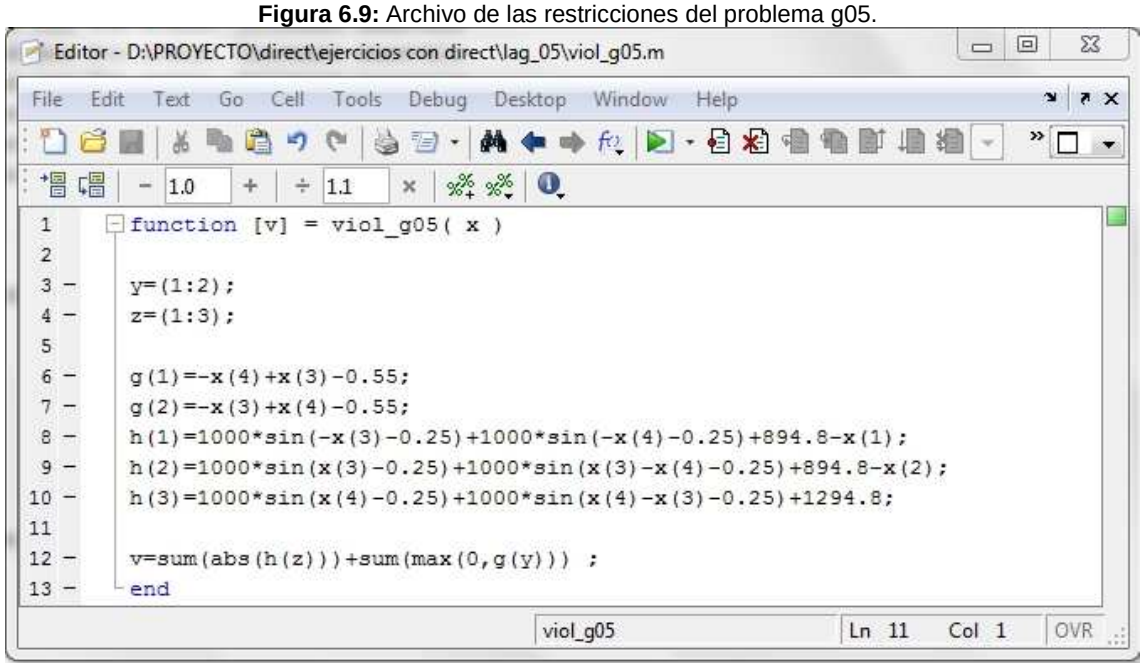


Figura 6.10: Archivo de la violación de las restricciones de g05.

Archivo del script

```
%definir problema
n=4;
m=3;
p=2;
j=(1:m);
s=(1:p);
i=(1:n);

x(i)=0.1;
bounds=[0 1200;0 1200;-0.55 0.55; -0.55 0.55];
LB=bounds(:,1);
UB=bounds(:,2);

it=0;
fun_evals=0;
total_it=0;
l(j)=0;
d(s)=0;
u=1;
viol=(viol_g05(x));

%valores do algoritmo
max_u=1e10;
eta_ast=1e-8;
eta_k=1;
it_max=100;
aumenta=2;

%option direct
options.tol = 1e-5; options.showits=0;
options.maxevals = 1e4; options.maxits = 1e3;

%option fmincon
op=optimset('algorithm','sqp');

while (( viol > eta_ast   &&    u < max_u) && it < it_max)

 % CALLING FMINCON
%    [x,fval,exitflag,output]=…
%    fmincon(@(x)g05_lag(x,u,l,d),x,[],[],[],[],LB,UB,[],op);
%    fun_evals=fun_evals+output.funcCount;
%    total_it=total_it+output.iterations;

%   CALLING DIRECT
 FUN=call_FUN_direct(@(x) g05_lag(x,u,l,d));
 Problem.f = FUN;
 [fval, x,hist] = Direct(Problem, bounds,options);
 total_it=total_it+hist(length(hist),1);
 fun_evals=fun_evals+hist(length(hist),2);

 viol=(viol_g05(x));
 [desig,ig]=rest_g05(x);

 %UPDATE MULTIPLIERS
 l(j)=l(j)+u.*ig(j);
 d(s)=max(0,d(s)+u.*desig(s));

%    UPDATE PENALTY PARAMETER
 if viol>eta_k
     u=aumenta*u;
 end
 eta_k=0.1*eta_k;
 it=it+1;
end
 fval, it, total_it, fun_evals
 violacao_final=viol
```

Los resultados con DIRECT y FMINCON en la resolución de los sub-problemas del método de los multiplicadores es el siguiente:

- Para DIRECT

  Valor final de la función encontrado (mínimo) fval = 5126.62

  Número de iteraciones realizadas hasta llegar al mínimo it = 34

  Valor de la violación violacion_final = 1.1040 e-6

  Número total de cálculos de la función fun_evals = 341758

  Número total de iteraciones realizadas hasta llegar al mínimo total_it =5225

- Para FMINCON

  Valor final de la función encontrado (mínimo) fval = 5395.64

  Número de iteraciones realizadas hasta llegar al mínimo it = 37

  Valor de la violación violacion_final = 2.381 e-4

  Número total de cálculos de la función fun_evals = 1767

  Número total de iteraciones realizadas hasta llegar al mínimo total_it = 70

Como vemos en los tres ejercicios de muestra propuestos la principal diferencia es la definición en el script de las diferentes variables para las restricciones. Para las restricciones de igualdad tomamos el vector l para representar el multiplicador asociado a esas restricciones, mientras si las restricciones son de desigualdad tomamos el vector d. Otra de las principales diferencias entre DIRECT y FMINCON es que FMINCON necesita que sea definido un punto inicial por el cual comenzará la búsqueda (x_i). Después, en cada iteración, lo punto inicial es lo que salió da optimización anterior e que fue utilizado para actualizar o parámetro de penalidad e los multiplicadores.

Cabe destacar también que en el script definimos la forma para resolverlo por DIRECT y por FMINCON, activando o desactivando cada una de ellas cuando se esté utilizando la otra. De la misma forma de activación o desactivación se procede con la actualización de los multiplicadores (update multipliers) para los diferentes tipos de restricciones utilizadas.

6.6 Presentación de los resultados de los problemas g

Dentro de este apartado tenemos los resultados de los problemas g con una comparativa entre el funcionamiento del algoritmo DIRECT y el del comando FMINCON, además de realizar una serie de variaciones dentro de las tolerancias, opciones, punto inicial, variación del valor de comienzo, de los multiplicadores pertenecientes a la función Lagrangeana etc.

Para llegar a la solución final en los problemas g teniendo en cuenta el algoritmo DIRECT se hace de forma diferente que para los problemas Dixon, puesto que en esta buscamos el mínimo global sin activar las opciones `options.testflag` y `options.globalmin`, es decir, sin decirle al algoritmo que tiene un mínimo y el valor de dicho mínimo al que debe llegar.

Los valores iniciales usados en el algoritmo han sido los siguientes:

- $x^{(1)}= 0.1$
- $\lambda^{(1)}= 0$
- $\delta^{(1)}=0$
- $\mu^{(1)}=1$
- $\mu_{max} = 1 \times 10^{10}$
- $k_{max} = 100$
- $\varepsilon^{(1)}=1$
- $\varepsilon^{*}= 1 \times 10^{-8}$
- $\pi = 2$

En la tabla siguiente se presentan los resultados de los problemas g con los que hemos trabajado, obtenidos con el método de la función Lagrangeana aumentada. Los valores de entrada de la tabla son los siguientes:

f opt: Es el valor óptimo conocido de la función

Solver: Nombre de método utilizado en la minimización de los sub-problemas.

fval: Resultado al que llegan nuestros algoritmos de prueba.

Viol: Valor final calculado de la violación en nuestro problema.

It_penal.: Numero de iteraciones para cada método dos multiplicadores.

It_total: Numero de iteraciones totales llevadas a cabo en el proceso.

Total fun_evals: Número de cálculos totales de la función.

Prob	f opt	Solver	Fval	Viol	It_penal	It_total	Total fun_evals
g04	-30665.53867	DIRECT	-30663.71	0	16	1387	165140
		FMINCON	-30665.54	0	23	56	474
g05	5126.496714	DIRECT	5126.62	1.104e-6	34	5225	341758
		FMINCON	5395.64	2.381e-4	37	70	1767
g06	-6961.813875	DIRECT	-6882.98	0	21	4746	210563
		FMINCON	-6940.51	5.751e-5	34	71	1470
g07	24.30620907	DIRECT	25.1598	0	3	281	30251
		FMINCON	24.3062	0	10	150	2659
g09	680.6300574	DIRECT	682.943	0	2	275	20158
		FMINCON	679.842	0	2	52	554
g11	0.7499	DIRECT	0.75	3.33e-16	23	4653	230731
		FMINCON	0.75	1.431e-8	42	41	914
g12	-1	DIRECT	-1	0	1	27	10365
		FMINCON	-0.97803	0	9	33	175
g15	961.7150223	DIRECT	963.3991	6.103e-7	35	7198	351089
		FMINCON	961.8582	7.068e-7	40	76	1568

Se puede concluir que en general en el método de los multiplicadores cuando es aplicado en DIRECT para resolver los sub-problemas basados en la función Lagrangeana aumentada son mucho más próximos a la solución óptima conocida. Cabe destacar que para el problema g05, que contiene restricciones de igualdad y de desigualdad, el método presenta mejores resultados que con FMINCON. Es de notar que FMINCON no consigue calcular óptimos globales, una vez que es un método de optimización local. Pero a veces puede conseguirlo, como en este caso, si manejamos los valores de aproximación inicial.

Pensamos que se pueden encontrar mejores resultados si se hiciese un análisis de sensibilidad a ciertos valores iniciales usados, es decir a los valores de los multiplicadores. También se pueden usar otros valores para el parámetro de penalidad así cómo usar otras formas de actualización.

7 CONCLUSIONES

El objetivo de este trabajo fue desarrollar la comprensión del algoritmo DIRECT. DIRECT es un algoritmo de optimización derivado, que busca mínimos globales de una función de valor real en un dominio determinado. Nuestro trabajo se centró en afirmar y probar los resultados con respecto a la convergencia de DIRECT. Hemos demostrado, en su límite, que aparece un conjunto de puntos de muestreo por conglomerado cerca de los mínimos buscados. Nuestro resultado sólo requiere la continuidad Lipschitz de la función objetivo. Cuanto más fuerte se hacen las suposiciones sobre la función objetivo, es decir, la diferenciabilidad, entonces somos capaces de demostrar las propiedades correspondientes de los puntos deseados.

En la segunda parte hemos ampliado este resultado a los problemas con restricciones adicionales. Si tiene problemas con las limitaciones generales, DIRECT requiere de una ayuda para asignar valores de sustitución a los puntos de función no admisible. Nuestra investigación ha descubierto que DIRECT responde bien cuando estos valores de sustitución se calculan sobre la base de información acerca de su inviabilidad, por funciones de penalización.

El método de los multiplicadores, más concretamente, fue basado en la función Lagrangeana Aumentada para definir los sub-problemas a resolver en cada iteración.

Como ya se ha mencionado, una de las ventajas del enfoque de la función Lagrangeana aumentada para resolver problemas de programación no lineal es su adaptabilidad intrínseca al problema de optimización global. Es decir, si uno sabe cómo resolver sub-problemas globales simples, el método de Lagrangeana aumentada permite resolver globalmente el problema original restringido de optimización.

En este trabajo, se demuestra con rigor este hecho, con una mejora adicional del método de los multiplicadores con la función Lagrangeana aumentada: los sub-problemas se redefinen en cada iteración mediante un conjunto de parámetros que incorpora la información obtenida en la búsqueda de la solución. La utilización del algoritmo DIRECT para la minimización de los sub-problemas, muestra que este método es posible ser aplicado y consigue obtener buenos resultados.

Por eso, se ha utilizado directamente para apoyar este proyecto una comparación con la participación de un algoritmo diferente de optimización (FMINCON) para un conjunto de problemas. Hemos utilizado la capacidad de búsqueda global de DIRECT, y de FMINCON para buscar en las regiones admisibles el mínimo global. Lo que nos lleva a comparar la solidez de cada paquete en la optimización de nuestras funciones. Es de notar que FMINCON es un método de optimización local, pero a veces puede conseguir obtener la solución global, a través de la manipulación de los valores da aproximación inicial.

El algoritmo DIRECT necesita aumentar rápidamente el número de evaluaciones de la función que hace el algoritmo prácticamente inutilizable en los espacios de búsqueda de alta dimensión, es por ese motivo por lo que el algoritmo converge muy lentamente y en algunos casos disminuye la violación también. Además si

aumentamos el valor del algoritmo “aumenta” o el parámetro “u” el tiempo de búsqueda de DIRECT es menor, lo que conlleva un menor número de iteraciones y número de funciones evaluadas. En problemas con muchas restricciones el problema se vuelve más lento y optimiza mejor cuando las restricciones son reducidas.

Los resultados numéricos obtenidos son bastante razonables. En DIRECT se muestra un buen desempeño en comparación con el otro algoritmo en nuestra aplicación y los resultados están a nuestro favor. En la mayoría de los casos en los problemas de prueba se mostró algunas limitaciones, pero estos problemas son considerados como problemas difíciles.

Nuestro desafío es mejorar la eficiencia. Nuestros experimentos parecen indicar que aún hay mucho que mejorar en la metodología Lagrangeana aumentada, ya que el número de iteraciones exterior es siempre moderado.

Pensamos en el futuro explorar más de DIRECT y experimentar otras funciones de penalidad y otras técnicas para resolver problemas con restricciones. Después, pretendemos evaluar el desempeño del algoritmo en la resolución de la colección de problemas g (g01-g24).

BIBLIOGRAFÍA:

[1] Bertsekas,D.P., *"Constrained Optimization and Lagrange Multipliers Methods"*, Academid Press, New York,1982.

[2] Birgin, E.G., Floudas, C.A. and Martinez, J.M., *"Global minimization using and Augmented Lagrangian method with variable lower-level constraints*", Mathematical programming A 125(1), 139-162, 2010.

[3] Dixon, L.C.W. and Szego, G.P., *"Towards Global Optimisation 2"*. North-Holland, New York, NY, first edition, 1978.

[4] Fernandes, E.M.G.P. e Rocha, A.M.A.C., *"Optimização Não Linear",* Univerdidade do Minho, Braga, 1999.

[5] Finkel, D.E., *"Global Optimization with the DIRECT Algorithm",* PhD. Thesis, North Carolina State University, 2005. http://www4.ncsu.edu/~ctk/Finkel_Direct/

[6] Finkel, D.E, Kelley, C.T., *"Convergence Analysis of the DIRECT Algorithm",* North Carolina State University, Center for Research in Scientific Computation, Raleihg, 2004.

[7] Liang, J.J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P.N., Coello, C.A.C., and Deb, K., *"Problem definitions and evaluation criteria for the CEC2006"*, Special Session on Constrained Real-Parameter Optimization, Technical Report, Nanyang Technological University, Singapore, 2006.

[8] Nocedal, J. and Wright, S., *"Numerical Optimization",* Springer, 2nd Edition, Ney York, 2006.

[9] Rayward-Smith, V.J., Osman, I.H., Reeves, C.R., and Smith, G.D., *"Moderm Heuristic Search Methods",* John Wiley & Sons, Chichester, 1996.

[10] The MathWorks. MATLAB Documentation, 7 edition.

[11] Enrique Castillo, Antonio J.. Conejo, Pablo Pedregal, Ricardo Garcia, Natalia Alguacil, *"Formulación Y Resolución de Modelos de Programación Matemática en Ingeniería y Ciencia",* 2002.

[12] Perttunen, C.D., Jones, D.R., Stuckman, B.E., *"Lipschitzian optimization without the lipschitz constant*". Journal of Optimization Theory and Application, 79(1):157–181, October 1993.

ANEXO I

MATLAB - Comando DIRECT

DIRECT- utiliza su propio método (división en rectángulos)

```
[fmin,xmin,hist] =direct(problem,bounds,options,varargin);
```

El comando DIRECT tiene como argumentos de entrada:

Problem(problema) –estructura que contiene el problema.

Problemas-definición de parámetros	
Problem.f	Función objetivo para minimizar (definir en fichero '.m') Si el problema no tiene restricciones este es el único campo que hay que rellenar.
Problem.numconstraints	Nº de restricciones
Problem.constraint(i).func	i-th restricción a manejar
Problem.constraint(i).penalty	pena de valor a elegir

Bounds(limites) - un nx2 vector que describe el dominio, (1ª columna límite inferior,
2ª columna límite superior).

Options(opciones) - argumentos opcionales para controlar el proceso de optimización.

Options-definición de parámetros	
Options.ep	factor Jones. Valor por defecto1e-4
Options.maxevals	nº máximo de evaluaciones de la función. Por defecto 20
Options.maxist	nº máximo de iteraciones. Por defecto 10
Options.maxdeep	nº máximo de divisiones del rectángulo. Por defecto 100
Options.testflag	1 si el mínimo global es conocido, 0 de otro modo. Por defecto 0
Options.globalmin	valor del mínimo global si se conoce, este parámetro se ignora si opts.testfag =1. Por defecto 1
Options.showits	1 muestra todas las estadísticas, 0 caso contrario. Por defecto 1
Options.tol	tolerancia de parado, si tflag=1. Por defecto 0.01
Options.impcons	convierte la capacidad de restricción implícita. Por defecto 0 1 la función objetivo se espera que regrese a la marca que representa la viabilidad del punto de muestreo

Varagim- (opcional) argumentos adicionales que se pasarán al de la función objetivo.

El comando DIRECT tiene como argumentos de salida:

fmin- valor mínimo de la función objetivo que Direct puede encontrar

xmin- localización del valor mínimo en el dominio, es un argumento de salida opcional.

hist- matriz de historial de iteraciones que es útil para los gráficos y tablas. La matriz contiene tres columnas: 1. Iteraciones (iter), 2. Valor mínimo encontrado (f_min), 3. Nº de evaluaciones de la función (fnevals).

ANEXO II

MATLAB - Comando `fmincon`

Universidade do Minho, Escola de Engenharia, Departamento de Produção e Sistemas

Ana Maria A. C. Rocha

FMINCON - usa o método da programação quadrática sequencial.

A função "`FMINCON`" da toolbox de optimização do MATLAB permite resolver problemas com restrições da forma (notação do Matlab)

minimizar	$FUN(x)$	
s.a	$C(x) \leq 0$	restrições **não lineares** de desigualdade
	$Ceq(x) = 0$	restrições **não lineares** de igualdade
	$A\,x \leq B$	restrições **lineares** de desigualdade
	$Aeq\,x = Beq$	restrições **lineares** de igualdade
	$LB \leq x \leq UB$	restrições de limites simples

```
[x,fval,exitflag,output] = fmincon('fun',x0,A,B,Aeq,Beq,LB,UB,'rest',options,...)
```

O comando `fmincon` tem como argumentos de entrada:

`fun` - é a função objectivo a minimizar (definir em ficheiro '.m').

`x0` - é o ponto inicial X.

`A` - matriz dos coeficientes das restrições lineares de desigualdade.

`B` - vector dos termos independentes das restrições lineares de desigualdade.
(Fazer `A=[]` e `B=[]` se não existirem restrições lineares de desigualdade).

`Aeq` - matriz dos coeficientes das restrições lineares de igualdade.

`Beq` - vector dos termos independentes das restrições lineares de igualdade.
(Fazer `Aeq=[]` e `Beq=[]` se não existirem restrições lineares de igualdade).

`LB` - vector dos limites simples inferiores.

`UB` - vector dos limites simples superiores.
Usar matrizes vazias para `LB` e `UB`, no caso de não existirem limites.

Fazer `LB(i) = -Inf` se `X(i)` é não limitado inferiormente.

Fazer `UB(i) = Inf` se `X(i)` é não limitado superiormente.

`rest` - é a função das restrições não lineares de igualdade e desigualdade (definir em ficheiro '.m').

`options` - opções para o controlo do processo de optimização.

Para ver as opções disponíveis fazer:

```
optimset('fmincon')
```

Options - definição de parâmetros	
Algorithm	Escolhe o algoritmo de optimização entre: 'interior-point', 'active-set' ou 'trust-region-reflective' (default)
DerivativeCheck	Compara o Jacoboiano com a sua aproximação por ddiferenças finitas
Diagnostics	Apresenta informação de diagnóstico da função a ser optimizada. 'off' (default).
DiffMaxChange	Diferença máxima nas variáveis para as diferenças finitas, 0.1 (default).
DiffMinChange	Diferença mínima nas variáveis para as diferenças finitas, 1e-8 (default).
Display	Nível de apresentação off - não apresenta nada iter - apresenta resultados em cada iteração notify - apresenta resultado se a função não convergir final - (default) apresenta apenas o resultado final
FunValCheck	Verifica se os valores da função objectivo não são números complexos, Inf ou NaN. on - apresenta erro off - (default) não apresenta erro
GradConstr	Gradiente das funções das restrições não lineares - 'off' (default)
GradObj	Gradiente da função objectivo definido pelo utilizador - 'off' (default)
MaxFunEvals	Número máximo de cálculos da função - (default = 100*numberofvariables)
MaxIter	Número máximo de iterações - (default = 1000)
OutputFcn	Especifica uma ou mais funções que o processo de optimização pode invocar, em cada iteração.
PlotFcns	Representa graficamente várias medidas do progresso do algoritmo, ao longo das iterações. @optimplotx - desenha o ponto actual. @optimplotfunccount - desenha o número de cálculos da função objectivo. @optimplotfval - desenha o valor da função objectivo. @optimplotconstrviolation - desenha o máxima violação das restrições. @optimplotstepsize - desenha o comprimento do passo. @optimplotfirstorderopt - desenha a medida de optimalidade de 1^a ordem.
TolFun	Tolerância de paragem relativamente à função objectivo - (default = 1e-6)
TolCon	Tolerância de paragem relativamente à violação das restrições - (default = 1e-6)
TolX	Tolerância de paragem relativamente a x - (default = 1e-6)

O comando fmincon tem como argumentos de saída:

x - é a solução óptima do problema.

fval - é o valor da função objectivo em x).

exitflag - descreve valores de saída do fminunc	
>0	indica que a função convergiu para uma solução.
0	o número de iterações excedeu o MaxIter ou o MaxFunEvals.
< 0	indica que a função não convergiu para uma solução.

output - estrutura que contém informação acerca do processo de optimização	
output.iterations	indica o número de iterações realizadas
output.funcCount	indica o número de cálculos da função
output.lssteplength	Size of line search step relative to search direction (active-set algorithm only)
output.constrviolation	Maximum of constraint violations (active-set and interior-point algorithms)
output.stepsize	indica o comprimento do passo final (active-set and interior-point algorithms)
output.algorithm	indica o algoritmo usado
output.cgiterations	indica o número total de iterações PCG (trust-region-reflective and interior-point algorithms
output.firstorderopt	indica o valor da medida de optimalidade de 1^a ordem
output.message	indica a mensagem de saída