Universidad Tecnológica de la Mixteca

“Método de Tensor para Programación no Lineal sin Restricciones”

T E S I S

Para obtener el Título de:

Licenciado en Matemáticas Aplicadas

PRESENTA:

Marcelino Ramírez Ibáñez

DIRECTOR DE TESIS:

M. C. Luis René Marcial Castillo

Huajuapan de León Oax., Octubre de 2003

Índice General

Agradecimientos

... Y me pregunto ¿Quién Soy?
soy polvo, soy piedra,
soy río, soy planta,
soy ave que vuela, que canta.
Soy hombre que da gracias,
al dador de la vida, por el amor que derrama
sobre la piedra, el río, la planta.

Introducción

Este trabajo de tesis resuelve el problema:

$$\min_{x \in <^n} f : <^n \; ! \; <, \tag{1}$$

donde $f \; 2 \; C^2$. Este es un problema de programación no lineal sin restricciones, en el que se intenta encontrar algún minimizador local de la función objetivo f sobre todo su dominio.

La estrategia que usan los métodos de programación, es moverse de una iteración a otra de tal manera que la función objetivo sea minimizada en cada una de ellas. Una de las estrategias es usar información local de la iteración actual, tal es el caso del método de máximo descenso, cuya dirección de descenso es el negativo del gradiente en el punto actual; otra de las estrategias es hacer uso de los valores de la primera y segunda derivada de la función objetivo.

Existen muchos métodos para encontrar el mínimo de (1); el que propone este trabajo de tesis es el Método de Tensor. Este método utiliza una estrategia distinta a los métodos estándar, ya que se basa en el modelo de cuarto orden:

$$M_T(x_c + d) = f(x_c) + \nabla f(x_c) \cdot d + \frac{1}{2} \nabla^2 f(x_c) \cdot d^2 + \frac{1}{6} T_c \cdot d^3 + \frac{1}{24} V_c \cdot d^4, \tag{2}$$

donde $d \; 2 \; <^n, x_c$ es el punto actual, $\nabla f(x_c)$ y $\nabla^2 f(x_c)$ son la primera y segunda deriva analítica de f en x_c o una aproximación en diferencias …nitas de ellas, y los términos tensoriales en x_c, $T_c \; 2 \; <^{n \times n \times n}$ y $V_c \; 2 \; <^{n \times n \times n \times n}$ son simétricos.

El modelo (2) en el que se basa el método de tensor, se puede ver como un polinomio de Taylor de orden cuatro, donde los términos tensoriales que aparecen en él interpolan a la función y al gradiente de la función en la iteración anterior. Este método, como todos los métodos de programación es iterativo: empieza con un punto inicial y genera una sucesión de puntos de tal manera que se alcance la solución. Además, el método de tensor hace uso de los términos tensoriales para lograr la convergencia más rápido.

El método de tensor fue introducido por Schnabel y Chow[1]. Este tipo de método trata de obtener mejores resultados que los métodos tradicionales de programación sin restricciones; e intenta ser al menos tan e…ciente como los métodos tradicionales en problemas donde $\nabla^2 f(x)$ es no singular. El método

propuesto por Schnabel y Chow[1] sólo trabaja para n pequeña(menor que cien), y tiene problemas cuando la matriz hessiana es rala, es decir, la matriz hessiana tiene muchos ceros.

Este trabajo se basa en el artículo de Ali Bouaricha[2], donde el autor extiende el método de tensor para n grande y matriz hessiana rala. Sólo se estudia el caso donde la matriz hessiana es no singular.

En este trabajo se compararon los resultados obtenidos al aplicar el método de tensor y los métodos estándar, y para ello se creó el programa OPTIMIN. En todos los casos el método de tensor obtuvo mejores resultados, mismos que se muestran en el capítulo 5.

OPTIMIN es un programa que se desarrolló con el …n de resolver (1), con los siguientes métodos: máximo descenso, gradientes conjugados y del tensor. Por medio de OPTIMIN, es posible comparar los métodos antes mencionados, ya que cuenta con la opción de gra…car el comportamiento del método a través del tiempo y de salvar los resultados que arroja el método instrumentado.

En el capítulo 1 se dan algunos conceptos preliminares que se ocupan a lo largo del trabajo y que son útiles para el desarrollo del mismo, no se hace una explicación extensa de los mismos, en lugar de ello se da una breve descripción de ellos. Para una explicación más detallada ver (Dennis y Schnabel,1991), (Gill, Murray y Wright, 1981), (Chong y Stanislaw,1996) y (Nocedal y Wright, 1999).

En el capítulo 2 se da una breve explicación de algunos métodos clásicos para la programación no lineal sin restricciones en la búsqueda de minimizadores locales de funciones no lineales, tales como el de máximo descenso y el de gradientes conjugados, los cuales nos servirán para comparar con el método de tensor. En la práctica es bien sabido que un factor muy importante en la convergencia de los métodos de programación sin restricciones es el tamaño de paso, por lo que se dedica parte de este capítulo al estudio del mismo.

En el capítulo 3 se desarrolla el método de tensor y se explican sus fundamentos teóricos.

En el capítulo 4 se explican los algoritmos instrumentados: máximo descenso, gradientes conjugados, método de tensor y búsqueda lineal.

En el capítulo 5 se dan algunas caracerísticas del programa OPTIMIN, así como resultados del método de tensor comparado con el método de máximo descenso y el de gradientes conjugados, aplicados a problemas prueba.

En el capítulo 6 se muestran algunas conclusiones sobre el método de tensor y su implementación en OPTIMIN.

El capítulo 7 esta dedicado al programa OPTIMIN, se da el manual de usuario y algunos ejemplos de cómo opera. Este programa cuenta con una interfaz grá…ca amigable que permite al usuario encontrar mínimos de funciones prueba ya instaladas. También esta diseñado para la minimización de nuevas funciones de…nidas por el usuario.

Por último, se muestra la bibliografía utilizada en este trabajo de tesis.

Capítulo 1

Generalidades

En este capítulo se desarrollan algunos conceptos básicos de la programación sin restricciones, dado que el trabajo pertenece a esta área, es importante entender bien las ideas y conceptos que están alrededor de ella. La mayoría de los conceptos provienen del cálculo, álgebra lineal y análisis numérico, entre otros. Donde surgen conceptos nuevos, es en cómo se plantea el método de tensor ya que contiene algunos términos tensoriales que requieren de ideas y notaciones nuevas, pero esto se estudia en el capítulo 4. Así que por el momento sólo se enfoca a los conceptos básicos de la programación sin restricciones.

1.1 De…niciones Básicas

Dado que la intención es encontrar el mínimo de una función f, a continuación daremos la de…nición matemática de este concepto para después, con la ayuda del cálculo, poder establecer algunos criterios que nos ayuden a reconocer dichos mínimos.

De…nición 1.1 (Minimizador local) Sea la función f : $<^n$! < y - un subconjunto de $<^n$. Un punto $x^¤$ 2 - es un minimizador local de f en - si existe $\epsilon > 0$ tal que $f(x)$ ¸ $f(x^¤)$ para todo x 2 -nf$x^¤$g con kx ¡ $x^¤$k $< \varepsilon$. En caso de que se cumpla $f(x) > f(x^¤)$ entonces decimos que el punto $x^¤$ es un minimizador local estricto.

De…nición 1.2 (Minimizador global) Sea la función f : $<^n$! < y - un subconjunto de $<^n$. Un punto $x^¤$ 2 - es un minimizador global de f sobre - si $f(x)$ ¸ $f(x^¤)$ para todo x 2 -nf$x^¤$g.

En la …gura 1.1 se da un ejemplo de minimizador local, minimizador local estricto y minimizador global.

De…nición 1.3 La función continua f : $<^n$! < se dice que es continuamente diferenciable en x 2 $<^n$, si $(\partial f/\partial x_i)(x)$ existe en una vecindad de

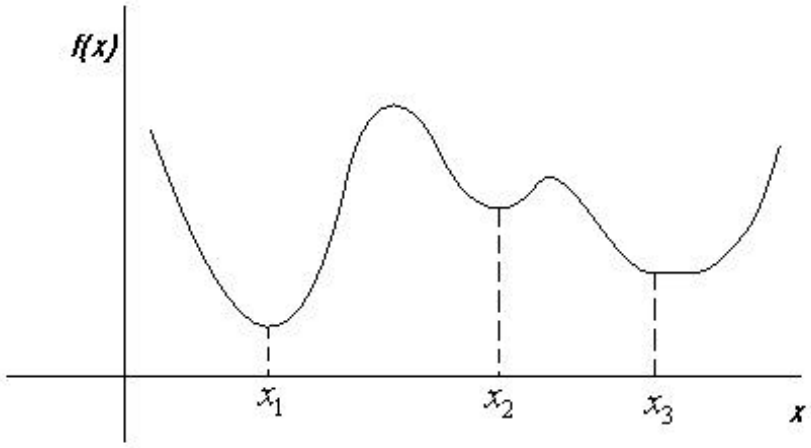


Figura 1.1: Ejemplos de minimizadores: x_1 : minimizador global, x_2 : minimizador local estricto, x_3 : minimizador local (no estricto).

x y es continua en x, con i = 1,2, . . . , n. En este caso el gradiente de f en x esta de…nido como

$$\nabla f(x) = \left[\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x) \right]^T.$$

Como se puede ver, el gradiente es una función de $<^n$ en $<^n$, si f es continuamente diferenciable en $<^n$.

De…nición 1.4 La función f se dice que es continuamente diferenciable en una vecindad abierta $D \subseteq <^n$, denotado por $f \in C^1(D)$, si es continuamente diferenciable en cada punto $x \in D$.

Teorema 1.5 (CNPO[1]) Sea $f : <^n \to <$ una función continuamente diferenciable en un conjunto abierto que contiene a x^*. Si x^* es un minimizador local de f, entonces $\nabla f(x^*) = 0$.

Es importante señalar que la implicación de regreso no se cumple: por ejemplo para la función continuamente diferenciable $f(x) = x^3$ el gradiente en $x = 0$ es cero , pero 0 no es un minimizador de f, de hecho $x = 0$ es un punto de in‡exión de f.

De…nición 1.6 La función continua $f : <^n \to <$ se dice que es dos veces continuamente diferenciable (continuamente diferenciable de orden dos) en

[1] Condición Necesaria de Primer Orden.

$x \in \mathbb{R}^n$, si $\left[\partial^2 f/\partial x_i \partial x_j\right](x)$ existe en una vecindad de x y es continua en x, con $1 \leq i, j \leq n$; la hessiana H de f en x esta de…nida como la matriz de $n \times n$ cuyos elementos i, j son:

$$\nabla^2 f(x)_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j},\ 1 \leq i, j \leq n.$$

Al punto x^* que satisface $\nabla f(x^*) = 0$, se le llamará punto estacionario. Luego, de acuerdo al teorema anterior cualquier minimizador local es un punto estacionario. Así, los puntos x_2 y x_3 de la …gura 1.1 son puntos estacionarios.

De…nición 1.7 La función f se dice que es continuamente diferenciable de orden dos en una vecindad abierta $D \subset \mathbb{R}^n$, denotado por $f \in C^2(D)$, si es dos veces continuamente diferenciable en cada punto $x \in D$.

Teorema 1.8 (CNSO[2]) Sea $f : \mathbb{R}^n \to \mathbb{R}$, $f \in C^2(D)$ con D una vecindad abierta de x^*. Si x^* es un minimizador local de f, entonces $\nabla f(x^*) = 0$ y $H(x^*)$ es semide…nida positiva, esto es, para toda $p \in \mathbb{R}^n$ se cumpla que $p^T H(x^*) p \geq 0$.

El siguiente teorema nos describe las condiciones su…cientes para saber si un punto es un minimizador local de una función f.

Teorema 1.9 (CSSO[3]) Sea $f : \mathbb{R}^n \to \mathbb{R}$, $f \in C^2(D)$ con D una vecindad abierta de x^*. Si $\nabla f(x^*) = 0$ y $H(x^*)$ es de…nida positiva, entonces x^* es un minimizador local estricto de f.

Recordar que uno de los criterios para que una matriz $A \in \mathbb{R}^{n \times n}$ sea de…nida positiva es que para toda $p \in \mathbb{R}^n, p \neq 0$ se cumple que $p^T A p > 0$.

Nótese que la implicación de regreso en CNSO no necesariamente se cumple: un punto x^* puede ser un minimizador local estricto y no cumplir alguna de las condiciones su…cientes. Por ejemplo la función $f(x) = x^4$, para la cual $x^* = 0$ es un minimizador estricto en el cual la matriz hessiana se anula, $H(x^*)$ no es de…nida positiva.

1.2 Estrategias de Búsqueda

Empezando en un punto inicial x_0, los algoritmos de programación generan una sucesión de iteraciones $\{x_k\}_{k=0}^{\infty}$ que terminan ya sea cuando no se puede hacer más progreso o cuando ya se ha aproximado su…cientemente a la solución con buena precisión. Al decidir como moverse de un punto x_k al siguiente, los algoritmos usan información acerca de f en x_k y posiblemente información de las puntos anteriores. Utilizan esta información para encontrar el nuevo punto x_{k+1} con un valor en la función objetivo menor al de x_k.

[2] Condición Necesaria de Segundo Orden.
[3] Condición Su…ciente de Segundo Orden.

Existen dos estrategias fundamentales para ir del punto actual x_k al siguiente x_{k+1}: búsqueda lineal y región de con…anza. Todos los algoritmos desarrollados en este trabajo solo utilizan una de las estrategias : la búsqueda lineal. En la sección 2.3 se hace un análisis más profundo de esta).

1.2.1 Búsqueda Lineal

Una de las estrategia es la llamada búsqueda lineal, en la cual el algoritmo elige una dirección d_k, y busca a lo largo de esta dirección un nuevo punto x_{k+1} que disminuya la función objetivo desde el punto actual x_k. Esto se puede hacer resolviendo el siguiente problema de programación unidimensional para α (el tamaño de paso).

$$\min_{\alpha>0} f(x_k + \alpha d_k). \tag{1.1}$$

Al resolver (1.1) se quiere obtener el máximo bene…cio en la dirección d_k, pero una minimización exacta es costosa computacionalmente hablando. En lugar de ello, el algoritmo de búsqueda lineal genera una sucesión …nita de tamaños de paso hasta que se aproxime a algún mínimo de (1.1). En el nuevo punto se calcula la dirección de descenso, el tamaño de paso y el proceso se repite.

1.2.2 Región de Con…anza

En la otra estrategia, conocida como región de con…anza, la información obtenida acerca de f es usada para construir una función modelo m_k cuyo comportamiento cerca del punto actual x_k es similar al de la función objetivo f. Como el modelo m_k puede no ser una buena aproximación de f cuando x está lejos de x_k , se restringe la búsqueda del minimizador de m_k a alguna región alrededor de x_k. Es decir, se encuentra el candidato p resolviendo el siguiente problema:

$$\min_{p \in \mathbb{R}^n} m_k = f_k + \nabla f_k^T p + p^T B_k p$$

$$s.a.$$

$$\|p\| < \Delta_k,$$

donde f_k y ∇f_k son la función y el gradiente evaluados en x_k. La matriz B_k puede ser la hessiana $\nabla^2 f_k$ o una aproximación a ella en el punto x_k.

Capítulo 2

Programación no Lineal sin Restricciones

En este capítulo se abordan los métodos de máximo descenso y gradientes conjugados, los cuales se eligieron para compararlos con el método de tensor. Se darán algunas de sus características principales y en el capítulo 5 se describe detalladamente su implementación algorítmica.

Todos los métodos de programación sin restricciones tratan de resolver el problema

$$\min_{x \in <^n} f : <^n \; ! \; <, \tag{2.1}$$

donde f es al menos dos veces continuamente diferenciable.

2.1 Método de Máximo Descenso

El método de máximo descenso es un algoritmo de gradiente donde el tamaño de paso α_k es seleccionado de tal manera que se alcanza el máximo decrecimiento de la función objetivo en cada paso en la dirección negativa del gradiente, esto es, $d_k = ¡ r f(x_k)$.

Para calcular un punto estacionario de (2.1), el método de máximo descenso usa el siguiente esquema:

Si kd_kk = 0, parar el método y x_k es un punto estacionario de f; de lo contrario se procede a calcular el tamaño de paso α_k.

$$\alpha_k = \arg \min_{\alpha , 0} \; f(x_k + \alpha d_k),$$

luego el nuevo punto será:

$$x_{k+1} = x_k + \alpha_k d_k,$$

y después se procede a iterar.

Una característica del método de máximo descenso, que no es difícil de probar, es que se mueve ortogonalmente en cada paso; esto es x_{k+1} ¡ x_k es ortogonal a x_{k+2} ¡ x_{k+1}.

2.2 Método de Gradientes Conjugados

La introducción del método de gradiente conjugado para funciones no lineales por Flecher-Reeves en los 60´s, marcó un gran avance en el campo de la programación sin restricciones. De hecho, es una de las primeras técnicas para resolver este tipo de problemas de programación. Numerosas variantes del método de Flecher-Reeves se han propuesto en los últimos años, y algunas son ampliamente usadas en la práctica. Algunas de las ventajas que tienen estos métodos es que no requieren de almacenamiento de matrices y son más rápidos que el método de máximo descenso.

2.2.1 Método de Flecher-Reeves

El método de Flecher-Reeves surge como una modi…cación al método de gradiente conjugado lineal, donde se resuelve un sistema lineal de ecuaciones de la forma $Ax = b$, con A matriz simétrica y de…nida positiva.

El algoritmo de gradiente conjugado propuesto por Flecher y Reeves (FR-GC), es el siguiente:

Algoritmo 2.1 Entrada: Punto inicial x_0.
$f_0 \leftarrow f(x_0)$, $\nabla f_0 \leftarrow \nabla f(x_0)$;
$p_0 \leftarrow -\nabla f_0$, $k \leftarrow 0$;
Mientras $\nabla f_k \neq 0$
Calcular α_k con una búsqueda lineal;
$x_{k+1} \leftarrow x_k + \alpha_k p_k$;
Evaluar ∇f_{k+1};

$$\beta_{k+1}^{FR} \leftarrow \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}; \tag{2.2}$$

$$p_{k+1} \leftarrow -\nabla f_{k+1} + \beta_{k+1}^{FR} p_k; \tag{2.3}$$

$$k \leftarrow k+1;$$

Fin(mientras).

Aquí α_k es el tamaño de paso, ∇f_k es el gradiente evaluado en x_k. Es necesario que α_k cumpla las condiciones fuertes de Wolfe para asegurar que la dirección p_k sea de descenso y el nuevo punto x_{k+1} sea satisfactorio. Se mencionan aquí las condiciones fuertes de Wolfe (2.17) y (2.18) que son necesarias para satisfacer la condición de descenso, y en la sección 2.3 se describen ampliamente.

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T p_k, \tag{2.4}$$

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k, \tag{2.5}$$

con $0 < c_1 < c_2 < 1$

Probemos que las direcciones p_k generadas por el algoritmo (FR-GC) son de descenso, para ello se utilizá el siguiente lema

Lema 2.2 Supóngase que se implementa el algoritmo (FR-GC) con tamaño de paso α_k que satisface las condiciones fuertes de Wolfe (2.4) y (2.5) con $0 < c_2 < \frac{1}{2}$. Entonces, el método genera direcciones de descenso p_k que satisfacen las siguientes desigualdades

$$\frac{-1}{1-c_2} \leq \frac{\nabla f_k^T p_k}{\|\nabla f_k\|^2} \leq \frac{2c_2-1}{1-c_2}, \quad para\ toda\ k = 0, 1, \ldots \tag{2.6}$$

Demostración. La prueba es por inducción. Para $k = 0$, el término $\frac{\nabla f_k^T p_k}{\|\nabla f_k\|^2} = -1$, luego como $c_2 - 1 < 2c_2 - 1$, entonces $-1 < \frac{2c_2-1}{1-c_2}$ y por tanto (2.6) se cumple.

Supóngase que (2.6) se cumple para algún $k \geq 1$. Multiplicando por ∇f_{k+1}^T en ambos lados de (2.3) tenemos

$$\nabla f_{k+1}^T p_{k+1} = -\|\nabla f_{k+1}\|^2 + \beta_{k+1}^{FR} \nabla f_{k+1}^T p_k,$$

dividiendo ambos lados de la ecuación anterior por $\|\nabla f_{k+1}\|^2$ y utilizando (2.2) obtenemos

$$\frac{\nabla f_{k+1}^T p_{k+1}}{\|\nabla f_{k+1}\|^2} = -1 + \frac{(\nabla f_{k+1}^T \nabla f_{k+1})\nabla f_{k+1}^T p_k}{(\nabla f_k^T \nabla f_k)\|\nabla f_{k+1}\|^2} = -1 + \frac{\nabla f_{k+1}^T p_k}{\|\nabla f_k\|^2}. \tag{2.7}$$

Usando la condición de Wolfe (2.5) tenemos

$$|\nabla f_{k+1}^T p_k| \leq -c_2 \nabla f_k^T p_k,$$

de donde

$$-1 + c_2 \frac{\nabla f_k^T p_k}{\|\nabla f_k\|^2} \leq -1 + \frac{\nabla f_{k+1}^T p_k}{\|\nabla f_k\|^2} \leq -1 - c_2 \frac{\nabla f_k^T p_k}{\|\nabla f_k\|^2},$$

y combinando la desigualdad anterior con la ecuación (2.7) tenemos

$$-1 + c_2 \frac{\nabla f_k^T p_k}{\|\nabla f_k\|^2} \leq \frac{\nabla f_{k+1}^T p_{k+1}}{\|\nabla f_{k+1}\|^2} \leq -1 - c_2 \frac{\nabla f_k^T p_k}{\|\nabla f_k\|^2}.$$

Sustituyendo el lado izquierdo de la hipótesis de inducción para $\nabla f_k^T p_k / \|\nabla f_k\|^2$ se obtiene

$$-1 - \frac{c_2}{1-c_2} \leq \frac{\nabla f_{k+1}^T p_{k+1}}{\|\nabla f_{k+1}\|^2} \leq -1 + \frac{c_2}{1-c_2},$$

equivalentemente

$$\left(-\frac{1}{1-c_2}\right)\cdot\frac{\nabla f_{k+1}^T p_{k+1}}{\|\nabla f_{k+1}\|^2}\cdot\frac{2c_2-1}{1-c_2},$$

por tanto, (2.6) se cumple para $k+1$. ■

Del lema anterior, como $c_2 \in \left(0,\frac{1}{2}\right)$, entonces $\frac{2c_2-1}{1-c_2} < 0$ y por tanto $\nabla f_k^T p_k < 0$, lo cual muestra que la dirección p_k es de descenso.

2.2.2 Método de Polak-Ribiere

Hay muchas variantes del método de Flecher-Reeves que di…eren principalmente en la elección del parámetro β_k. Uno de los más importantes y también uno de los más usados, es el propuesto por Polak-Ribiere, el cual de…ne el parámetro de la siguiente manera:

$$\beta_{k+1}^{PR} = \frac{\nabla f_{k+1}^T(\nabla f_{k+1}-\nabla f_k)}{\|\nabla f_k\|^2}. \tag{2.8}$$

Si en el algoritmo (FR-GC) se reemplaza (2.2) por (2.8) se tiene el algoritmo propuesto por Polak y Ribiere(PR-GC). Un hecho sorprendente es que para este algoritmo, las condiciones fuertes de Wolfe no garantizan que p_k sea siempre una dirección de descenso. Si se de…ne el parámetro β_k como

$$\beta_{k+1}^{+} = \max\{\beta_{k+1}^{PR}, 0\}, \tag{2.9}$$

y se hace una simple adaptación a las condiciones fuertes de Wolfe, sí se puede asegurar que se satisfaga la condición de descenso.

2.2.3 Método de Hestenes-Stiefel

Otra de las variantes del método de Flecher-Reeves es el propuesto por Hestenes-Stiefel donde β_k está dada por:

$$\beta_{k+1}^{PR} = \frac{\nabla f_{k+1}^T(\nabla f_{k+1}-\nabla f_k)}{(\nabla f_{k+1}-\nabla f_k)^T p_k}. \tag{2.10}$$

En la práctica, ninguna de las β_k propuestas ha probado ser más e…ciente que la del método de Polak-Ribiere (2.8). Esto no quiere decir que siempre el método de Polak-Ribiere converge más rápido hacia el minimizador o que siempre de las mejores aproximaciones; sino que en la mayoría de los casos ofrece mejores resultados, pero hay ocasiones en que es necesario usar algún otro método. En el capítulo 6 se muestran algunos ejemplos que ilustran lo anterior.

2.3 Tamaño de Paso

En esta sección se mencionan algunas consideraciones para hacer la correcta elección del tamaño de paso α_k, ya que el éxito de los métodos de búsqueda lineal depende de las correctas elecciones de la dirección d_k y del tamaño de paso α_k.

En cada iteración, los métodos de búsqueda lineal calculan una dirección de descenso d_k y deciden que tanto moverse a lo largo de esa dirección. La iteración está dada por:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2.11}$$

donde el escalar positivo α_k es llamado el tamaño de paso.

Al calcular el tamaño de paso α_k se requiere hacer una reducción signi…cativa de f, pero al mismo tiempo, no se quiere gastar mucho tiempo haciendo tal elección. La elección ideal sería el minimizador global de la función univariada $\phi(\phi)$ de…nida por

$$\phi(\alpha) = f(x_k + \alpha d_k),\ \alpha > 0, \tag{2.12}$$

pero en general es muy costoso encontrar este valor. También, encontrar un minimizador local de ϕ con moderada precisión generalmente requiere de muchas evaluaciones de la función objetivo f y posiblemente del gradiente r f. Estrategias más prácticas efectúan búsquedas lineales inexactas para identi…car el tamaño de paso que alcance adecuadas reducciones en f al mínimo costo.

Una condición simple que podemos imponer a α_k es que produzca una reducción en cada paso en f, esto es, $f(x_k + \alpha_k d_k) < f(x_k)$. Pero esto no es su…ciente como se ilustra en la …gura 2.1, donde el mínimo es $f(x^¤) = ¡1, con\ x^¤ = 1$; y la sucesión de puntos x_k no generan su…ciente reducción en f. Lo que veremos mas adelante es cómo impedir que suceda este tipo de situaciones.

2.3.1 Las Condiciones de Wolfe

Una condición muy popular en la búsqueda lineal inexacta es la que especi…ca que el tamaño de paso α_k, primero que todo, proporcione un su…ciente decrecimiento en la función objetivo f, medido por la siguiente desigualdad:

$$f(x_k + \alpha d_k) \cdot f(x_k) + c_1 \alpha \text{ r } f_k^T d_k, \tag{2.13}$$

para alguna constante c_1 2 (0,1). En otras palabras, la reducción en f debe ser proporcional a ambos, el tamaño de paso α_k y la derivada direccional r $f_k^T d_k$. La desigualdad (2.13) es algunas veces llamada como la condición de Armijo.

El lado derecho de la ecuación (2.13), puede ser denotado por $l(\alpha)$. La función $l(\phi)$ tiene pendiente negativa c_1 r $f_k^T d_k$. En la práctica, el valor elegido de c_1 es pequeño, por ejemplo $c_1 = 10^{¡4}$. Es así, como la condición de su…ciente decrecimiento se satisface cuando α cumple que $\phi(\alpha) \cdot l(\alpha)$.

La condición de Armijo no es su…ciente por si misma para asegurar que el algoritmo haga progresos razonables, ya que puede ser satisfecha por valores de

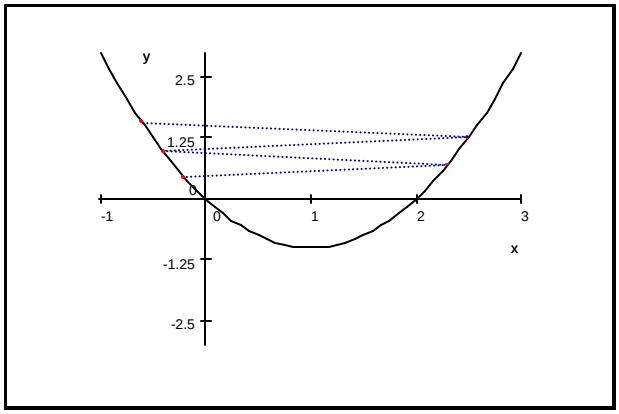


Figura 2.1: Sucesión de puntos que genera insu…ciente reducción en f.

α muy pequeños. Para no aceptar tamaños de paso muy pequeños se introduce una segunda condición, llamada la condición de curvatura, la cual pide que se satisfaga

$$\nabla f(x_k + \alpha_k d_k)^T d_k \geq c_2 \nabla f_k^T d_k, \tag{2.14}$$

para alguna constante $c_2 \in (c_1, 1)$, donde c_1 es la constante de (2.13). Para mejores resultados, el valor de c_2 se toma como 0.9 cuando la dirección de búsqueda d_k es elegida por un método de Newton o Quasi-Newton y 0.1 cuando d_k es obtenida por un método de gradientes conjugados.

Las condiciones de curvatura y su…ciente decrecimiento colectivamente son conocidas como las condiciones de Wolfe. Estas condiciones se vuelven a plantear aquí para futuras referencias:

$$f(x_k + \alpha d_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T d_k, \tag{2.15}$$

$$\nabla f(x_k + \alpha_k d_k)^T d_k \geq c_2 \nabla f_k^T d_k, \tag{2.16}$$

con $0 < c1 < c2 < 1$.

Se puede modi…car la condición de curvatura para forzar que α_k esté en la vecindad de un minimizador local o punto estacionario de ϕ, exigiendo que satisfaga las llamadas condiciones fuertes de Wolfe:

$$f(x_k + \alpha d_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T d_k, \tag{2.17}$$

$$|\nabla f(x_k + \alpha_k d_k)^T d_k| \leq c_2 |\nabla f_k^T d_k|, \tag{2.18}$$

con $0 < c_1 < c_2 < 1$.

La pregunta inmediata que se deriva de las condiciones de Wolfe es, ¿existen tamaños de paso que las satisfagan para cualquier función f suave y acotada inferiormente?. La respuesta es sí, y nos lo dice el siguiente lema

Lema 2.3 Supóngase que $f : <^n \; ! \; <$ es continuamente diferenciable. Sea d_k una dirección de descenso en x_k, y supóngase que f es acotada inferiormente a lo largo del rayo fx_k + $\alpha d_k : \alpha > 0$g. Entonces, si $0 < c_1 < c_2 < 1$, existen intervalos de tamaño de paso que satisfacen las condiciones de Wolfe (2.15)-(2.16) y las condiciones fuertes de Wolfe (2.17)-(2.18) [ver [6]].

Capítulo 3

Método de Tensor

En este capítulo se describe el método de tensor para resolver el problema de programación sin restricciones.

$$\begin{aligned} \text{Dada } f \quad &: \quad <^n \ ¡ \ ! \ <, \text{ encontrar } x^{¤} \ 2 \ <^n \\ \text{tal que } \ f(x^{¤}) \quad &\cdot \quad f(x) \ \text{ para toda } x \ 2 \ D, \end{aligned} \tag{3.1}$$

donde D es algún conjunto abierto conteniendo a $x^{¤}$ y f es convexa en D . Se asume que f 2 C^2 y n es grande.

El método de tensor para programación sin restricciones en cada iteración se basa en el modelo de cuarto orden de la función objetivo $f(x)$,

$$M_T \ (x_c + d) = f(x_c) + \mathrm{r} \ f(x_c) \natural d + \frac{1}{2} \mathrm{r}^2 f(x_c) \natural d^2 + \frac{1}{6} T_c \natural d^3 + \frac{1}{24} V_c \natural d^4, \tag{3.2}$$

donde d 2 $<^n$, x_c es el punto actual, r $f(x_c)$ y r $^2 f(x_c)$ son la primera y segunda deriva analítica de f en x_c o una aproximación en diferencias …nitas de ellas, y los términos tensoriales en x_c, T_c 2 $<^{n £ n £ n}$ y V_c 2 $<^{n £ n £ n £ n}$ son simétricos. Los términos tensoriales son seleccionados de tal manera que el modelo interpola a la función y el gradiente en un punto anterior, lo cual es crucial para la e…ciencia del método de tensor. Este método no requiere de más evaluaciones en la función o en las derivadas, y difícilmente más almacenamiento u operaciones aritméticas que aquellos métodos basados en el método de Newton.

Este capítulo esta organizado de la siguiente manera. En la sección 3.1 se dan algunas de…niciones y notaciones que se utilizan a lo largo del capítulo. En la sección 3.2 se hace una corta revisión de las técnicas introducidas por Schnabel y Chow [1] para formar el modelo de tensor. En la sección 3.3 se desarrolla el método de tensor cuando la matriz hessiana es no singular. En la sección 3.4 se describe brevemente el algoritmo del modelo de tensor para el caso de la matriz hessiana no singular.

3.1 De…niciones y Notación

Por comodidad se usa la notación $\nabla f(x_c) \cdot d$ para $\nabla f(x_c)^T d$, $\nabla^2 f(x_c) \cdot d^2$ para $d^T \nabla^2 f(x_c) d$ para ser consistentes con la notacion tensorial $T_c \cdot d^3$ y $V_c \cdot d^4$. También por simplicidad, se abrevian términos de la forma $dd, ddd, dddd$ por d^2, d^3, d^4, respectivamente. Antes de continuar, se de…ne la notación tensorial usada arriba.

De…nición 3.1 Sean $T \in \mathbb{R}^{n \times n \times n}$ y $u, v, w \in \mathbb{R}^n$. Se de…nen $T \cdot uvw \in \mathbb{R}$, $T \cdot vw \in \mathbb{R}^n$, como

$$T \cdot uvw = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} T(i,j,k)\, u(i)\, v(j)\, w(k),$$

$$(T \cdot vw)(i) = \sum_{j=1}^{n} \sum_{k=1}^{n} T(i,j,k)\, v(j)\, w(k), \text{ con } i = 1, 2, \ldots, n.$$

De…nición 3.2 Sean $V \in \mathbb{R}^{n \times n \times n \times n}$ y $r, u, v, w \in \mathbb{R}^n$. Se de…nen $V \cdot ruvw \in \mathbb{R}$, $V \cdot uvw \in \mathbb{R}^n$ como

$$V \cdot ruvw = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} V(i,j,k,l)\, r(i)\, u(j)\, v(k)\, w(l),$$

$$(V \cdot uvw)(i) = \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} V(i,j,k,l)\, u(j)\, v(k)\, w(l), \text{ con } i = 1, 2, \ldots, n.$$

3.2 Formulación del Método de Tensor

El método de tensor para programación sin restricciones basa cada iteración sobre el modelo de cuarto orden de la función no lineal $f(x)$ dado por (3.2).

Las elecciones de T_c y V_c en (3.2) hacen que los términos $T_c \cdot d^3$ y $V_c \cdot d^4$ de tercer y cuarto orden respectivamente, tengan formas simples y muy útiles. Estos términos tensoriales son seleccionados de tal manera que el modelo de tensor interpola información de la función en el punto anterior x_{-1}.

Las condiciones de interpolación en el punto anterior x_{-1} estan dados por

$$f(x_{-1}) = f(x_c) + \nabla f(x_c) \cdot s + \frac{1}{2} \nabla^2 f(x_c) \cdot s^2 + \frac{1}{6} T_c \cdot s^3 + \frac{1}{24} V_c \cdot s^4, \qquad (3.3)$$

y

$$\nabla f(x_{-1}) = \nabla f(x_c) + \nabla^2 f(x_c) \cdot s + \frac{1}{2} T_c \cdot s^2 + \frac{1}{6} V_c \cdot s^3, \qquad (3.4)$$

donde

$$s = x_{-1} - x_c.$$

Schnabel y Chow[1] eligieron T_c y V_c que satisfagan (3.3) y (3.4). Ellos primero mostraron que éstas condiciones determinan de manera única a $T_c \ddagger d^3$ y $V_c \ddagger d^4$. En efecto, multiplicando (3.4) por s tenemos

$$\nabla f(x_{+1}) \ddagger s = \nabla f(x_c) \ddagger s + \nabla^2 f(x_c) \ddagger s^2 + \frac{1}{2} T_c \ddagger s^3 + \frac{1}{6} V_c \ddagger s^4. \tag{3.5}$$

Sean $\alpha, \beta \in \mathbb{R}$ tales que

$$\begin{aligned} \alpha &= T_c \ddagger d^3 \\ \beta &= V_c \ddagger d^4. \end{aligned}$$

De (3.3) y (3.5) ellos obtuvieron que

$$\begin{aligned} f(x_{+1}) - f(x_c) - \nabla f(x_c) \ddagger s - \frac{1}{2} \nabla^2 f(x_c) \ddagger s^2 &= \frac{1}{6}\alpha + \frac{1}{24}\beta \\ \nabla f(x_{+1}) \ddagger s - \nabla f(x_c) \ddagger s - \nabla^2 f(x_c) \ddagger s^2 &= \frac{1}{2}\alpha + \frac{1}{6}\beta, \end{aligned}$$

de donde resulta el siguiente sistema de ecuaciones

$$q_1 = \frac{1}{6}\alpha + \frac{1}{24}\beta, \tag{3.6}$$

$$q_2 = \frac{1}{2}\alpha + \frac{1}{6}\beta, \tag{3.7}$$

con q_1 y $q_2 \in \mathbb{R}$ dados por

$$\begin{aligned} q_1 &= f(x_{+1}) - f(x_c) - \nabla f(x_c) \ddagger s - \frac{1}{2} \nabla^2 f(x_c) \ddagger s^2. \\ q_2 &= \nabla f(x_{+1}) \ddagger s - \nabla f(x_c) \ddagger s - \nabla^2 f(x_c) \ddagger s^2. \end{aligned}$$

El sistema (3.6) - (3.7) es no singular; y por tanto los valores de α y β estan determinados de manera única. Luego, las condiciones de interpolación determinan de manera única a $T_c \ddagger d^3$ y $V_c \ddagger d^4$. Como solo estas son las condiciones de interpolación, T_c y V_c no estan su...cientemente determinadas.

Schnabel y Chow[1] eligieron T_c y V_c primeramente seleccionando la más pequeña V_c simétrica en la norma de Frobenius para la cual

$$V_c \ddagger d^4 = \beta,$$

donde β esta determinada por (3.6) -(3.7). Luego ellos sustituyeron este valor de V_c en (3.4), obteniendo

$$\nabla f(x_{+1}) - \nabla f(x_c) - \nabla^2 f(x_c) \ddagger s - \frac{1}{6} V_c \ddagger s^3 = \frac{1}{2} T_c \ddagger s^2,$$

de donde

$$T_c \cdot s^2 = a, \tag{3.8}$$

con

$$a = 2 \left[\nabla f(x_{+1}) - \nabla f(x_c) - \nabla^2 f(x_c) \cdot s - \frac{1}{6} V_c \cdot s^3 \right]. \tag{3.9}$$

Este es un conjunto de n ecuaciones con n^3 incógnitas $T_c(i,j,k)$, $1 \leq i,j,k \leq n$. Para ser más precisos, Schnabel y Chow[1] eligieron las más pequeñas T_c y V_c en la norma de Frobenius, tales que satisfagan las ecuaciones (3.8) - (3.9). Esto es,

$$\begin{aligned} &\min_{V_c \in \mathbb{R}^{n \times n \times n}} \|V_c\|_F \\ &s.a \quad V_c \cdot s^4 = \beta, \text{ y } V_c \text{ simétrica,} \end{aligned} \tag{3.10}$$

y

$$\begin{aligned} &\min_{T_c \in \mathbb{R}^{n \times n \times n}} \|T_c\|_F \\ &s.a \quad T_c \cdot s^3 = a, \text{ y } T_c \text{ simétrica.} \end{aligned} \tag{3.11}$$

La solución a (3.10) es

$$V_c = \gamma (s - s - s - s), \ \gamma = \frac{\beta}{(s^T s)^4},$$

donde el tensor $V_c = \gamma (s - s - s - s) \in \mathbb{R}^{n \times n \times n \times n}$ es llamado tensor de cuarto orden rango uno para el cual, $V_c(i,j,k,l) = s(i)\, s(j)\, s(k)\, s(l)$, $1 \leq i,j,k,l \leq n$. Se usa la notación - para ser consistentes con [1].

En efecto, por de…nición

$$V_c \cdot ssss = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} V(i,j,k,l)\, s(i)\, s(j)\, s(k)\, s(l),$$

sustituyendo $V_c = \gamma (s - s - s - s)$ se tiene que

$$V_c \cdot sssss = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} [\gamma (s(i)s(j)s(k)s(l))s(i)s(j)s(k)s(l)]$$

$$= \gamma \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \left[s(i)^2 s(j)^2 s(k)^2 s(l)^2\right]$$

$$= \gamma \sum_{i=1}^{n} s(i)^2 \left[\sum_{j=1}^{n} s(j)^2 \left[\sum_{k=1}^{n} s(k)^2 \left[\sum_{l=1}^{n} s(l)^2\right]\right]\right], \text{como } \gamma = \frac{\beta}{(s^T s)^4}$$

$$= \frac{\beta}{(s^T s)^4} (s^T s)^4.$$

$$\therefore \quad V_c \cdot sssss = \beta.$$

La solución a (3.11) es

$$T_c = b \circ s \circ s + s \circ b \circ s + s \circ s \circ b, \tag{3.12}$$

donde la notación $T = u \circ v \circ w$, $u, v, w \in <^n$, $T \in <^{nxnxn}$, es llamado un tensor de tercer orden rango uno para el cual $T(i,j,k) = u(i)v(j)w(k)$. Aquí $b \in <^n$ es el único vector para el cual (3.12) satisface (3.8) , y esta dado por

$$b = \frac{3a(s^T s) - 2s(s^T a)}{3(s^T s)^3}.$$

En efecto, por de…nición

$$(T \cdot ss)(i) = \sum_{j=1}^{n}\sum_{k=1}^{n} T(i,j,k)s(j)s(k)$$

$$= \sum_{j=1}^{n}\sum_{k=1}^{n} [b(i)s(j)s(k) + s(i)b(j)s(k) + s(i)s(j)b(k)]s(j)s(k)$$

$$= \sum_{j=1}^{n}\sum_{k=1}^{n} \left[b(i)s(j)^2 s(k)^2 + s(i)b(j)s(j)s(k)^2 + s(i)s(j)^2 b(k)s(k)\right]$$

$$= \sum_{j=1}^{n} b(i)s(j)^2 \sum_{k=1}^{n} s(k)^2 + s(i)b(j)s(j) \sum_{k=1}^{n} s(k)^2 + s(i)s(j)^2 \sum_{k=1}^{n} b(k)s(k)$$

$$= \sum_{j=1}^{n} \left[b(i)s(j)^2 (s^T s) + s(i)b(j)s(j)(s^T s) + s(i)s(j)^2 (b^T s)\right]$$

$$= b(i)(s^T s)\sum_{j=1}^{n} s(j)^2 + s(i)(s^T s)\sum_{j=1}^{n} b(j)s(j) + s(i)(b^T s)\sum_{j=1}^{n} s(j)^2$$

$$= b(i)(s^T s)^2 + 2s(i)(s^T s)(b^T s), \text{ sustituyendo } b = \frac{3a(s^T s) - 2s(s^T a)}{3(s^T s)^3}$$

$$= \frac{3a(i)\left(s^T s\right) - 2s(i)\left(s^T a\right)}{3\left(s^T s\right)^3}\left(s^T s\right)^2 + 2s(i)\left(s^T s\right)\left[\sum_{i=1}^{n}\frac{3a(i)\left(s^T s\right) - 2s(i)\left(s^T a\right)}{3\left(s^T s\right)^3}s(i)\right]$$

$$= \frac{3a(i)\left(s^T s\right) - 2s(i)\left(s^T a\right)}{3\left(s^T s\right)} + 2s(i)\left(s^T s\right)\left[\frac{3\left(s^T s\right)\left(s^T a\right) - 2\left(s^T a\right)\left(s^T s\right)}{3\left(s^T s\right)^3}\right]$$

$$= \frac{3a(i)\left(s^T s\right) - 2s(i)\left(s^T a\right)}{3\left(s^T s\right)} + 2s(i)\left(s^T s\right)\left[\frac{\left(s^T s\right)\left(s^T a\right)}{3\left(s^T s\right)^3}\right]$$

$$= \frac{3a(i)\left(s^T s\right) - 2s(i)\left(s^T a\right)}{3\left(s^T s\right)} + \frac{2s(i)\left(s^T a\right)}{3\left(s^T s\right)} = \frac{3a(i)\left(s^T s\right)}{3\left(s^T s\right)}.$$

) $(T \cdot ss)(i) = a(i)$.

3.3 Método de Tensor para Matriz Hessiana no Singular

En esta sección se muestra cómo encontrar e…cientemente un minimizador del modelo de tensor (3.2) cuando la matriz hessiana es no singular.

Las sustituciones en (3.2) de los valores de T_c y V_c encontrados en el apartado anterior dan como resultado:

$$M_T(x_c + d) = f(x_c) + \nabla f(x_c) \cdot d + \frac{1}{2}\nabla^2 f(x_c) \cdot d^2 + \frac{1}{2}\left(b^T d\right)\left(s^T d\right)^2 + \frac{\gamma}{24}\left(s^T d\right)^4. \tag{3.13}$$

Ahora se mostrará que el minimizador de (3.13) puede ser reducido a la solución de un polinomio de tercer grado con una incógnita. Para ser más concisos se usará la notación $g = \nabla f(x_c)$ y $H = \nabla^2 f(x_c)$.

Una condición necesaria para que d sea un minimizador local de (3.13) es que la derivada del modelo de tensor con respecto a d sea cero (ver CNPO). Esto es,

$$\nabla M_T(x_c + d) = g + Hd + \left(b^T d\right)\left(s^T d\right)s + \frac{1}{2}\left(s^T d\right)^2 b + \frac{\gamma}{6}\left(s^T d\right)^3 s = 0,$$

despejando d de la ecuación anterior tenemos que,

$$d = -H^{-1}\left[g + \left(b^T d\right)\left(s^T d\right)s + \frac{1}{2}\left(s^T d\right)^2 b + \frac{\gamma}{6}\left(s^T d\right)^3 s\right]. \tag{3.14}$$

Si primero se premultiplica la ecuación 3.14 por s^T en ambos lados, se tiene una ecuación cúbica en las incógnitas $\beta = s^T d$ y $\theta = b^T d$, esto es,

$$
\begin{aligned}
s^T d &= -\left(s^T H^{-1}\left(g + \left(b^T d^2\right)s + \frac{1}{2}\left(s^T d^2\right)b + \frac{\gamma}{6}\left(s^T d^3\right)s\right)\right). \\
\beta &= -\left(s^T H^{-1}\left(g + \theta\beta s + \frac{1}{2}\beta^2 b + \frac{\gamma}{6}\beta^3 s\right)\right). \\
0 &= s^T H^{-1} g + \beta + s^T H^{-1} s\theta\beta + \frac{1}{2}s^T H^{-1} b\beta^2 + \frac{\gamma}{6}s^T H^{-1} s\beta^3. \quad (3.15)
\end{aligned}
$$

Si se premultiplica la ecuación (3.14) por b^T en ambos lados, se obtiene una ecuación cúbica en las incógnitas β y θ, esto es,

$$
\begin{aligned}
b^T d &= -\left(b^T H^{-1}\left(g + \left(b^T d^2\right)s + \frac{1}{2}\left(s^T d^2\right)b + \frac{\gamma}{6}\left(s^T d^3\right)s\right)\right). \\
\theta &= -\left(b^T H^{-1}\left(g + \theta\beta s + \frac{1}{2}\beta^2 b + \frac{\gamma}{6}\beta^3 s\right)\right). \\
0 &= b^T H^{-1} g + \theta + b^T H^{-1} s\theta\beta + \frac{1}{2}b^T H^{-1} b\beta^2 + \frac{\gamma}{6}b^T H^{-1} s\beta^3. \quad (3.16)
\end{aligned}
$$

Luego, de las ecuaciones (3.15) y (3.16) se consigue un sistema de dos ecuaciones cúbicas en dos incógnitas β y θ, las cuales pueden ser resueltas analíticamente.

Ahora se mostrará como calcular las soluciones del sistema de dos ecuaciones con dos incógnitas, calculando las soluciones de una sola ecuación cúbica en una incógnita β.

Sean $u = s^T H^{-1} g, v = s^T H^{-1} b, w = s^T H^{-1} s, y = b^T H^{-1} g, z = b^T H^{-1} b$. Primero se calcula el valor de θ como una función de β usando la ecuación (3.15):

$$
\begin{aligned}
0 &= s^T H^{-1} g + \beta + s^T H^{-1} s\theta\beta + \frac{1}{2}s^T H^{-1} b\beta^2 + \frac{\gamma}{6}s^T H^{-1} s\beta^3. \\
0 &= u + \beta + w\theta\beta + \frac{1}{2}v\beta^2 + \frac{\gamma}{6}w\beta^3. \\
\theta &= -\frac{u + \beta + \frac{1}{2}v\beta^2 + \frac{\gamma}{6}w\beta^3}{w\beta}. \quad (3.17)
\end{aligned}
$$

Hay que notar que el denominador de (3.17) es igual a cero, cuando $\beta = 0$ o $w = 0$. Se asumirá que $\beta \neq 0$; de lo contrario el modelo de tensor se reduce al modelo de Newton(ver ecuación (3.14)). Ahora, si $w = 0$, entonces (3.15) es una ecuación cuadrática en β :

$$
0 = u + \beta + \frac{1}{2}v\beta^2,
$$

y por tanto

$$
\beta = \frac{-1 \pm \sqrt{1 - 2uv}}{v},
$$

y en caso de que $v = 0$ en la ecuación anterior, tomamos $\beta = -u$.

Los minimizadores con valores reales del modelo 3.13 existen solo si $1 - 2uv \geq 0$. Es fácil probar que para que los valores de θ esten bien de…nidos, $1 + u\beta$ no debe ser cero.

Si $\beta \neq 0$ y $w \neq 0$, sustituimos θ dada por (3.17) en la ecuación (3.16), y obtenemos que

$$\begin{aligned}
0 &= b^T H^{-1} g + \theta + b^T H^{-1} s\theta\beta + \frac{1}{2} b^T H^{-1} b\beta^2 + \frac{\gamma}{6} b^T H^{-1} s\beta^3. \\
0 &= y - \frac{u + \beta + \frac{1}{2}v\beta^2 + \frac{\gamma}{6}w\beta^3}{w\beta} + v \left(- \frac{u + \beta + \frac{1}{2}v\beta^2 + \frac{\gamma}{6}w\beta^3}{w\beta} \right) \beta \\
&\quad + \frac{1}{2} z\beta^2 + \frac{\gamma}{6} v\beta^3. \\
0 \neq w\beta &= yw\beta - u - \beta - \frac{1}{2}v\beta^2 - \frac{\gamma}{6}w\beta^3 - uv\beta - v\beta^2 - \frac{1}{2}v^2\beta^3 - \frac{\gamma}{6}vw\beta^4 \\
&\quad + \frac{1}{2} zw\beta^3 + \frac{\gamma}{6} vw\beta^4,
\end{aligned}$$

y agrupando términos se concluye que

$$-u + (yw - uv - 1)\beta - \frac{3}{2}u\beta^2 + \left(\frac{1}{2}wz - \frac{\gamma}{6}w - \frac{1}{2}v^2 \right) \beta^3 = 0, \qquad (3.18)$$

el cual es un polinomio de tercer tercer grado con una incógnita, β. Las raíces de (3.18) son calculadas analíticamente. Se sustituyen los valores de β en (3.17) para calcular los valores de d. El mayor costo de todo este proceso es el cálculo de $H^{-1} g,\ H^{-1} b,\ H^{-1} s$.

3.4 Algoritmo del Método de Tensor

A continuación se presenta el algoritmo de tensor para programación sin restricciones. La implementación sólo se hizo para el caso en que la matriz hessiana es no singular.

Algoritmo 3.3 (Método del tensor) Sean x_c el punto actual, x_+ el siguiente punto, x_- el punto anterior, d_t la dirección de tensor y d_n la dirección de Newton.

1. Evaluar $\nabla f(x_c)$, si $\|\nabla f(x_c)\| < tol$ parar, de lo contrario ir a 2.
2. Evaluar $\nabla^2 f(x_c)$.
3. Calcular b y γ en el modelo de tensor (3.13), de tal manera que el modelo interpole $f(x)$ y $\nabla f(x)$ en x_-.

4. Encontrar un minimizador potencial d_k del modelo de tensor, ecuación (3.13)

   (a) Hacer $\beta_* = \min(|\beta_i|)$,
   con β_i raíces de la ecuación $0 = -u + (yw - uv - 1)\beta - \frac{3}{2}u\beta^2 + \left(\frac{1}{2}wz - \frac{\gamma}{6}w - \frac{1}{2}v^2\right)\beta^3$.

   (b) Hacer $\theta_* = -\dfrac{u + \beta_* + \frac{1}{2}v\beta_*^2 + \frac{\gamma}{6}w\beta_*^3}{w\beta_*}$.

   (c) Hacer $d_t = -\left[\nabla^2 f(x_c)\right]^{-1}\left[\nabla f(x_c) + \theta_*\beta_* s + \frac{1}{2}\beta_*^2 b + \frac{\gamma}{6}\beta_*^3 s\right]$.

5. Revisar si la dirección de tensor es una dirección de descenso,

   Si $\nabla^T f(x_c)\, d_t < 0$

   hacer $d_n = -\left[\nabla^2 f(x_c)\right]^{-1}\nabla f(x_c)$.

6. Calcular el siguiente x_+ usando una búsqueda lineal.

7. $x_c = x_+$, $f(x_c) = f(x_+)$, ir al paso 1.

Capítulo 4

Algoritmos Instrumentados

Los métodos instrumentados fueron: máximo descenso, gradientes conjugados y método de tensor. Todos ellos requieren de una búsqueda lineal para el tamaño de paso en cada iteración, por lo que también se implementó un algoritmo de búsqueda lineal que satisfaciera las condiciones fuertes de Wolfe.

4.1 Búsqueda Lineal

A continuación se presenta el algoritmo empleado para la implementación de la búsqueda lineal. Este algoritmo se utiliza en todos los métodos programados y es la clave para obtener buenos resultados a la hora de calcular el mínimo.

El algoritmo consta de dos partes, en la primera se comienza con un valor inicial y se empieza a iterar hasta que se encuentra un valor aceptable o un intervalo de acotación para el tamaño de paso. En la segunda parte (algoritmo Zoom) se decrementa sucesivamente el tamaño del intervalo obtenido en el paso anterior, hasta que se encuentra un tamaño de paso aceptable. Este algoritmo está basado en el que proponen Nocedal y Wright[6].

Algoritmo 4.1 (Búsqueda lineal) Sean x_c la iteración actual, d la dirección de descenso, c_1 y c_2 los escalares en las condiciones fuertes de Wolfe (ver (2.17) y (2.18)).

1. Hacer $\alpha_0 \leftarrow 0$, $\alpha_1 \leftarrow 1$;
   $f_0 \leftarrow f(x_c)$, $g_0 \leftarrow \nabla f(x_c)$;
   $k \leftarrow 1$;
2. $x_k \leftarrow x_c + \alpha_k d$;
3. $f_k \leftarrow f(x_k)$;
4. Si $\left[f_k > f_0 + c_1 \alpha d^T g_0 \right]$ ó $[f_k \geq f_{k-1}$ y $k > 1]$
   $\alpha_* \leftarrow$ Zoom(α_{k-1}, α_k) y detener.
   de lo contrario continuar;

5. g_k Ã r $f(x_k)$;
6. Si $d^T g_k \cdot c_2 d^T g_0$
   $\alpha_¤$ Ã α_k parar.
7. Si $d^T g_k$ ¸ 0
   $\alpha_¤$ Ã zoom$(\alpha_{k\,¡\,1}, \alpha_k)$ y detener.
   de lo contrario continuar;
8. $\alpha_{k\,¡\,1}$ Ã α_k;
9. α_k Ã $2\alpha_k$;
10. $f_{k\,¡\,1}$ Ã f_k;
11. Hacer k Ã k + 1; ir a 2.

El algoritmo anterior usa el hecho de que el intervalo $(\alpha_{k\,¡\,1}, \alpha_k)$ continene un tamaño de paso que cumple las condiciones fuertes de Wolfe si se satisface alguna de las siguientes tres condiciones:

i) α_k no cumple la condición de su…ciente decrecimiento.

ii) f_k ¸ $f_{k\,¡\,1}$.

iii) $d^T g_k$ ¸ 0.

Todas las llamadas al siguiente algoritmo tienen la forma Zoom$(\alpha_{lo}, \alpha_{hi})$, donde:

a) El intervalo acotado por α_{lo} y α_{hi}, contiene tamaños de paso que satisfacen las condiciones fuertes de Wolfe.

b) α_{lo} satisface la condición de su…ciente decrecimiento y da el más pequeño valor en f.

c) α_{hi} es elegida de tal manera que d^T r $f(x_c + \alpha_{lo} d)(\alpha_{hi} ¡ \alpha_{lo}) \cdot 0$.

En cada iteración Zoom genera una iteración α_j entre α_{lo} y α_{hi}, que después reemplazará alguno de los extremos de tal manera que se sigan satisfaciendo los incisos anteriores.

Algoritmo 4.2 (Zoom) Sea (a_{lo}, a_{hi}) un intervalo que contiene tamaños de paso que satisfacen las condiciones fuertes de Wolfe (2.17)-(2.18).

1. Interpolar (Usando bisección ó función cuadrática) para encontrar el tamaño de paso α_j entre α_{lo} y α_{hi};
2. x_j Ã $x_c + \alpha_j d$;
3. f_j Ã $f(x_j)$;

4. Si(4a) $\left[f_j > f_0 + c_1 \alpha_j d^T g_0 \right]$ ó $f_j \geq f(x_c + \alpha_{lo} d)$

   Hacer $\alpha_{hi} \leftarrow \alpha_j$;

   En caso contrario;

   Hacer $g_j \leftarrow \nabla f(x_j)$;

   Si $d^T g_j \geq c_2 d^T g_0$

   Hacer $\alpha_* \leftarrow \alpha_j$ y detener.

   Fin(si);

   Si $d^T g_j (\alpha_{hi} - \alpha_{lo}) \geq 0$

   Hacer $\alpha_{hi} \leftarrow \alpha_{lo}$;

   Fin(si);

   Hacer $\alpha_{lo} \leftarrow \alpha_j$;

   Fin(si 4a);

5. Hacer $j \leftarrow j + 1$; ir a 1.

4.2 Máximo Descenso

El algoritmo de máximo descenso es uno de los más sencillos de implementar y por eso es uno de los más usados cuando las funciones son más o menos bien comportadas , aunque tiene convergencia lenta comparada con otros métodos. El algoritmo instrumentado es el siguiente

Algoritmo 4.3 (Máximo descenso) Sea x_0 el punto inicial

1. $k \leftarrow 0$;
2. $d_k \leftarrow -\nabla f(x_k)$;
3. Si $\|d_k\|_2 < tol$

   detener.

   Fin(si);

4. Hacer una búsqueda lineal para encontrar el α_k apropiado;
5. $x_{k+1} \leftarrow x_k + \alpha_k d_k$;
6. Hacer $k \leftarrow k + 1$; ir a 2.

4.3 Gradientes Conjugados

Como se mencionó en el capítulo 2, uno de los algoritmos más usados es el de gradientes conjugados, ya que no es necesario calcular la hessiana de la función y es uno de los más e...cientes. El algoritmo instrumentado para el método de gradientes conjugados es el siguiente:

Algoritmo 4.4 (Gradientes conjugados) Sea x_0 el punto inicial

1. k Ã 0;
2. g_0 Ã r f (x_0) ;
3. Si kg_0k$_2 < tol$
   Detener.
   En caso contrario
   d_0 Ã ¡ g_0;
   Fin(si);
4. Hacer una búsqueda lineal para encontrar el α_k apropiado;
5. x_{k+1} Ã x_k + $\alpha_k d_k$;
6. g_{k+1} Ã r f (x_{k+1}) ;
7. Si kg_{k+1}k$_2 < tol$
   Detener.
   Fin(si);
8. Seleccionar el β_k;
9. d_{k+1} Ã ¡ g_{k+1} + $\beta_k d_k$;
10. Hacer k Ã k + 1; ir a 3.

El número β_k en el paso 8 puede ser el de Flecher-Reeves(2.2), Hestenes-Stiefel(2.10) o el de Polak-Ribiere(2.8). Ya que se tiene el mismo algoritmo para cualquier método de gradientes conjugados excepto por el paso 8, es innecesario escribir todos los algoritmos. Así, dependiendo de β_k será como se llame el algoritmo.

4.4 Método de Tensor

A continuación se presenta el algoritmo de tensor para programación sin restricciones. La implementación sólo se hizo para el caso en que la matriz hessiana es no singular.

Algoritmo 4.5 (Método del tensor) Sea x_c el punto actual, x_+ el siguiente punto, x_- el punto anterior, d_t la dirección de tensor y d_n la dirección de Newton.

1. Evaluar $\nabla f(x_c)$,
2. Si $\|\nabla f(x_c)\| < tol$
   Parar.
   Fin(si);
3. Evaluar $\nabla^2 f(x_c)$ y aplicarle factorización de Cholesky para matrices ralas;
4. Calcular b y γ en el modelo de tensor (3.13), de tal manera que el modelo interpole $f(x)$ y $\nabla f(x)$ en x_-;
5. Encontrar un minimizador potencial d_k del modelo de tensor, ecuación(3.13)
   (a) Hacer $\beta_* \leftarrow \min(|\beta_i|)$;
   con β_i raices de la ecuación $0 = -u + (yw - uv - 1)\beta - \frac{3}{2}u\beta^2 + \left(-\frac{1}{2}wz - \frac{\gamma}{6}w - \frac{1}{2}v^2\right)\beta^3$ (3.18);
   (b) Hacer $d_t \leftarrow -\left(\nabla^2 f(x_c)\right)^{-1}\nabla f(x_c) + \theta_*\beta_* s + \frac{1}{2}\beta_*^2 b + \frac{\gamma}{6}\beta_*^3 s$ con
   (c) $\theta_* = -\frac{\left(u + \beta_* + \frac{1}{2}v\beta_*^2 + \frac{\gamma}{6}w\beta_*^3\right)}{w\beta_*}$;
6. Ver si la dirección de tensor es una dirección de descenso
   (a) Si $\nabla^T f(x_c)\, d_t > 0$
   $d_n \leftarrow -H^{-1}\nabla^T f(x_c)$;
   Fin(si);
7. Calcular el siguiente $x_+ = x_c + \alpha d$ usando una búsqueda lineal, con $d = d_t$ o $d = d_n$;
8. $x_c \leftarrow x_+$, $f(x_c) \leftarrow f(x_+)$, ir al paso 1.

El procedimiento para calcular b y γ en el paso 3 se discutió en la sección 3.2. En el paso 3, a la matriz hessiana H se le aplica una descomposición de Cholesky de tal manera que siga siendo de…nida positiva y simétrica, y así proceder a aplicar la dirección de Newton. El algoritmo utilizado es el que viene incluido en MATLAB.

El algoritmo de Cholesky factoriza una matriz simétrica A casi de…nida positiva en una matriz triangular L de tal manera que $LL^T = A$, y LL^T es de…nida positiva. Si la matriz ya es de…nida positiva el algoritmo de Cholesky no la afecta.

Capítulo 5

Software y Pruebas Realizadas

Este capítulo esta dedicado al software creado para comparar los métodos antes descritos y los problemas prueba que en el se implementaron. Recordar que el método de tensor es un método que trata de obtener mejores resultados que los otros métodos y esta diseñado para funciones con hessiana ralas, esto es, que tienen muchos ceros. Se muestran pruebas de dimensión n = 2, n = 4, n = 5000 y los resultados obtenidos.

5.1 OPTIMIN

Se creó el programa OPTIMIN para resolver el problema de programación no lineal sin resticciones:

$$\min_{x \in <^n} f : <^n \; ! \; <,$$

donde al menos f 2 C^2 con gradiente y hessiana analíticos.

Todas las pruebas se realizaron en OPTIMIN, en el cual se implementan los métodos de máximo descenso, gradientes conjugados (Hestenes-Stiefel, Polak-Ribiere y Flecher-Reeves) y el método de tensor, descritos en este trabajo. Todos los algoritmos instrumentados requieren de gradiente y hessiana analíticos.

Toda la implementación de OPTIMIN se desarrolló en el siguiente software:

software: MATLAB[1]
versión: 6.1.0.450

La instrumentación de OPTIMIN se realizó en una computadora HP Notebook con las siguientes características:

sistema operativo: Microsoft Windows XP Home edition versión 2002;
procesador: mobil AMD athlon(tm) 4;

[1] MATrix LABoratory, por sus siglas en Ingles.

velocidad: 950 Mhz;
disco duro: 19 GB;
Ram: 256 MB en Memoria Ram.

La creación de OPTIMIN tiene como objetivo comparar los métodos descritos en este trabajo mediante problemas pruebas preestablecidos. Tiene la característica de gra…car el comportamiento de los métodos para dar una comparación visual de los mismos, lo cual lo hace una herramienta buena para dar resultados inmediatos. Si lo que se desea es mostrar resultados más precisos de funciones prueba, el software cuenta con la opción de salvar los resultados del método empleado. Dichos resultados son: nombre de la función, número de variables, punto inicial, método empleado, evaluación …nal de f, número de evaluaciones de la función y del gradiente, tiempo utilizado y el número de iteraciones.

OPTIMIN cuenta con una interfaz grá…ca amigable que permite al usuario trabajar comodamente en problemas de programación sin restricciones, se recomienda ver el manual de usuario del siguiente capítulo para una mayor comprensión del software.

5.2 Pruebas

A continuación se muestran los resultados obtenidos al implementar tres problemas prueba en el software antes descrito.

En la primer y tercer prueba se utilizaron problemas del libro de Dennis y Schnabel[4, pags 361-363], son problemas prueba para algoritmos y programas de programación sin restricciones. La función en la segunda prueba pertenece a los problemas prueba del CUTE[7], y esta creado para probar la e…ciencia de los algoritmos de programación sin restricciones.

Las columnas de las tablas 5.1 - 5.3 tienen el siguiente signi…cado:

Método : nombre del método instrumentado.
f (x) inicial : valor inicial de la función objetivo.
f (x) ¯nal : valor …nal de la función objetivo.
nf : número de evaluaciones de la función.
ng : número de evaluaciones del gradiente.
iter: número de iteraciones.
Tiempo : tiempo realizado por el algoritmo, medido en segundos.

5.2.1 Función de Rosenbrock

La primera prueba se realizó con la función de Rosenbrock, cuya ecuación es la siguiente:

$$f(x) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2, \tag{5.1}$$

esta función tiene como minimizador a $x = (1,1)$, ya que $\nabla f(1,1) = 0$ y $\nabla^2 f(1,1) > 0$ y se satisface el teorema 1.9.

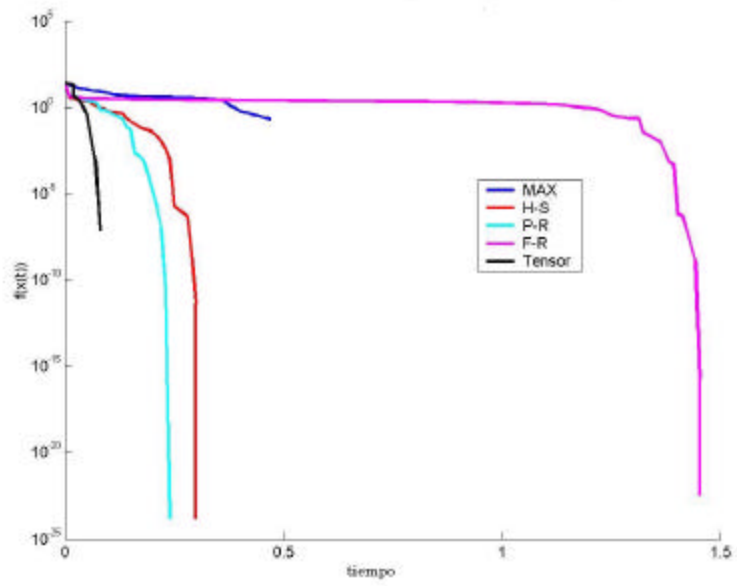


Figura 5.1: Comportamiento grá…co de los métodos para la función de Rosenbrock.

$M\acute{e}todo$	$f(x)\ inicial$	nf	ng	$iter$	$Tiempo$	$f(x)\ final$
MAX	24.2	609	338	200	0.47	0.2056
H-S	24.2	426	307	32	0.30	1.465 ± 10^{-24}
P-R	24.2	322	237	30	0.24	1.495 ± 10^{-24}
F-R	24.2	1906	1333	79	1.452	3.408 ± 10^{-23}
Tensor	24.2	89	68	15	0.16	0

Tabla 5.1: Resultados de los métodos implementados a la función de Rosenbrock

En la figura 5.1 se graficó el comportamiento de cada uno de los métodos, aplicados a la función de Rosenbrock, con respecto al tiempo de ejecución de los mismos. Se puede observar que el método de tensor obtuvo mejores resultados con respecto a los otros métodos, y se ve claramente que todos los métodos convergen al mínimo, excepto el método de máximo descenso ya que es el más lento de todos ellos y requiere de más iteraciones para alcanzar el mínimo.

Todos los métodos utilizan como punto inicial $x_0 = (-1.2, 1)$, un número máximo de iteraciones de 200 y como criterio de parada que $\|\nabla f(x_k)\| < 10^{-10}$. Los datos obtenidos por cada uno de los métodos se muestran en la tabla 5.1.

5.2.2 Función de Rosenbrock Extendida

La segunda prueba se realizó con la función de Rosenbrock extendida de dimensión $n = 5000$ que está definida como sigue:

$$
\begin{aligned}
&n = 2k,\ k \in \mathbb{Z}^+ \\
&Para\ i = 1 : \tfrac{n}{2} \\
&\qquad f_{2i-1}(x) = 10\left(x_{2i} - x_{2i-1}^2\right) \\
&\qquad f_{2i}(x) = 1 - x_{2i-1} \\
&Fin(para). \\
&f(x) = \sum_{i=1} f_i(x)^2
\end{aligned}
$$

Esta función tiene como mínimizador a $x^* = (1, 1, \ldots, 1)$. Esta prueba se desarrolló con este tipo de función debido a que se quiere mostrar la eficiencia del método cuando la dimensión es grande y la hessiana es rala[2].

En la figura 5.2 se muestra el comportamiento de los métodos aplicados a esta función. Es claro que el método que obtuvo mejores resultados es el del tensor con un tiempo de 120.453 segundos, siguiendo el de Polak-Ribiere con un tiempo de 255.328 segundos. El método de tensor es el único que alcanza el minimizador $x^* = (1, 1, \ldots, 1)$. Aunque el método de Hestenes-Stiefel obtuvo buenos resultados, su tiempo de 1245.2 segundos le resta eficiencia.

Todos los métodos utilizan como punto inicial $x_0 = 100(-1.2, 1, \ldots, -1.2, 1)$, un número máximo de iteraciones de 200 y criterio de parada $\|\nabla f(x_k)\| <$

[2] Sparse en inglés.

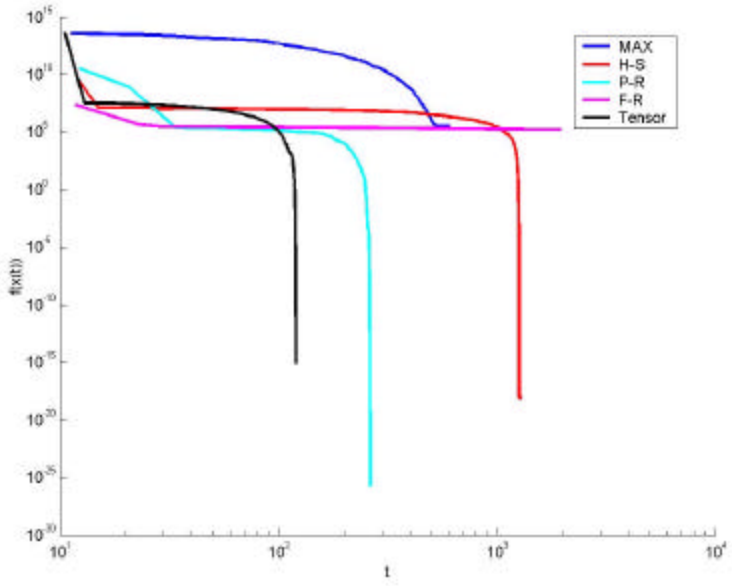


Figura 5.2: Grá…ca del comportamiento de los métodos aplicados a la función de Rosenbrock extendida de dimensión n = 5000.

$M\P todo$	$f(x)\ inicial$	nf	ng	$iter$	$Tiempo$	$f(x)\ final$
MAX	5.1123 £ 10^{13}	3314	2112	200	572.7030	305579
H-S	5.1123 £ 10^{13}	7069	4668	187	1245.2	0
P-R	5.1123 £ 10^{13}	1454	1021	68	255.328	1.837 £ 10^{-26}
F-R	5.1123 £ 10^{13}	10756	7324	200	1858.4	164527
Tensor	5.1123 £ 10^{13}	633	566	147	120.453	0

Tabla 5.2: Resultados de los métodos aplicados a la función de Rosenbrock extendida con n=5000.

$M\P todo$	$f(x)\ inicial$	nf	ng	$iter$	$Tiempo$	$f(x)\ final$
MAX	1.9192 £ 10^{4}	1064	651	200	0.8110	7.877
H-S	1.9192 £ 10^{4}	1647	1201	142	1.2320	1.247 £ 10^{-23}
P-R	1.9192 £ 10^{4}	1582	1187	178	1.2210	3.598 £ 10^{-22}
F-R	1.9192 £ 10^{4}	6363	4391	200	4.2860	11.3671
Tensor	1.9192 £ 10^{4}	202	154	29	0.3210	0

Tabla 5.3: Resultados de los métodos aplicados a la función wood

10^{-10}. Los datos obtenidos por cada uno de los métodos se muestran en la tabla 5.2.

5.2.3 Función Wood

Esta tercer prueba se realizó con la función wood, tomada del libro de Dennis y Schnabel[4, pag 363], dada por

$$\begin{aligned} f(x) &= 100\left(x_1^2 - x_2\right)^2 + (1 - x_1)^2 + 90\left(x_3^2 - x_4\right)^2 + (1 - x_3)^2 \\ &\quad + 10.1\left[(1 - x_2)^2 + (1 - x_4)^2\right] + 19.8\,(1 - x_2)(1 - x_4), \end{aligned}$$

con punto inicial $x_0 = (-3, -1, -3, -1)$ y minimizador $x^* = (1, 1, 1, 1)$.

La figura 5.3 muestra que el método de tensor es el que converge más rápido al mínimo. El método de tensor es el único que alcanza a $x^* = (1, 1, 1, 1)$. Todos los métodos utilizan como punto inicial $x_0 = (-3, -1, -3, -1)$, un número máximo de iteraciones de 200 y como criterio de parada que $\|\nabla f(x_k)\| < 10^{-10}$. Los datos obtenidos por cada uno de los métodos se muestran en la tabla 5.3.

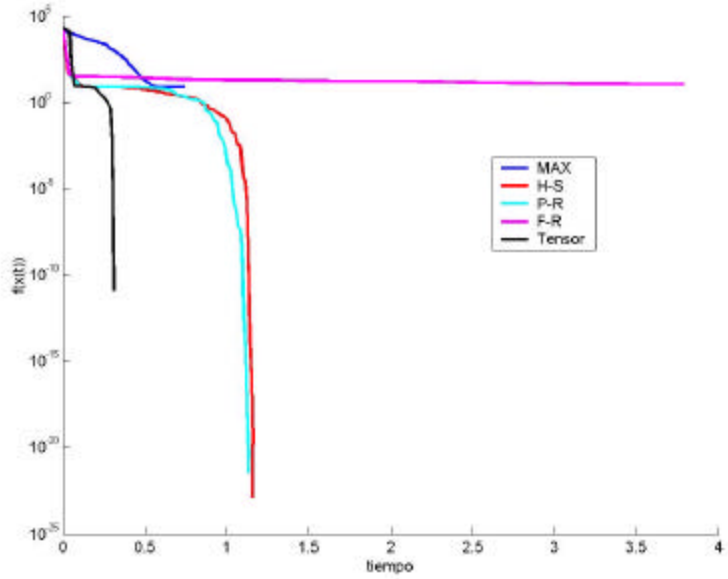


Figura 5.3: Grá…ca de los métodos aplicados a la función wood.

Capítulo 6

Conclusiones

Para mostrar la e…ciencia de los métodos programados se utilizaron dos tipos de problemas prueba, cuyos resultados se mostraron en el capítulo anterior. Estos tipos de problemas fueron seleccionados debido a que:

(a) Tienen matriz hessiana no singular en el minimizador.

(b) Son problemas diseñados para probar algoritmos.

(c) Uno de ellos tiene matriz rala, cuando la dimensión es grande.

Para medir la e…ciencia de los métodos, se utilizaron los siguientes parámetros:

1. Número de evaluaciones de la función.
2. Número de evaluaciones del gradiente.
3. Tiempo realizado por el algoritmo.

Aunque en las tablas del capítulo anterior se muestra el número de iteraciones que realizan los algoritmos, estos no se utilizaron como medida de convergencia dado que el número de iteraciones es independiente de la convergencia del algoritmo, como se pudo comprobar en la función Rosenbrock extendida, en la que el método de tensor realiza 147 iteraciones mientras que el de Polak-Ribiere 68, pero el tiempo que realiza el método de tensor es de 120.453 segundos contra 255.328 segundos del de Polak-Ribiere.

$M¶todo$	$nf\ total$	$ng\ total$	$Tiempo\ total\ (seg)$
H-S	9142	6176	1246, 732
P-R	3358	2445	256.789
Tensor	924	788	120.934

Tabla 6.1: Resultados totales de los métodos

La tabla 6.1 muestra los resultados totales de los métodos aplicados a los problemas prueba del capítulo anterior. Aquí se muestran el número de evaluaciones totales de la función($nf\ total$), del gradiente($ng\ total$) y el tiempo total empleado en resolver los problemas prueba. Sólo se muestran los métodos que mejores resultados obtuvieron.

De la tabla 6.1 se concluye que:

- En los problemas prueba realizados, el método de tensor es el que menos tiempo utiliza.
- El método de tensor tiene menor número de evaluaciones del gradiente y de la función.
- El segundo mejor método es el de Polak-Ribiere, como se había mencionado en el capítulo 3.
- El método de tensor es el que más cerca está del mínimo de las funciones prueba instrumentadas, según las tablas 5.1 - 5.3.
- El método de tensor funciona para problemas de gran dimensión (n = 5000).

Algunas de las de las caracteristicas del software OPTIMIN son:

- Funciona únicamente con gradiente y hessiana analítico.
- Permite escribir nuevos problemas en un formato preestablecido (ver manual de usuario).
- Salva los resultados de los métodos en archivos que se pueden manipular.

Capítulo 7

Manual de Usuario

En este capítulo se explica cómo operar OPTIMIN, el funcionamiento de cada una de las partes de las que esta compuesto y el tipo de formato de entrada. Al …nal se muestra un ejemplo para aclarar bien su uso.

OPTIMIN resuelve problemas de la forma

$$\min_{x \in <^n} f : <^n \; ! \; <, \tag{7.1}$$

donde f es al menos dos veces continuamente diferenciable. Utiliza gradiente y hessiana analíticos.

La implementación del programa OPTIMIN se realizó en MATLAB 6.1.0.450 y los requerimientos mínimos del programa son:

- Procesador Pentium.
- Microsoft Windows 95.
- Unidad de CD-ROM (para instalar el programa OPTIMIN).
- 64 MB en RAM mínimo, 128 MB RAM recomendable.

NOTA:Debido a que el programa OPTIMIN cuenta con interfaz grá…ca, este no pude trabajar con versiones anteriores a MATLAB 6.1.0.450.

Otros requerimientos recomendables son:

- Microsoft Word 7.0 (O¢ce 95), o versiones más recientes. Esto es necesario para correr MATLAB Notebook.
- Espacio disponible para guardar datos al correr OPTIMIN.

Se puede trabajar con OPTIMIN de dos maneras:

- Desde el CD, para trabajar con los problemas prueba que ya tiene instalados.
- En la PC (para esto se requiere copiar la carpeta optimizacion del CD de instalación del programa).

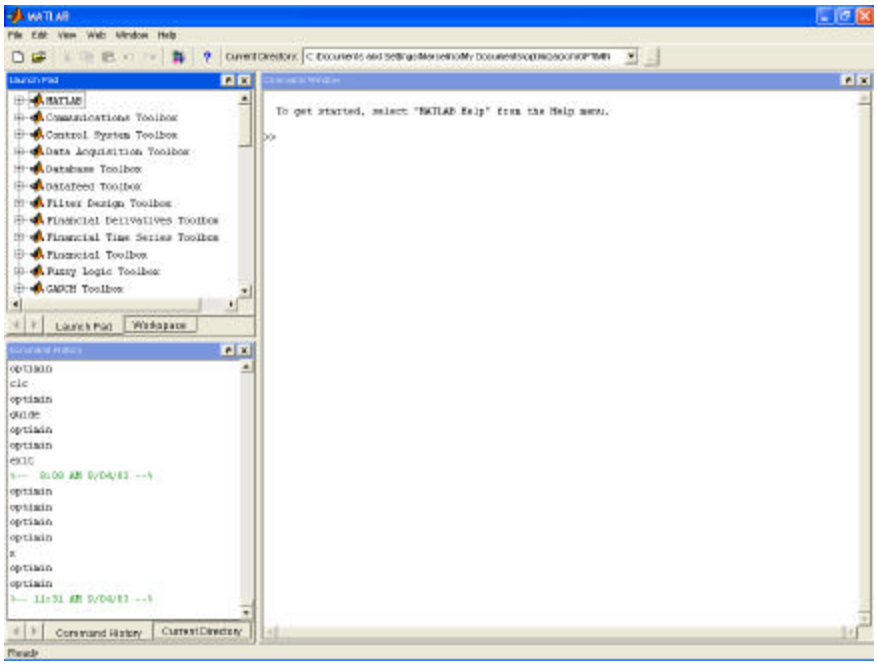


Figura 7.1: Ventana principal de MATLAB 6.0

7.1 Ejecución y Funcionamiento

Para ejecutar OPTIMIN es necesario primero ejecutar MATLAB. Ya activado MATLAB mostrará una ventana como la de la …gura 7.1. Para aquellos que estén familiarizados con MATLAB ir a la sección 7.1.1 página 40, para empezar a trabajar en OPTIMIN.

Para empezar a trabajar con OPTIMIN es necesario cambiarse al directorio donde se encuentra, por lo que en Current Directory hay que darle la dirección donde está el programa. Por ejemplo, si OPTIMIN está en C:nDocuments and SettingsnMarcelinonMy DocumentsnOptimizacion, poner dicha dirección en Current Directory, como lo muestra la …gura 7.2.

Una manera más fácil de dar el directorio de trabajo es como lo muestra la …gura 7.3, donde hay que hacer click en el ícono donde están los tres puntos [. . .] y buscar la carpeta donde se encuentre OPTIMIN, seleccionarla y dar Ok.

7.1.1 Ventana Principal de OPTIMIN

Una vez que se tiene la dirección de trabajo correcta, teclear en el Command Window de MATLAB:

>>optimin Ã›

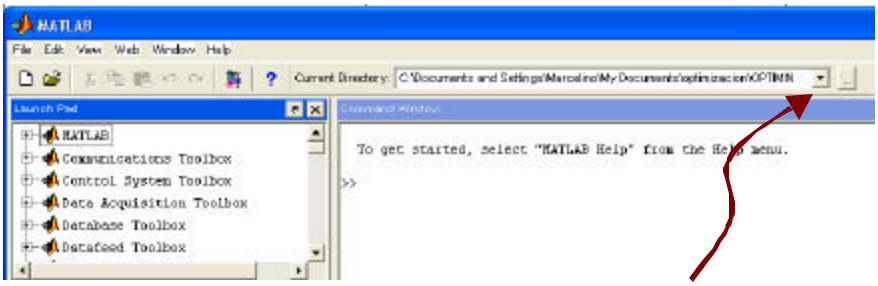


Figura 7.2: En el Current Directory se da la dirección de la carpeta donde se va a trabajar.

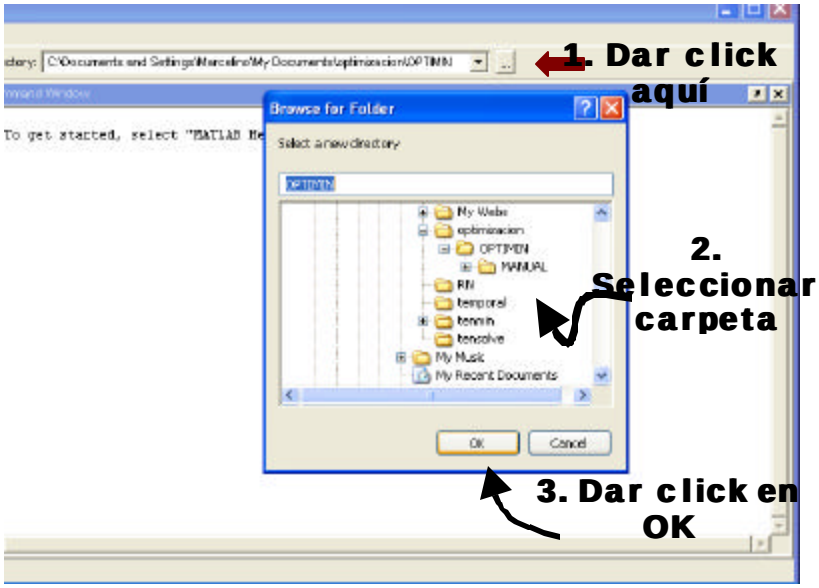


Figura 7.3: Muestra cómo accesar a la dirección de trabajo.

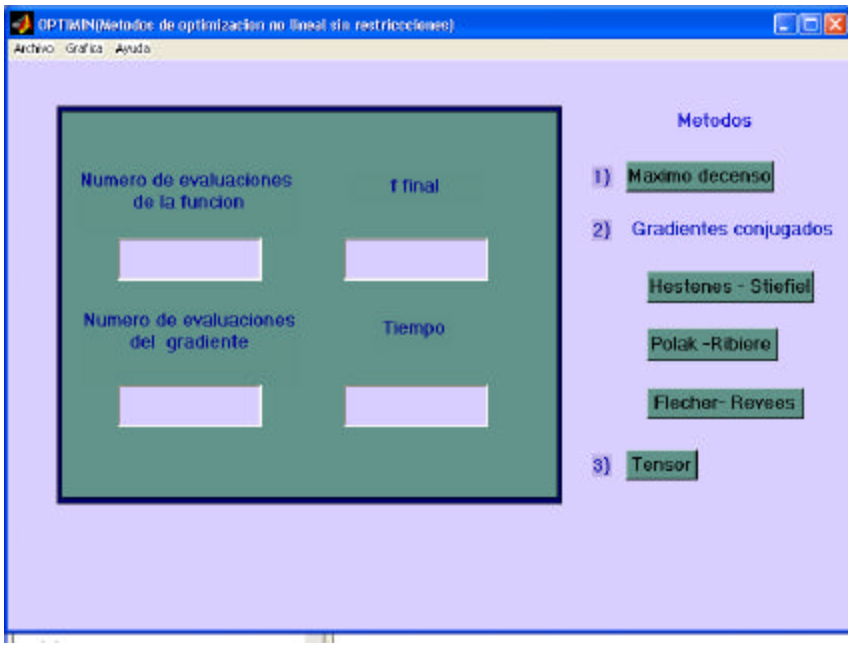


Figura 7.4: Ventana principal de OPTIMIN.

A continuación aparecerá una ventana como la de la …gura 7.4. Dicha ventana contiene los Menús: Archivo, Gra…ca[1] y Ayuda; así como los iconos de los métodos programados y los campos de texto donde se muestran los resultados después de implementar el método seleccionado.

Cuando se inicia OPTIMIN los iconos de los métodos programados se encuentran desactivados ya que aún no se ha declarado ninguna función a minimizar; por lo que para activarlos es necesario dar una función en el menú Archivo.

En la …gura 7.4, los campos de texto que se encuentran en la parte inferior de Numero de evaluaciones de la funcion, f …nal, Numero de evaluaciones del gradiente, y Tiempo, nos indican cual es el número de evaluaciones de la función f que hizo el método instrumentado, el valor de f en el punto …nal, el número de evaluaciones del gradiente $\nabla f(x)$ y el tiempo que utilizó dicho método, respectivamente.

7.1.2 Menú Archivo

El menú Archivo de la ventana principal (ver …gura 7.5), permite:

⋆ Introducir los datos de una función.

[1] MATLAB no permite acentuar las palabras.

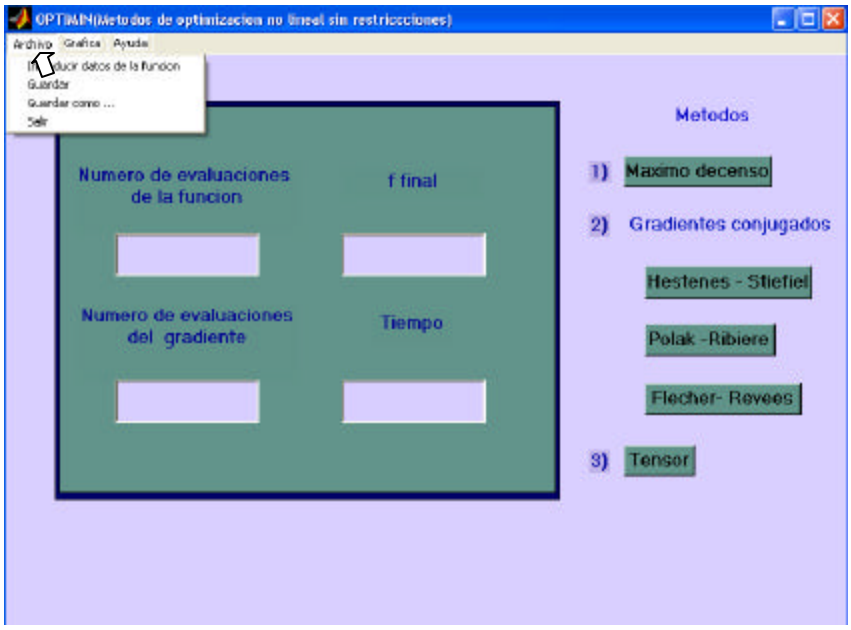


Figura 7.5: Menú Archivo.

⋆ Guardar los resultados obtenidos al correr algún algoritmo.

⋆ Salir del programa. (Sólo hacer click en Salir para cerrar OPTIMIN).

Insertar los datos de una Función

Al seleccionar Introducir datos de la funcion en el menú Archivo, aparecerá una ventana como la de la …gura 7.6, donde esta declarada automáticamente la función de Rosenbrock extendida (ver página 32) como una función prueba.

Si desea minimizar la función prueba dada, sólo es necesario:

1. Dar el número de variables (que para este caso es un número par).
2. El punto inicial a usar: $x_0, 10x_0$ o $100x_0$, con $x_0 = [\,¡\,1.2, 1, \ldots, ¡\,1.2, 1]$.
3. El número máximo de iteraciones.
4. Hacer click en Aplicar.

Cuando ya se han dado los datos correctamente, los botones de los métodos (en color verde) se activan, permitiendo la búsqueda del mínimo al hacer click en ellos.

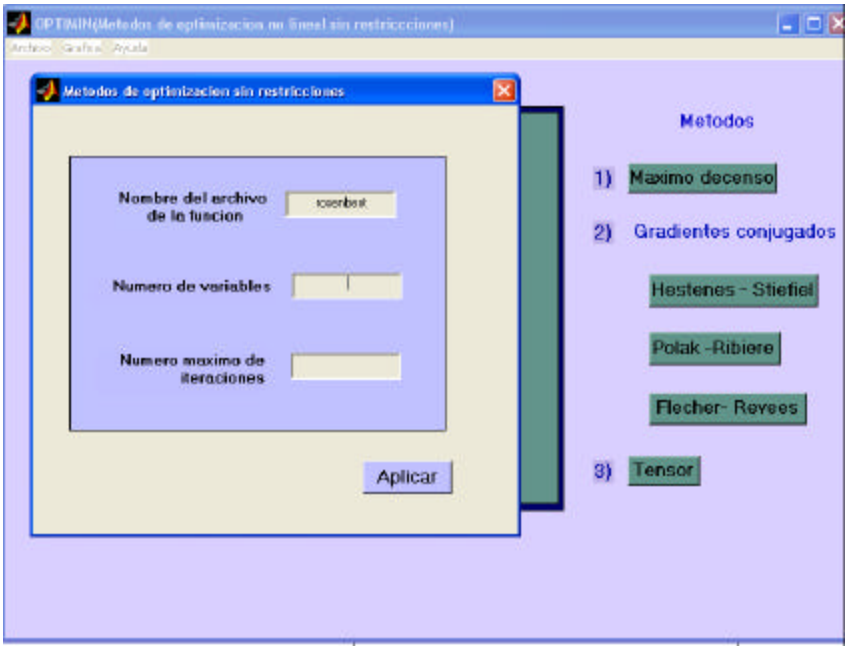


Figura 7.6: Ventana que permite introducir los datos de la función a minimizar.

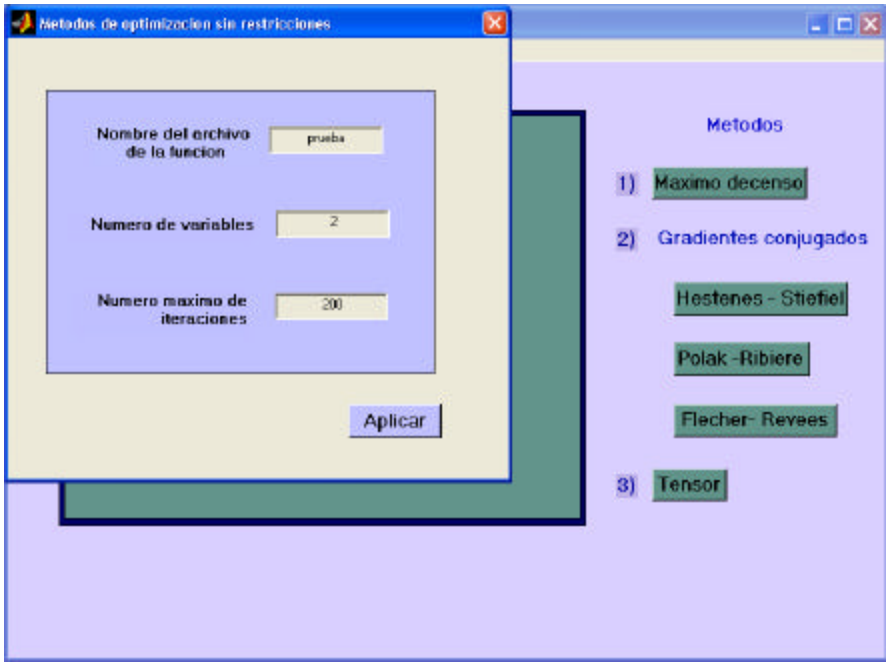


Figura 7.7: Ejemplo de como se dan los datos de una función a minimizar.

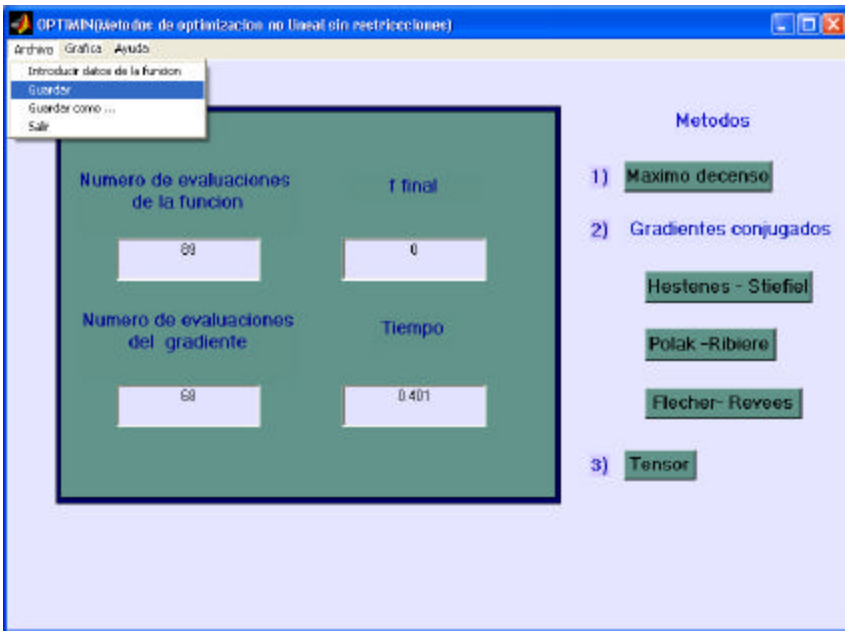


Figura 7.8: Al seleccionar Guardar, OPTIMIN almacena los resultados del método en saloioptimin.mat.

Si lo que se desea es encontrar el mínimo de una función que no esta aún declarada, es necesario escribir en los campos de texto (…gura 7.6) el nombre del archivo ¤.m (previamente escrito en MATLAB), el número de variables y el número máximo de iteraciones de…nidos para esa función, como lo muestra la …gura 7.7.

Para saber como escribir los archivos ¤.m de las funciones que utilizará OPTIMIN, ver el apartado Declaración de funciones a minimizar página 50 de esta misma sección.

Cuando ya se haya dado toda la información necesaria, hacer click en el método que se desee para empezar a encontrar el mínimo de la función dada.

Guardar información

Una vez que ya se haya corrido algún método y se tenga la información proporcionada por el mismo, ya se puede guardar dicha información si se desea. La …gura 7.8 nos muestra cómo podemos almacenar dicha información.

Al dar click en la opción Guardar, OPTIMIN automáticamente guarda la información generada por el algoritmo elegido en un archivo llamado salioptimin.mat, donde se encuentra una variable de tipo struct llamada salida que es donde está la información.

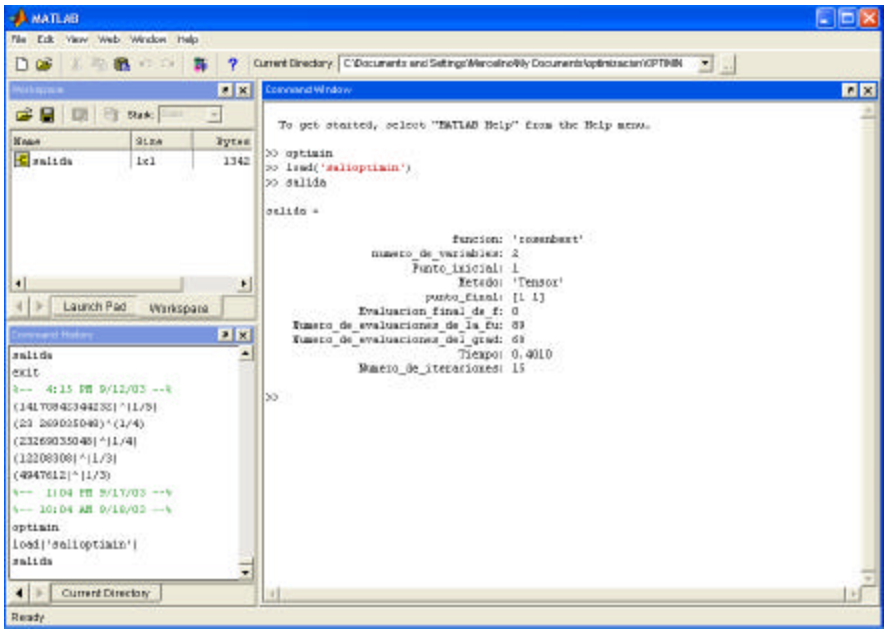


Figura 7.9: Muestra la forma de accesar a los resultados almacenados.

Si se desea accesar a dicha información, en el Command Window de MATLAB escribir:

>> load('salioptimin') Ã›

Esto hace que MATLAB cargue la información almacenada en salioptimin. Para accesar a dicha información escribir:

>> salida Ã› para lo cual aparecerá algo similar a lo de la …gura 7.9.

La opción Guardar como ... del menú Archivo (ver …gura 7.8) realiza lo mismo que la opción salvar antes descrita, excepto que esta última permite que el usuario asigne el nombre que él desee al archivo donde se guardará la información. Para accesar a dicha información se hace lo mismo que en la opción salvar, solo que en este caso hay que escribir:

>> load('nombre del archivo') Ã› y después

>>salida Ã›

7.1.3 Menú Grá…ca

Una vez que ya se tiene la información proporcionada por algún algoritmo, podemos gra…car el comportamiento del mismo en el tiempo. Es decir, el tiempo que se requirió para encontrar el punto x^{α} se particiona en intervalos (t_i, t_{i+1}) y en cada extremo del intervalo se calcula $f(x_{t_i})$, $i = 0, 1, 2, \ldots$. Para hacer todo esto solo es necesario hacer click en Ver gra…ca del metodo del menú Gra…ca (ver …gura 7.10) para que aparezca la grá…ca (ver …gura 7.11).

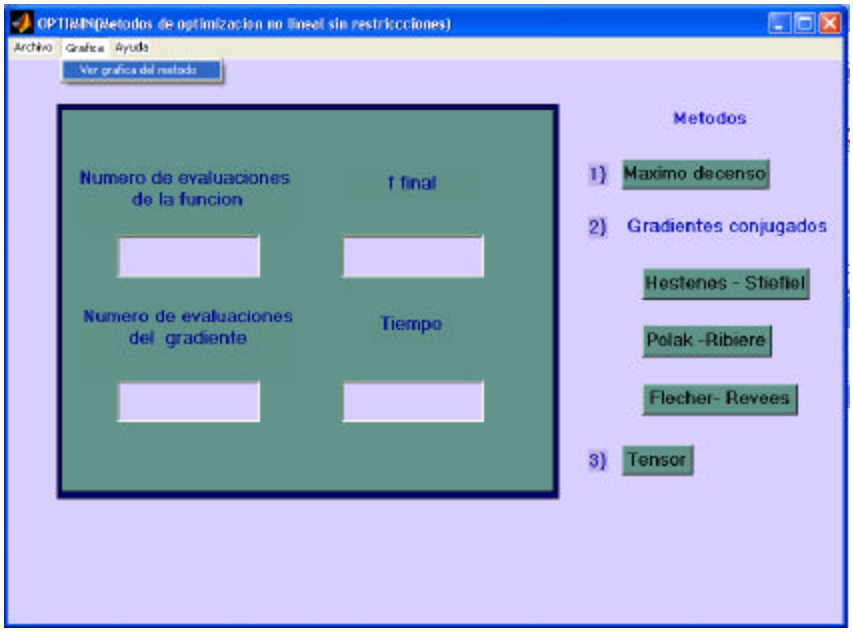


Figura 7.10: La opción Ver gra…ca del metodo muestra el comportamiento del algoritmo (implementado) en el tiempo.

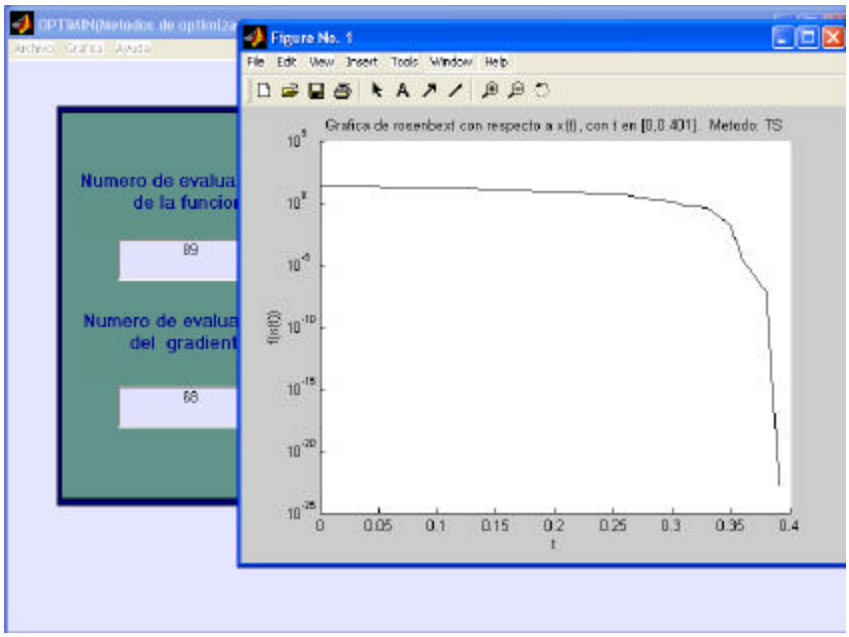


Figura 7.11: Grá…ca del comportamiento del método implementado.

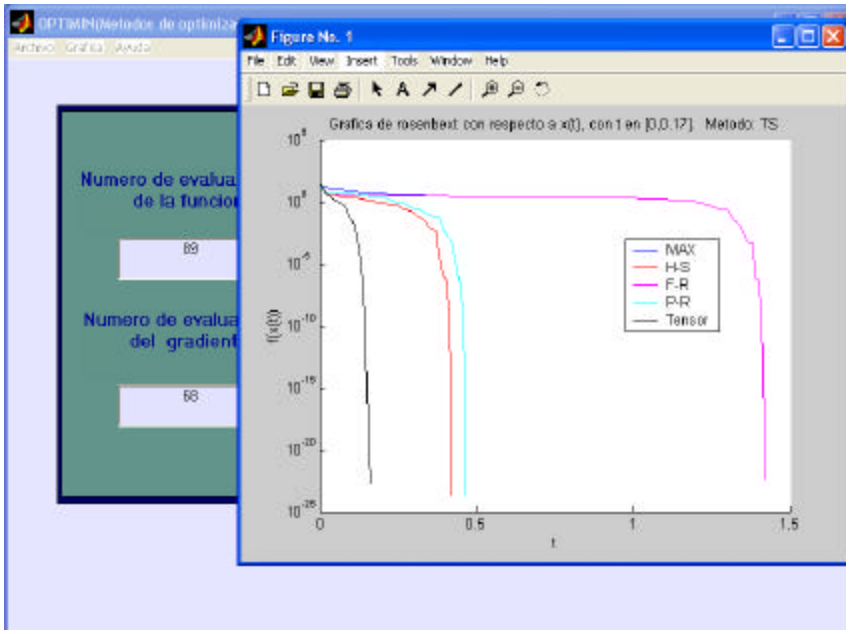


Figura 7.12: Grá…ca de los métodos implementados.

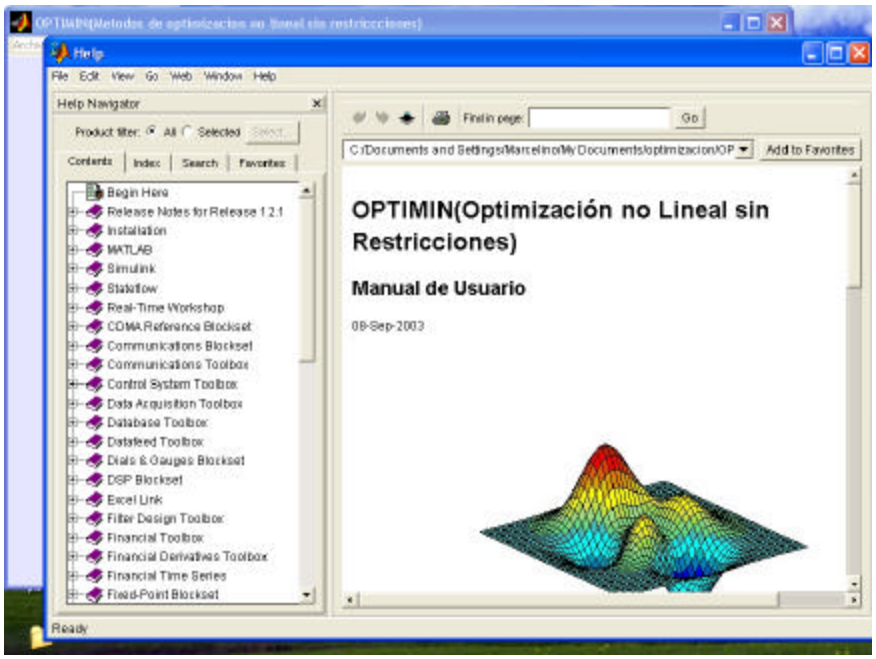


Figura 7.13: El menú Ayuda permite al usuario tener ayuda en línea.

Si no se cierra la ventana que muestra la grá…ca del método y cada vez que se corra un método se gra…ca éste, se puede obtener una grá…ca como la de la …gura 7.12. La cual se muestra con el eje y en forma logarítmica.

7.1.4 Menú Ayuda

El menú Ayuda contiene el manual de usuario descrito en este capítulo y su objetivo es proporcionar ayuda en línea, ya que es más cómodo y fácil de accesar porque esta incluido en el help de MATLAB. Cuando se hace click en el menú Ayuda, aparece una ventana como la que muestra la …gura 7.13.

7.2 Declaración de funciones a minimizar

Para minimizar una función que no ha sido declara antes es necesario escribir un archivo de tipo ¤.m en MATLAB. Para lo cual hay que realizar los siguientes pasos:

1. Abrir el M-…le editor(en caso de no estar abierto): en MATLAB hacer click en el ícono que señala la ‡echa roja de la …gura 7.14, la cual abrirá una ventana como la que muestra la …gura 7.15 (este es el M-…le

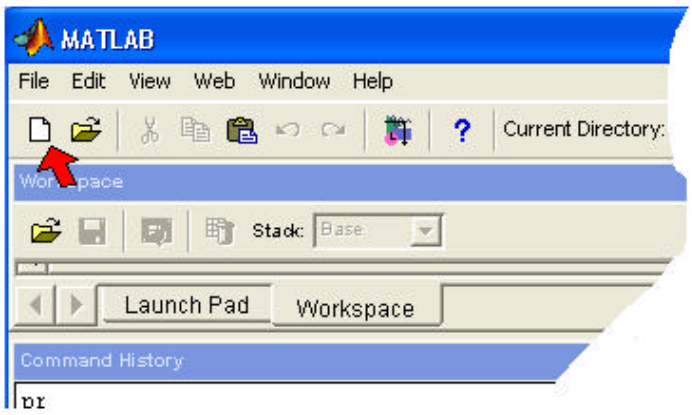


Figura 7.14: Dar click donde señala la ‡echa roja, para crear un nuevo archivo ¤.m.

editor). Una manera equivalente de hacer esto es la siguiente: en el archivo File de MATLAB elegir New y después M-…le.

2. Abrir la función rosenbext(Rosenbrock extendida): presionar el ícono que muestra la …gura 7.16, el cual permitirá abrir cualquier archivo tipo ¤.m. En este caso se requiere abrir el archivo rosenbext. El código de está función se muestra en la …gura 7.17.

3. Copiar todo el código del archivo rosenbext.m a el nuevo archivo, como lo muestra la …gura 7.18. Hay que tener mucho cuidado de NO modi…car ninguna parte del codigo de rosenbext, ya que de hacerlo puede suceder que esta función deje de operar correctamente.

4. Al revisar el código copiado al nuevo archivo, se puede notar que existen cuatro partes donde se enuncia

   %

   % CORTE AQUÍ.

   %

   %

   Cada una de ellas corresponde a la de…nición de: punto inicial, función a minimizar, gradiente de la función y hessiana de la función. Todas esas partes deben de ser reemplazadas por las de…niciones de la nueva función a minimizar.

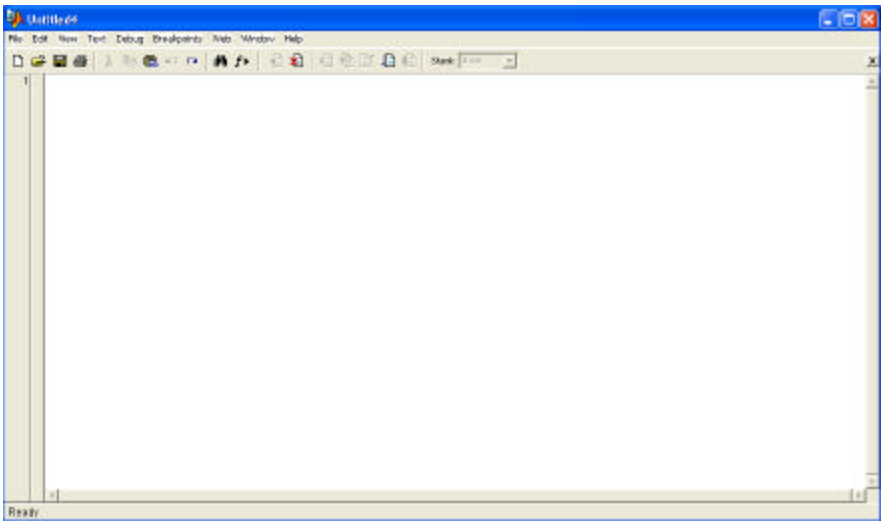


Figura 7.15: M-…le editor.

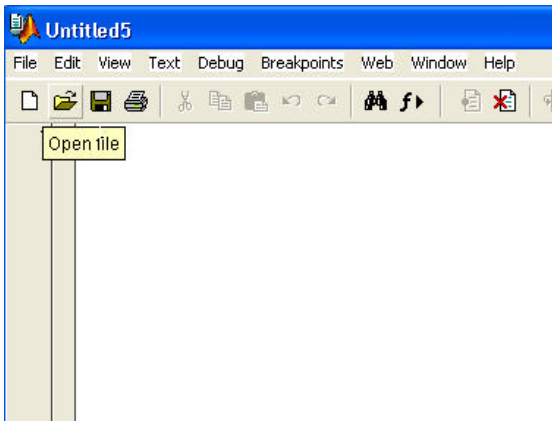


Figura 7.16: Este icono permite abrir archivos tipo ¤.m.

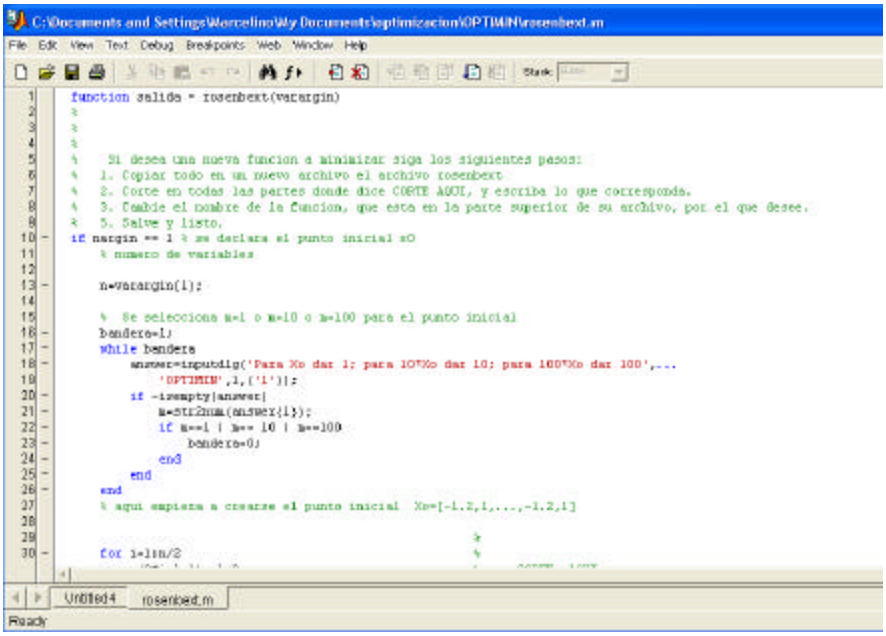


Figura 7.17: Función rosenbext.m que nos servirá de base para que se creen nuevos archivos de funciones a minimizar.

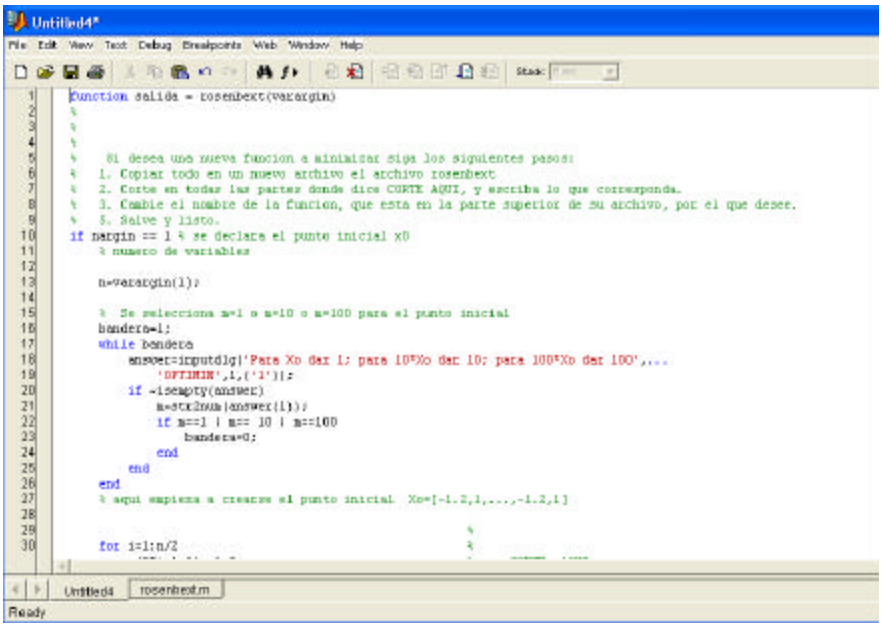


Figura 7.18: Código del archivo rosenbext.m insertado en el nuevo archivo.

Supóngase por ejemplo que se quiere minimizar la función:

$$f(x_1, x_2) = \frac{5}{3}x_1^6 + 4x_1^4 + (x_1 - 2x_2)^2 + 4x_2^4, \tag{7.2}$$

con punto inicial $x_0 = (4.7, -0.9)$.

El gradiente y la hessiana de la función 7.2, son respectivamente:

$$\nabla f(x_1, x_2) = \begin{bmatrix} 10x_1^5 + 16x_1^3 + 2(x_1 - 2x_2) \\ -4(x_1 - 2x_2) + 16x_2^3 \end{bmatrix}, \quad y$$

$$\nabla^2 f(x_1, x_2) = \begin{bmatrix} 50x_1^4 + 48x_1^2 + 2 & -4 \\ -4 & 48x_2 + 8 \end{bmatrix}.$$

Donde aparece el primer CORTE AQUÍ, se elimina esa parte de código para dar el punto inicial, por lo que escribiremos

x=[4.7;-0.9]; ver la figura 7.19.

Hay que mencionar que el punto inicial depende del tipo de función y de la dimensión de la misma. Para el ejemplo de Rosenbrock extendida el número de variables viene dada por n en el codigo de rosenbext.m. Se suguiere dar una buena revisada a dicho archivo para familiarizarce con la creación de funciones a minimizar.

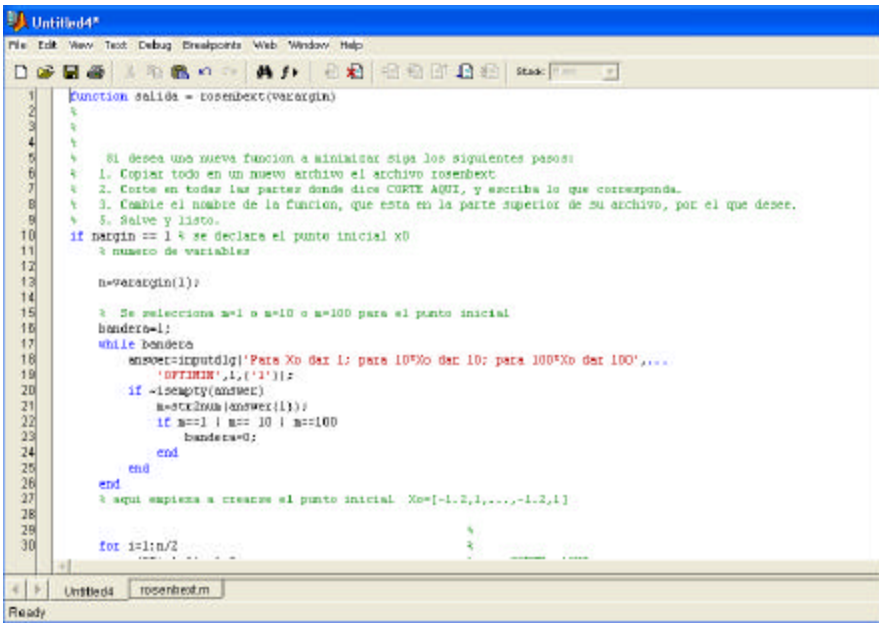


Figura 7.18: Código del archivo rosenbext.m insertado en el nuevo archivo.

Supóngase por ejemplo que se quiere minimizar la función:

$$f(x_1, x_2) = \frac{5}{3}x_1^6 + 4x_1^4 + (x_1 - 2x_2)^2 + 4x_2^4, \qquad (7.2)$$

con punto inicial $x_0 = (4.7, -0.9)$.

El gradiente y la hessiana de la función 7.2, son respectivamente:

$$\nabla f(x_1, x_2) = \begin{bmatrix} 10x_1^5 + 16x_1^3 + 2(x_1 - 2x_2) \\ -4(x_1 - 2x_2) + 16x_2^3 \end{bmatrix}, \quad y$$

$$\nabla^2 f(x_1, x_2) = \begin{bmatrix} 50x_1^4 + 48x_1^2 + 2 & -4 \\ -4 & 48x_2 + 8 \end{bmatrix}.$$

Donde aparece el primer CORTE AQUÍ, se elimina esa parte de código para dar el punto inicial, por lo que escribiremos

x=[4.7;-0.9]; ver la figura 7.19.

Hay que mencionar que el punto inicial depende del tipo de función y de la dimensión de la misma. Para el ejemplo de Rosenbrock extendida el número de variables viene dada por n en el codigo de rosenbext.m. Se suguiere dar una buena revisada a dicho archivo para familiarizarce con la creación de funciones a minimizar.

5. Donde aparece el segundo CORTE AQUÍ, se debe eliminar esa parte de código y de…nir la nueva función a minimizar. Continuando con el ejemplo, se debe escribir (tal cual se muestra):

```
sal=(5/3)*x(1)^6+4*x(1)^4+(x(1)-2*x(2))^2+4*x(2)^4;
```

ver …gura 7.20.

De manera análoga en los últimos dos CORTE AQUÍ, se elimina la parte de código y se introduce el gradiente y la hessiana en dichas partes. En el caso del ejemplo que estamos de…niendo, se introduce[2]:

```
y=[10*x(1)^5+16*x(1)^3+2*(x(1)-2*x(2));-4*(x(1)-2*x(2))+16*x(2)^3];
```

para el gradiente y

```
H=[50*x(1)^4+48*x(1)^2+2,-4;...
-4,8+48*x(2)^2];
```

en el caso de la hessiana.

6. Ya que se ha escrito correctamente el archivo *.m de la función a minimizar, lo siguiente es guardar.

   Lo primero que hay que hacer es cambiar rosenbext, nombre del archivo localizado en la línea uno del código, por el que se desee.Y para esto se hacen dos recomendaciones:

   (a) Al guardar, siempre hay que dar el mismo nombre de la función, al archivo ¤.m tal y como lo muestran la …gura 7.21.

   (b) Los archivos guardados deberán estar en la misma carpeta que optimin.m ver …gura 7.21.

7. El archivo almacenado es como el que se muestra en la …gura 7.22.

[2] Se debe escribir tal cual se muestra

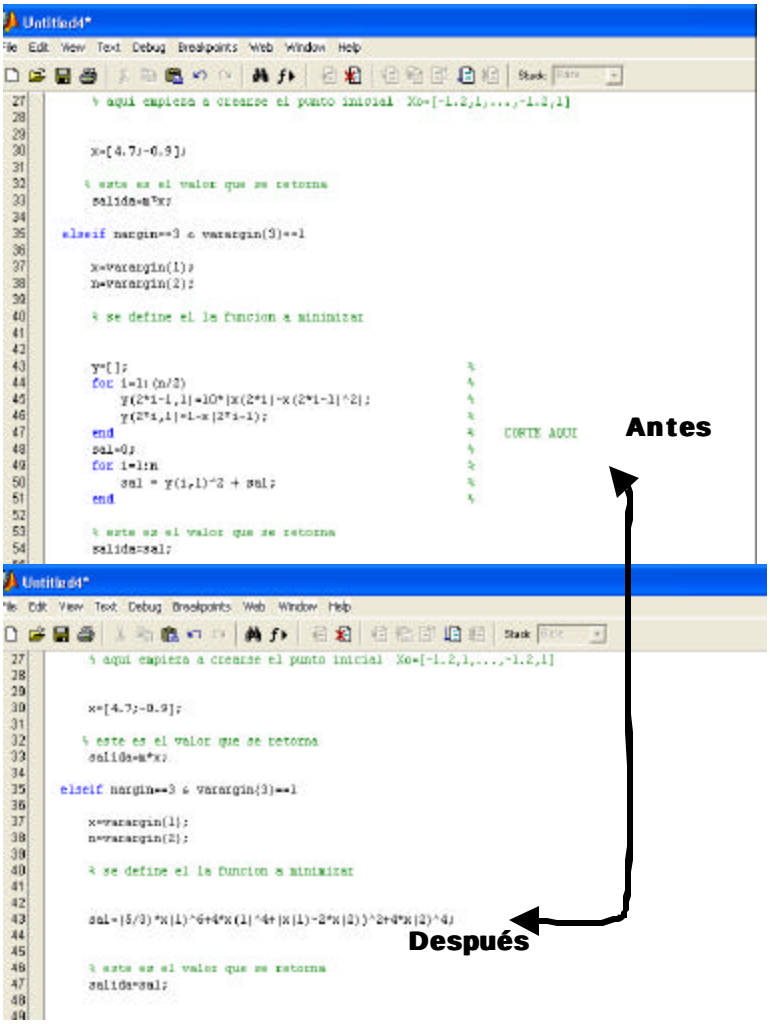


Figura 7.20: Ejemplo de la forma en que se introduce la función a minimizar.

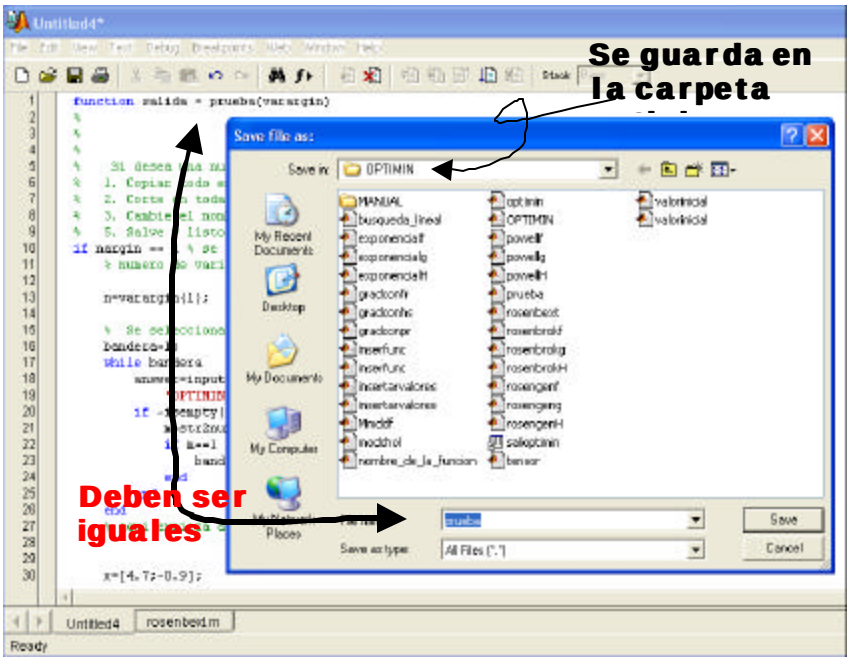


Figura 7.21: Muestra como se guarda una nueva función a minimizar escrita en un archivo ¤.m.

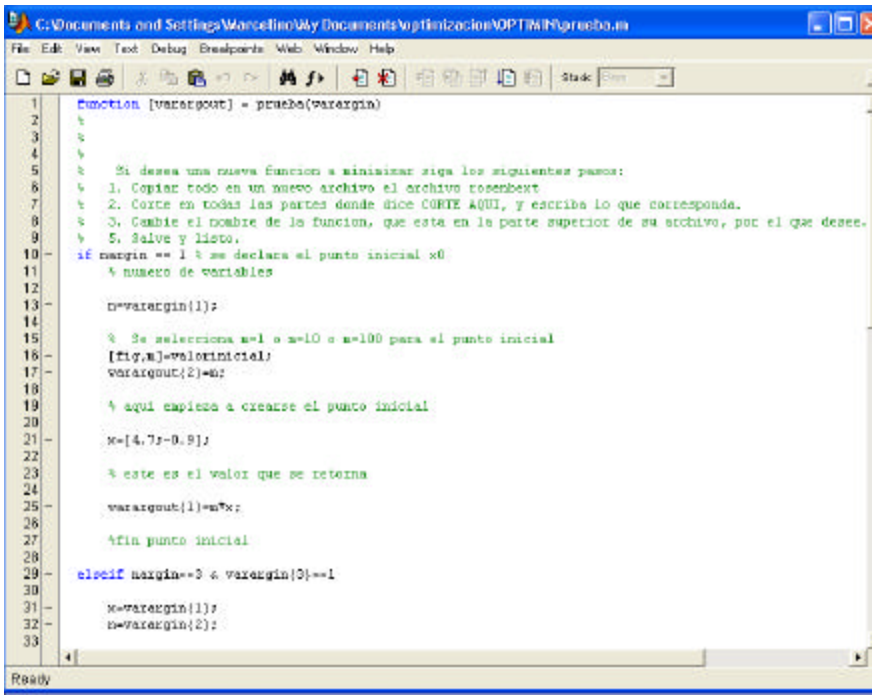
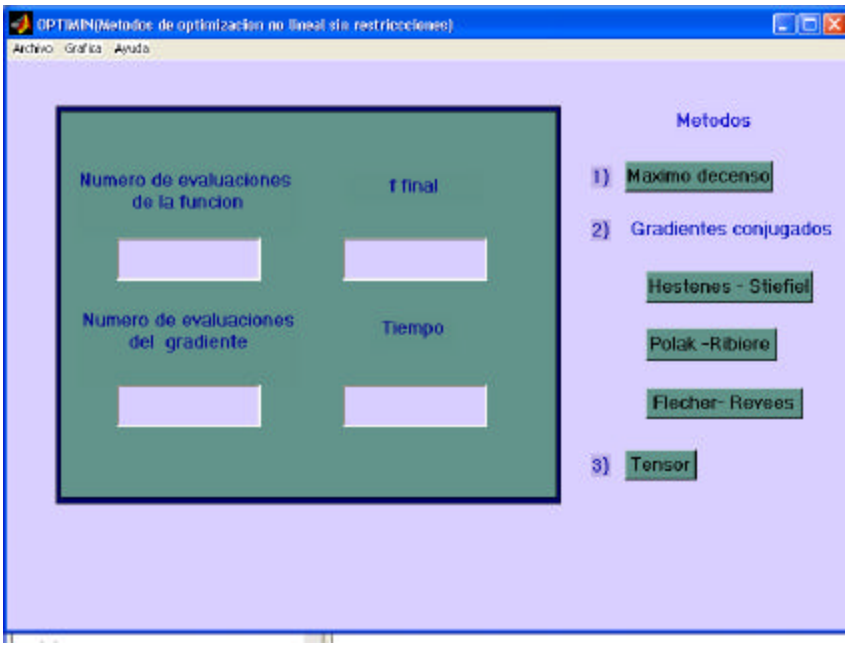


Figura 7.22: Así queda el archivo de la función a minimizar .

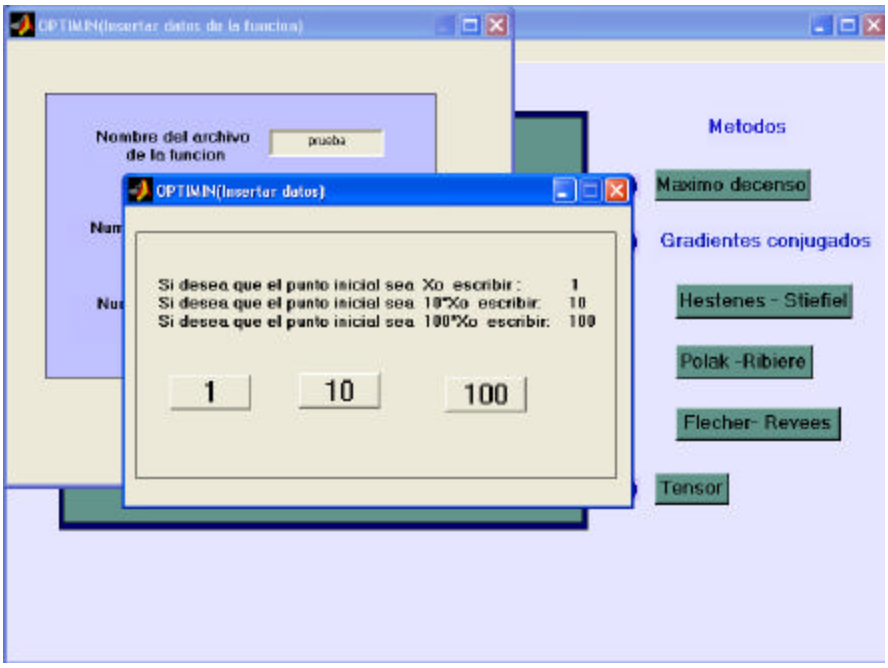
7.3 Ejemplo

A continuación se muestra un ejemplo de como utilizar OPTIMIN con la función prueba dada en la sección anterior

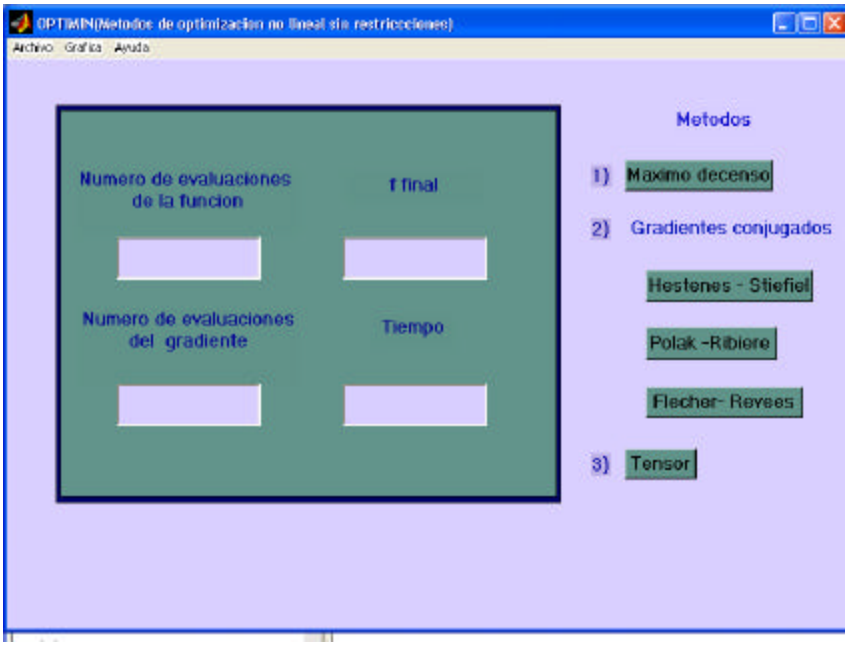
1. Se teclea optimin en el prompt del Command Window de MATLAB, y aparece una ventana como la siguiente:



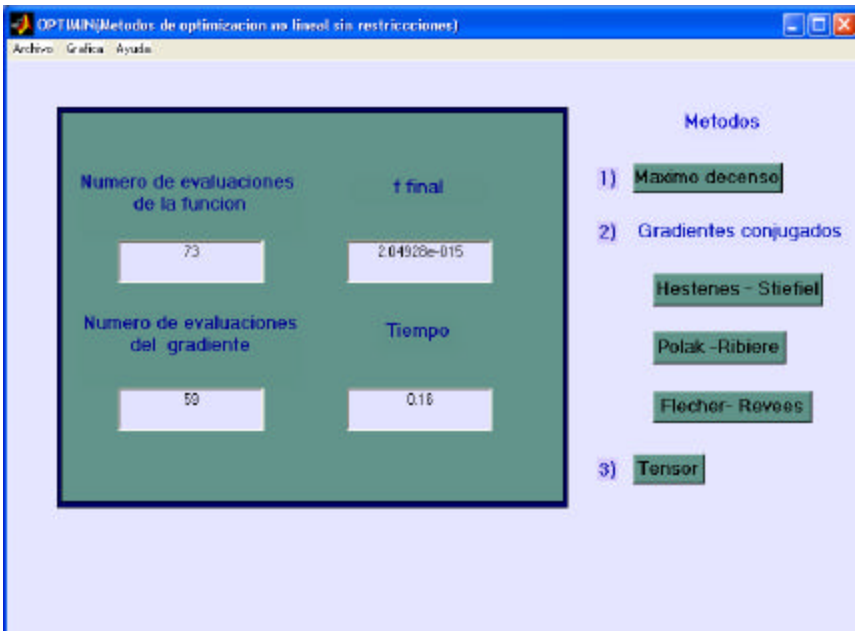
2. Después en el Menú Archivo se selecciona Introducir datos de la función. En la nueva ventana que aparece, escribir prueba en la ventanilla junto a Nombre del archivo de la funcion. Después junto a Numero de variables escribir 2 y aparecerá una ventana como la siguiente, en la cual seleccionar 1.



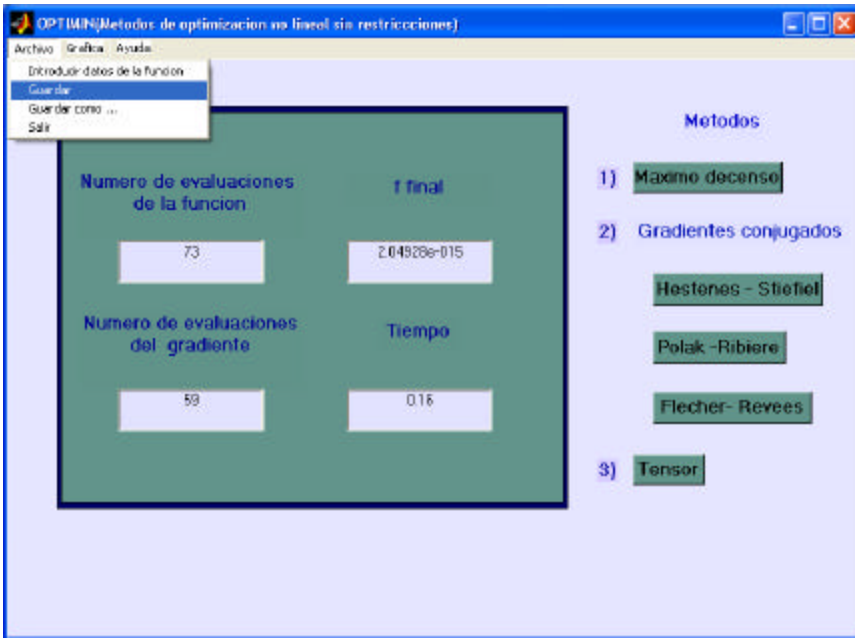
3. Después escribir 200 (se puede elegir cualquier número entero positivo) en la ventanilla junto a Numero maximo de iteraciones y hacer click en Aplicar. Si todos los datos son correctos apararecerá la ventana principal de OPTIMIN.



4. Elegir algún método y hacer click en él, por ejemplo Tensor, luego en la ventana principal de OPTIMIN aparece lo siguiente:

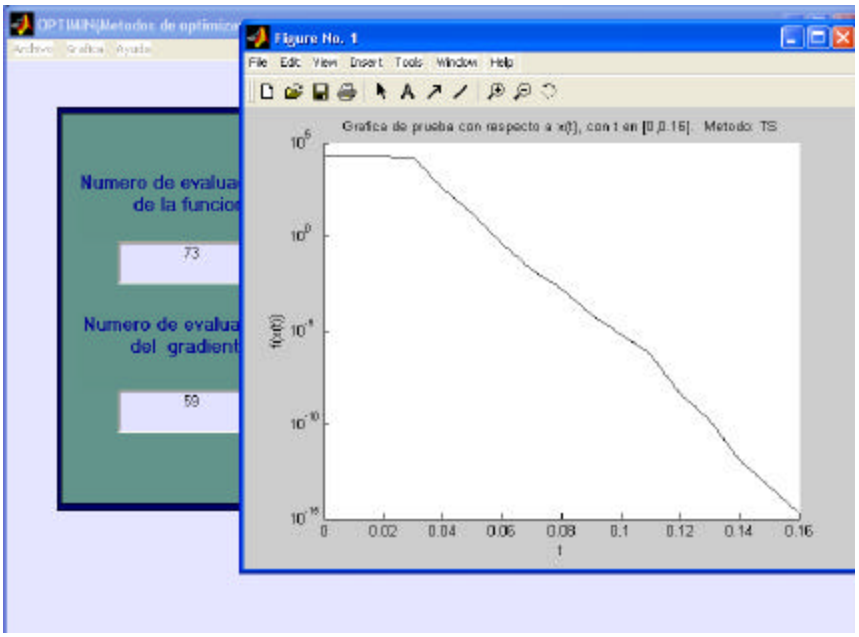


5. Seleccionar Guardar en el menú archivo para almacenar la información.

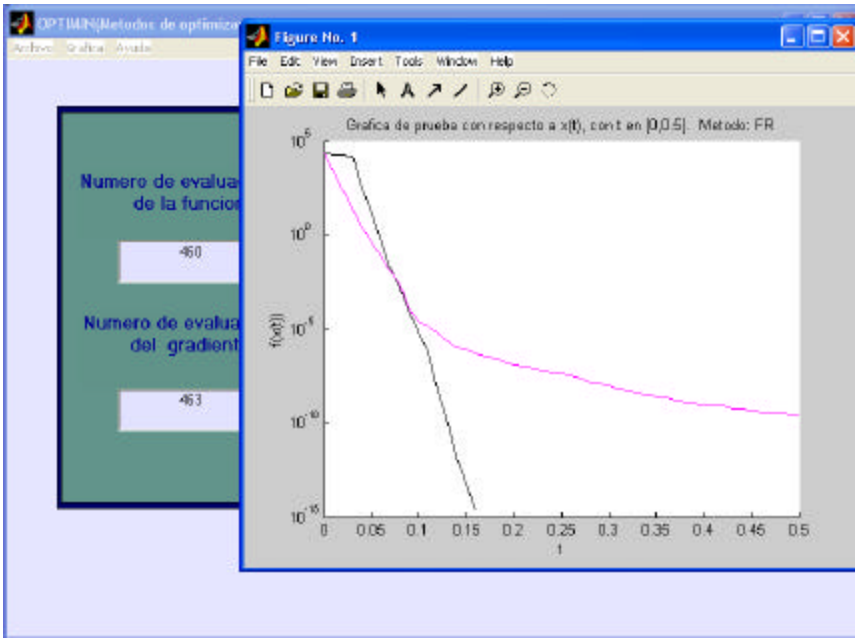


6. Para gra...car seleccionar Ver gra...ca del método en el menú Gra...car,

para lo cual aparece la ventana:



7. Se procede a seleccionar otro método por ejemplo Flecher-Reeves, pero sin cerrar la …gura No. 1, y después gra…car el método para obtener:



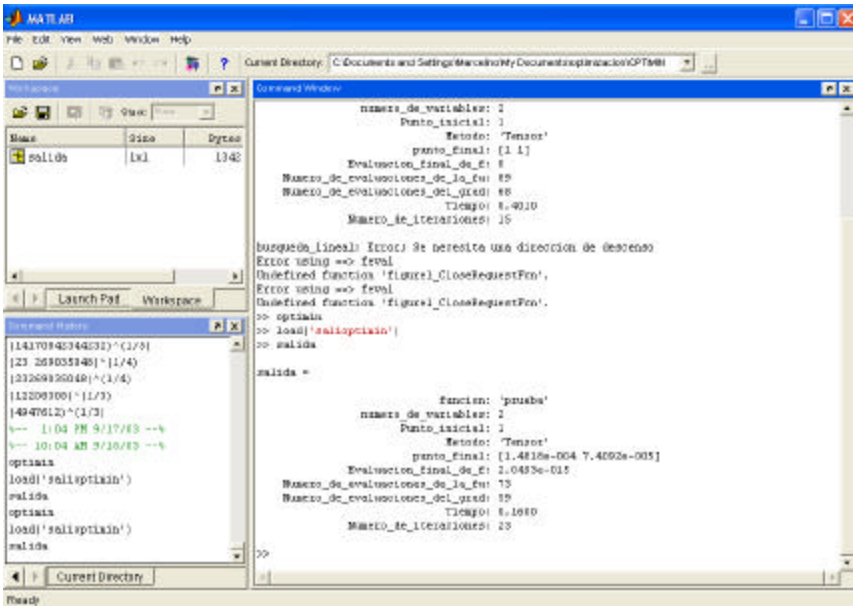
8. Seleccionamos Salir en el menú Archivo para cerrar OPTIMIN.

9. En el prompt de MATLAB se escribe

   >> load('salioptimin') Ã›

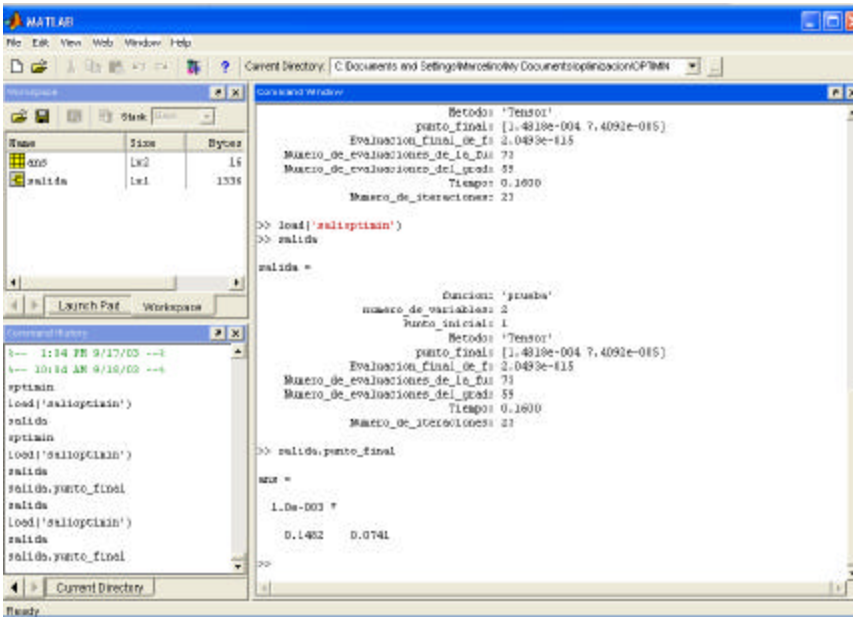
   Aquí salioptimin es el archivo con extensión mat, donde se guardó el resultado del método de tensor. Después escribir en el prompt

   >>salida Ã› como resultado se obtiene:



10. Si se desea obtener cualquier dato que tiene almacenado salida solo escribimos salida.dato, donde dato es cualquiera de los nombres de la izquierda en la …gura anterior. Por ejemplo si se desea obtener el punto …nal del método, escribir

    >> salida.punto_…nal Ã› (ver la …gura siguiente).



El punto $x^* = (0,0)$ es el minimizador global de esta función que se esta minimizando (ver (7.2) página 54). Esto porque $f(x_1, x_2)$ es convexa y x^* satisface las condiciones del teorema 1.9 (csso), página 3, esto es, $\nabla f(0,0) = 0$ y $\nabla^2 f(0,0) = \begin{pmatrix} 2 & -4 \\ -4 & 8 \end{pmatrix} > 0$. Todo esto confirma lo obtenido por OPTIMIN.

11. Fin.

Bibliografía

[1] R. B. Schnabel and T. Chow, “Tensor Methods for Unconstrained Optimization using second derivates”, SIAM Journal on Optimization, 1991, pp.293-315.

[2] Bouaricha Ali, “Tensor Methods for Large, Sparce Unconstrained Optimization”, SIAM Journal on Optimization, 1997, pp.732-756.

[3] Nocedal Jorge, “Theory of Algorithms for Unconstrained Optimization”, Acta Numérica(1991), pp.1-37.

[4] J. E. Dennis and R. B. Schnabel, “Numerical Methods for Unconstrained Optimization and Nolinear Equations”, SIAM, Usa, 1996.

[5] Edwing K. P. Chong and Stanislaw H. Żak, "An introduction to optimization", John Wiley & Sons, 1996.

[6] Nocedal Jorge and Stephen J. Wright, "Numerical Optimization", Springer-verlang, USA, 1999.

[7] I. Bongartz, A.R. Conn, Nick Gould, and Ph.L. Toint, "Constrained and Unconstrained Testing Environment", PUBLICATIONS DU DÉPARTEMENT DE MATHÉMATIQUE, 1993.

Índice Analítico