UNIVERSIDAD NACIONAL DE LA PLATA ★ FACULTAD DE CIENCIAS EXACTAS ★ CENTRO DE MATEMÁTICA · BIBLIOTECA

Estudio de alternativas para determinar una dirección de descenso en el método de Newton Truncado cuando existe curvatura negativa *

María Laura Schuverdt †

Mayo 2001

*Trabajo de tesis de grado dirigido por la Profesora Nélida Echebest.

†Departamento de Matemática, Facultad de Ciencias Exactas, UNLP, Argentina.

Iniciación 23

1 Introducción

Los métodos de búsqueda lineal al igual que el método de regiones de confianza son utilizados para optimizar problemas irrestrictos de la forma

$$\min_{x \in \Re^n} f(x)$$

donde f es una función a valores reales con $\nabla f(x)$ y $\nabla^2 f(x)$ continuos, a la que llamaremos función objetivo.

Los métodos de búsqueda lineal en cada iterado x_k calculan una dirección p_k y luego deciden cuanto se desplazan a lo largo de esa dirección para obtener el siguiente iterado $x_{k+1} = x_k + \alpha_k\ p_k$ donde α_k es un escalar positivo llamado longitud de paso.

La mayoría de los algoritmos requieren que p_k sea una dirección de descenso, ya que esto garantiza que la función objetivo decrece en esa dirección al menos en un intervalo.

Una vez encontrada la dirección p_k se elige el paso α_k de manera tal que se dé una notoria reducción de f, tratando de no realizar demasiados cálculos en dicha elección. Una posible elección sería buscar el mínimo global de la función unidimensional $\Phi(\alpha) = f(x_k + \alpha\ p_k)$ con $\alpha > 0$, pero en general requiere mucho costo computacional identificar este valor. Estrategias más prácticas realizan búsquedas inexactas para encontrar una longitud de paso con menores costos.

Los algoritmos standard de búsqueda lineal tratan de generar una sucesión de valores candidatos para α, parando cuando uno de estos valores satisface ciertas condiciones. Las condiciones que suelen usarse para encontrar longitudes de paso son las conocidas como condiciones de Wolfe que se enuncian a continuación:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1\ \alpha_k\ \nabla f(x_k)^t\ p_k,\ \ con\ c_1 \in (0, 1/2) \tag{1.1}$$

$$\nabla f(x_k + \alpha_k p_k)^t p_k \geq c_2\ \nabla f(x_k)^t\ p_k,\ \ con\ c_2 \in (c_1, 1) \tag{1.2}$$

La desigualdad (1.1) es una condición de suficiente decrecimiento de la función objetivo, y nos dice que la reducción de f debe ser proporcional a la longitud del paso α_k y a la derivada direccional $\nabla f(x_k)^t p_k$. Esta desigualdad es conocida como condición de Armijo.

La desigualdad (1.2), llamada condición de curvatura, asegura suficiente alejamiento del punto x_k y nos dice que la pendiente en el nuevo punto tiene que ser significativamente más grande que la pendiente inicial.

La condición (1.2) puede ser reemplazada por la siguiente desigualdad

$$|\nabla f(x_k + \alpha_k p_k)^t p_k| \leq c_2 \, |\nabla f(x_k)^t \, p_k|, \; \textit{con } c_2 \in (c_1, 1) \tag{1.3}$$

que, junto a la condición (1.1) son conocidas como condiciones fuertes de Wolfe.

El siguiente Lema garantiza que para ciertas funciones se verifican las condiciones de Wolfe. Una demostración del mismo puede ser hallada en Nocedal, Wright [3].

Lema 1.1 *Sea f una función a valores reales continua con $\nabla f(x)$ continuo. Sea p_k una dirección de descenso en x_k y supongamos que f está acotada inferiormente en la recta $\{x_k + \alpha p_k \; ; \; \alpha > 0\}$. Si $0 < c_1 < c_2 < 1$ entonces existe un intervalo de longitudes de paso que satisfacen las condiciones de Wolfe.*

Un Lema análogo vale para las condiciones fuertes de Wolfe.

A continuación se describe un esquema general de un algoritmo iterativo de minimización con búsqueda lineal.

Algoritmo:

Dados un punto inicial x_0, $\epsilon > 0$, $0 < c_1 < c_2 < 1$, $k = 0$

Paso Iterativo: Dados x_k, $g_k = \nabla f(x_k)$,

- *Criterio de finalización.*
- *Obtiene de alguna manera una dirección p_k de descenso.*

- *Calcula el paso α_k que satisface las condiciones (1.1) y (1.2) o (1.1) y (1.3).*
- *Define $x_{k+1} = x_k + \alpha_k p_k$.*
- $k : k + 1$ $\diamond$

Fin

El criterio de finalización que usaremos será $\|g_k\| < \epsilon$ y el algoritmo GSRCH de Powell será el empleado para calcular la longitud del paso α_k que satisfaga las condiciones (1.1) y (1.3).

En el esquema general queda por definir una dirección de búsqueda.

Si se define la dirección p_k mediante la fórmula $p_k = -H_k^{-1} g_k$, siendo $H_k = \nabla^2 f(x_k)$, el esquema anterior corresponde al Método de Newton, y p_k se denomina dirección de Newton. Para asegurar que tal dirección de Newton está bien definida y que es de descenso se requiere que la función objetivo sea estrictamente convexa.

En la siguiente sección se describirá un método que puede ser utilizado para obtener una dirección de descenso cercana a la de Newton.

2 Método de Newton Truncado

Este método se obtiene usando en el esquema previo como dirección de descenso una aproximación a la dirección de Newton. Las propiedades de convergencia local del método de Newton pueden mantenerse para el método de Newton Truncado si la dirección aproximada satisface particulares condiciones [1].

Recordemos que el método de Newton en cada iterado x_k aproxima la función $f(x_k+p)$ por el modelo cuadrático

$$q_k(p) = f(x_k) + p^t g_k + \frac{1}{2}\, p^t H_k p$$

siendo $g_k = \nabla f(x_k)$ y $H_k = \nabla^2 f(x_k)$. Cuando la función objetivo es estrictamente convexa la dirección de Newton se obtiene al minimizar en forma exacta el problema

$$\min_{p \in \Re^n} q_k(p)$$

es decir, es la solución del siguiente sistema de ecuaciones

$$H_k p = -g_k. \tag{2.4}$$

Para resolver aproximadamente este sistema puede utilizarse el método de Gradientes Conjugados mientras no se encuentren direcciones de curvatura negativa o cero, truncando el proceso iterativo cuando el residuo alcance determinada magnitud o cumpla alguna propiedad que mida la aproximación requerida a la solución.

Recordemos que el método de Gradientes Conjugados, en cada iteración, minimiza q_k en el espacio de Krylov generado por las direcciones definidas hasta ese momento. Por lo tanto es conveniente utilizarlo para calcular, con bajo costo computacional, una dirección de descenso cercana a la de Newton.

En general, los criterios o reglas de terminación de los métodos iterativos más utilizados, para resolver en forma aproximada el sistema (2.4), están basadas en el residuo $r_j = H_k p_j + g_k$, donde p_j es la solución iterativa hallada por el método. Un criterio, definido por Dembo y Steihaug en [1], para detener el proceso iterativo es

$$\|r_j\| \leq \eta_k \|g_k\|$$

donde η_k puede ser una constante fija en $(0, 1)$, o una sucesión $\{\eta_k\}$ de números reales en $(0, 1)$ llamada sucesión de forzamiento.

En lo que resta de esta sección se suprimirán los índices k para simplificar la notación, recordando que en cada iteración k del método de Newton Truncado se resuelve de manera aproximada el sistema (2.4) usando el método de Gradientes Conjugados.

A continuación se describe el algoritmo básico del método de Gradientes Conjugados Truncado (Dembo, Steihaug [1]).

Algoritmo GCT:

Inicio: Dados $\epsilon_c > 0$, $\eta > 0$

Sean $p_0 = 0$, $r_0 = g$, $d_0 = -g$

$\delta_0 = \|r_0\|^2$

$j = 0$

Paso 1: Si $d_j^t H d_j \leq \epsilon_c \delta_j$

Sale con $p = d_0$ si $j = 0$, y con $p = p_j$ en otro caso

Sino,

Paso 2: Calcula $\alpha_j = \|r_j\|^2 / (d_j^t H d_j)$

Define $p_{j+1} = p_j + \alpha_j d_j$

Calcula $r_{j+1} = r_j + \alpha_j H d_j$

Si $\|r_{j+1}\| / \|r_0\| \leq \eta$ sale con $p = p_{j+1}$

Sino,

Paso 3: Calcula $\beta_j = \|r_{j+1}\|^2 / \|r_j\|^2$

Define $d_{j+1} = -r_{j+1} + \beta_j d_j$

Calcula $\delta_{j+1} = \|d_{j+1}\|^2 = \|r_{j+1}\|^2 + \beta_j^2 \delta_j$

$j : j + 1$ y retorna al Paso 1.

Las distintas maneras en las que puede finalizar el algoritmo GCT son:

Caso 1: Cuando en una iteración del algoritmo GCT se cumple la condición $d_j^t H d_j \leq \epsilon_c \delta_j$ el proceso se detiene. Si $j = 0$, sale con $p = -g$, la dirección de máximo descenso. Si $j > 0$, sale con $p = p_j$. Este es el criterio de terminación descripto en el Paso 1 del algoritmo anterior.

Caso 2: Cuando en el algoritmo GCT se satisface la desigualdad $\|r_{j+1}\| / \|r_0\| \leq \eta$, es decir, se cumple la condición de suficiente reducción del residuo, el procedimiento finaliza con $p = p_{j+1}$. Este es el criterio de terminación descripto en el Paso 2.

Proposición 2.1 *Dado un modelo cuadrático q no convexo. Si el algoritmo GCT obtiene una dirección d_j de curvatura negativa, entonces d_j es una dirección de descenso de la función objetivo en el iterado actual x.*

Demostración

Si $j = 0$, el resultado es obvio por ser $d_0 = -g$, siendo g el gradiente en el iterado actual x del algoritmo principal.

Si $j > 0$, como el residuo en p_j es $r_j = g + Hp_j$, usando propiedades del método de Gradientes Conjugados [3] se tiene

$$d_j^t g = d_j^t r_j - d_j^t H p_j = d_j^t r_j$$

considerando que p_j es H-conjugada con d_j. Luego, considerando la definición de d_j, y la propiedad de ortogonalidad de r_j respecto de las direcciones $d_0, \ldots, d_{j-1}$, si $j \geq 1$, se obtiene

$$d_j^t g = (-r_j + \beta_{j-1} d_{j-1})^t r_j = -\|r_j\|^2 + \beta_{j-1} d_{j-1}^t r_j = -\|r_j\|^2 < 0.$$

$\diamond$

Denominamos algoritmo $NTGC$ al método de Newton Truncado que usa el algoritmo

GCT para calcular, en cada iteración k, la dirección de descenso aproximada a la dirección de Newton.

En la siguiente sección se enunciarán los resultados que garantizan que el algoritmo $NTGC$ es globalmente convergente.

3 Convergencia global del método NTGC

Veremos ahora los resultados necesarios para probar que el método $NTGC$ es globalmente convergente, es decir, para cualquier punto inicial x_0, la sucesión generada satisface

$$\|g_k\| \to 0 \text{ cuando } k \to \infty.$$

Los resultados del Lema siguiente son fundamentales para obtener la propiedad de convergencia global del método.

Lema 3.1 *Si p es la dirección definida por GCT, entonces existen constantes positivas μ_0 y μ_1, que dependen de $\|H\|$ y de ϵ_c respectivamente, tales que*

$$i) g^t p \leq -\mu_0 \|g\|^2 \tag{3.5}$$

$$ii) \|p\| \leq \mu_1 \|g\| \tag{3.6}$$

De la desigualdad (3.5) se desprende que todas las direcciones halladas por el algoritmo GCT son de descenso.

Una demostración de este Lema puede encontrarse en [1], así como los valores de las constantes que permiten obtener i) y ii) son $\mu_0 = \min\{1, \frac{1}{\|H\|}\}$ y $\mu_1 = \max\{1, \frac{n}{\epsilon_c}\}$.

Teorema 3.1 *Asumiendo que f es acotada inferiormente en $\Re^n$, continua con $\nabla f(x)$ continuo en cualquier conjunto abierto B que contiene a $A = \{x : f(x) \leq f(x_0)\}$, donde x_0 es el punto inicial de un algoritmo de búsqueda lineal. Si además ∇f es Lipschitz continuo sobre B, es decir,*

$$\exists L > 0 : \|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \ \forall x, y \in B$$

Entonces

$$\sum_{k \geq 0} \cos^2(\theta_k)\|g_k\|^2 < \infty \tag{3.7}$$

siendo $\cos(\theta_k) = -g_k^t p_k / (\|g_k\| \ \|p_k\|)$, *y* p_k *la dirección de descenso del método en la iteración* k.

Una demostración de este Teorema puede hallarse en [3].

La desigualdad (3.7), llamada condición de Zoutendijk, implica que $\cos^2(\theta_k)\|g_k\|^2 \to 0$ cuando $k \to \infty$. Este límite puede ser utilizado para probar la convergencia global de los algoritmos de búsqueda lineal y en particular la del Método de Newton Truncado.

Si el coseno de θ_k, ángulo formado por las direcciones $-g_k$ y p_k, se mantiene mayor o igual que una constante estrictamente positiva e independiente de k, se tiene que $\|g_k\| \to 0$ cuando $k \to \infty$, es decir, el algoritmo es globalmente convergente.

Para demostrar la convergencia global del método *NTGC* se asumen las siguientes hipótesis generales:

i) f es una función a valores reales con $\nabla f(x)$ Lipschitz continuo y $\nabla^2 f(x)$ continuo;

ii) f es acotada inferiormente en $\Re^n$ y el conjunto de nivel $A = \{x : f(x) \leq f(x_0)\}$ es acotado;

iii) Las matrices H_k están uniformemente acotadas en norma, es decir $\|H_k\| \leq M \ \forall k \geq 0$.

Teorema 3.2 *Supongamos que se verifican las condiciones i), ii) y iii). Entonces la sucesión de iterados x_k generada por el algoritmo NTGC cumple que $\|g_k\| \to 0$ cuando $k \to \infty$, es decir, el algoritmo es globalmente convergente.*

Demostración

Sea θ_k el ángulo formado por las direcciones $-g_k$ y p_k siendo p_k la dirección de descenso obtenida en la iteración k por el algoritmo *GCT*.

Usando las desigualdades (3.5) y (3.6) obtenemos que

$$\cos(\theta_k) = \frac{-g_k^t p_k}{\|g_k\| \ \|p_k\|} \geq \mu_0 \frac{\|g_k\|}{\|p_k\|} \geq \frac{\mu_0}{\mu_1}.$$

Teniendo en cuenta la condición iii) y que $\mu_0 = \min\{1, \frac{1}{\|H_k\|}\}$ se obtiene la siguiente cota que es independiente de k

$$\cos(\theta_k) \geq \frac{1}{\mu_1} \min\{1, \frac{1}{M}\}.$$

Por lo tanto, como consecuencia del Teorema 3.1 se tiene que $\|g_k\| \to 0$ cuando $k \to \infty$.

◇

Además de los resultados de convergencia global del método *NTGC* se tienen los siguientes resultados cuyas demostraciones pueden ser encontradas en [1].

Teorema 3.3 *Si $x_k \to x^*$ y $H(x^*)$ es definida positiva entonces, existe un índice k_0 tal que $\forall k \geq k_0$ se satisface el criterio de finalización del Paso 2 del algoritmo GCT y $\alpha_k = 1$ es un valor que satisface las condiciones requeridas (1.1) y (1.2) en la búsqueda lineal en NTGC.*

Teorema 3.4 *Si $x_k \to x^*$ donde $H(x^*)$ es definida positiva y si H es Lipschitz continuo en x^* entonces, la convergencia es superlineal si y sólo si $\|r_k\|/\|g_k\| \to 0$ cuando $k \to \infty$.*

Además, la convergencia es de orden $1 + t$ si y sólo si lim sup $\|r_k\|/\|g_k\|^{1+t} < \infty$ cuando $k \to \infty$.

Recordemos que cuando el algoritmo GCT detecta una dirección de curvatura negativa la desecha y se detiene aceptando la dirección $p = \sum \alpha_i d_i$. *la cual es una combinación positiva de todas las direcciones conjugadas de curvatura positiva previamente halladas.*

En la siguiente sección se propondrá una modificación en el algoritmo *GCT*, en el caso en el que éste encuentra una dirección de curvatura negativa. Veremos que tal modificación no alterará la global convergencia del método de Newton Truncado.

En los métodos de regiones de confianza [3], cuando el algoritmo de Gradientes Conjugados es usado para minimizar el modelo cuadrático y detecta una dirección de curvatura negativa, el proceso se detiene y un punto sobre tal dirección es aceptado. En particular, el método de Steihaug [3] considera en tal situación el punto que se encuentra en la frontera de la región.

El uso de la dirección de curvatura negativa, afirman Lucidi, Rochetich y Roma en [2], permite moverse más rápidamente de la zona no convexa a una zona donde la función objetivo sea convexa y donde las propiedades del método de Newton sean aplicables.

Esto motiva a investigar nuevos algoritmos que, para definir las direcciones de búsqueda, usen la información aportada por las direcciones de curvatura negativa halladas.

En el apéndice se describe como Lucidi, Rochetich y Roma [2] utilizan las direcciones de curvatura negativa en la resolución de problemas de minimización irrestricta.

4 Modificación propuesta en la salida del algoritmo GCT

Vamos a proponer para el caso cuando el algoritmo *GCT* encuentra una dirección de curvatura negativa d_j, con $j > 0$, otra forma de definir una dirección de descenso.

En lugar de desechar la dirección encontrada y salir con el iterado previo p_j, usaremos la información de la dirección de curvatura negativa para hallar otra dirección p de

descenso. Tal modificación preservará que el método de Newton Truncado sea globalmente convergente.

Proponemos buscar

$$p = p_j + \lambda d_j$$

siendo $\lambda = b\ \alpha$ con b un escalar positivo y α tal que la dirección $d = p_j + \alpha d_j$ sea de curvatura nula (cambio de curvatura de positiva a negativa).

Usando que p_j es combinación positiva de las direcciones $d_0, d_1, ..., d_{j-1}$, que son direcciones H-conjugadas y de curvatura positiva, y d_j también es H-conjugada respecto de las previas, se obtienen las siguientes igualdades

$$0 = d^t H d = (p_j + \alpha d_j)^t H (p_j + \alpha d_j) =$$

$$= p_j^t H p_j + 2\alpha\ d_j^t H p_j + \alpha^2\ d_j^t H d_j = p_j^t H p_j + \alpha^2\ d_j^t H d_j.$$

Resolviendo la anterior ecuación cuadrática en α encontramos

$$\alpha = \sqrt{-\frac{p_j^t H p_j}{d_j^t H d_j}}.$$

Observamos que α está bien definida pues el radicando es positivo, considerando que p_j es de curvatura positiva, $d_j^t H d_j < 0$ y $p_j^t H p_j > 0$.

Si se elige $b \in (0, 1)$ la dirección propuesta será una dirección de curvatura positiva y si b fuese mayor a 1 será una dirección de curvatura negativa.

Nuestro objetivo es estudiar el comportamiento de la dirección combinada con distintos valores de b. En particular, cuando b es casi cero la nueva dirección es muy cercana a la dirección original del algoritmo GCT. Si $b = 1/2$, es una dirección de curvatura positiva intermedia, mientras que si $b > 1$ es de curvatura negativa. Se restringirá el dominio de elección de b al intervalo $(0, 2)$.

Se obtiene entonces la siguiente modificación del algoritmo GCT.

Algoritmo GCT modificado (GCTM):

Inicio: Dados $\epsilon_c > 0$, $\eta > 0$, $0 < b < 2$

Sean $p_0 = 0$, $r_0 = g$, $d_0 = -g$

$\delta_0 = \|r_0\|^2$

$j = 0$

Paso 1: Si $|d_j^t H d_j| \leq \epsilon_c \delta_j$

Sale con $p = d_0$ si $j = 0$, y con $p = p_j$ en otro caso

Si $d_j^t H d_j < -\epsilon_c \delta_j$

Si $j = 0$ sale con $p = d_0$

Si $j > 0$ calcula $\alpha = \sqrt{-\frac{p_j^t H p_j}{d_j^t H d_j}}$

Sale con $p = p_j + b\alpha d_j$

Sino,

Paso 2: Calcula $\alpha_j = \|r_j\|^2 / (d_j^t H d_j)$

Define $p_{j+1} = p_j + \alpha_j d_j$

Calcula $r_{j+1} = r_j + \alpha_j H d_j$

Si $\|r_{j+1}\| / \|r_0\| \leq \eta$ sale con $p = p_{j+1}$

Sino,

Paso 3: Calcula $\beta_j = \|r_{j+1}\|^2 / \|r_j\|^2$

Define $d_{j+1} = -r_{j+1} + \beta_j d_j$

Calcula $\delta_{j+1} = \|d_{j+1}\|^2 = \|r_{j+1}\|^2 + \beta_j^2 \delta_j$

$j : j + 1$ y retorna al Paso 1.

La modificación que se ha introducido en el algoritmo anterior está descripta en el Paso 1. Cuando se obtiene que $d_j^t H d_j < -\epsilon_c \delta_j$ con $j > 0$ se propone la nueva dirección $p = p_j + b\alpha d_j$ con $\alpha = \sqrt{-\frac{p_j^t H p_j}{d_j^t H d_j}}$ y $0 < b < 2$ fijo.

Demostraremos a continuación que la nueva dirección p verifica las condiciones que garantizan la convergencia global del método Newton Truncado usando *GCTM*.

Lema 4.1 *Si p es la dirección definida por GCTM, entonces existen constantes positivas γ_0 y γ_1, que dependen de $\|H\|$ y de ϵ_c, tales que*

$$i) g^t p \leq -\gamma_0 \|g\|^2 \tag{4.8}$$

$$ii) \|p\| \leq \gamma_1 \|g\| \tag{4.9}$$

Demostración

Si $|d_j^t H d_j| \leq \epsilon_c \delta_j$, la salida p coincide con la salida original del algoritmo *GCT* y por lo tanto, por el Lema 3.1 valen las desigualdades (4.8) y (4.9) con $\gamma_0 = \mu_0 = \min\{1, \frac{1}{\|H\|}\}$ y $\gamma_1 = \mu_1 = \max\{1, \frac{n}{\epsilon_c}\}$.

Si $d_j^t H d_j < -\epsilon_c \delta_j$ y $j = 0$ se procede como en el caso anterior.

Si $d_j^t H d_j < -\epsilon_c \delta_j$ y $j > 0$ la salida del algoritmo modificado es $p = p_j + b\alpha d_j$ siendo α el escalar propuesto en el Paso 1 del algoritmo *GCTM*. Usando que d_j es dirección de descenso, y que p_j, por coincidir con la salida original del algoritmo *GCT* verifica la desigualdad (3.5), se obtiene que

$$g^t p = g^t p_j + b\ \alpha\ g^t d_j \leq g^t p_j \leq -\ \mu_0 \|g\|^2.$$

Por lo tanto se verifica la desigualdad (4.8) con $\gamma_0 = \min\{1, \frac{1}{\|H\|}\}$.

Veamos entonces que se verifica la desigualdad (4.9).

$$\|p\| = \|p_j + b\ \alpha d_j\| \leq \|p_j\| + b\ \alpha\|d_j\|.$$

La dirección p_j coincide con la salida del algoritmo GCT y por lo tanto verifica la desigualdad (3.6), es decir $\|p_j\| \leq \mu_1\|g\|$. Por lo tanto

$$\|p\| \leq \mu_1\|g\| + b\ \alpha\|d_j\|.$$

Como $d_j^t H d_j < -\epsilon_c\delta_j$, $0 < p_j^t H p_j \leq \|p_j\|^2\|H\|$ y $\|p_j\| \leq \mu_1\|g\|$ se tiene que

$$\alpha < \sqrt{\frac{p_j^t H p_j}{\epsilon_c\delta_j}} \leq \frac{\|p_j\|\sqrt{\|H\|}}{\sqrt{\epsilon_c\delta_j}} \leq \frac{\mu_1\|g\|\sqrt{\|H\|}}{\sqrt{\epsilon_c}\sqrt{\delta_j}}.$$

Se obtiene entonces

$$\|p\| \leq \mu_1\|g\| + b\frac{\mu_1\|g\|\sqrt{\|H\|}\|d_j\|}{\sqrt{\epsilon_c}\sqrt{\delta_j}} = \mu_1(1 + b\frac{\sqrt{\|H\|}}{\sqrt{\epsilon_c}})\|g\|.$$

En la última igualdad se ha usado que $\delta_j = \|d_j\|^2$.

Como se considera $0 < b < 2$ se obtiene que

$$\|p\| \leq \mu_1(1 + 2\frac{\sqrt{\|H\|}}{\sqrt{\epsilon_c}})\|g\|.$$

Por lo tanto se verifica la desigualdad (4.9) para la nueva dirección p.

Se obtiene entonces, que tomando $\gamma_1 = \mu_1(1 + 2\frac{\sqrt{\|H\|}}{\sqrt{\epsilon_c}})$, la salida del algoritmo $GCTM$ verifica la desigualdad (4.9). $\diamond$

De la desigualdad (4.8) se observa que la nueva dirección generada por el algoritmo $GCTM$ es también dirección de descenso.

Teorema 4.1 *Supongamos que se verifican las condiciones i), ii) y iii) del Teorema 3.2. Entonces la sucesión de iterados x_k generada por el algoritmo de Newton Truncado*

usando GCTM cumple que $\|g_k\| \to 0$ *cuando* $k \to \infty$, *es decir, el algoritmo es globalmente convergente.*

Demostración

De manera análoga a la demostración del Teorema 3.2 se tiene que $\cos(\theta_k) \geq \frac{\gamma_0}{\gamma_1}$, siendo θ_k el ángulo formado por las direcciones $-g_k$ y p_k con p_k la dirección de descenso obtenida por el algoritmo *GCTM* en la iteración k.

Teniendo en cuenta la condición iii), que $\gamma_0 = \min\{1, \frac{1}{\|H_k\|}\}$ y que $\gamma_1 = \mu_1(1+2\frac{\sqrt{\|H_k\|}}{\sqrt{\epsilon_c}})$ se obtiene la siguiente cota que es independiente de k

$$\cos(\theta_k) \geq \frac{\min\{1, \frac{1}{M}\}}{\mu_1(1+2\frac{\sqrt{M}}{\sqrt{\epsilon_c}})}$$

siendo $\mu_1 = \min\{1, \frac{n}{\epsilon_c}\}$.

Por lo tanto, como consecuencia del Teorema 3.1 se tiene que $\|g_k\| \to 0$ cuando $k \to \infty$.

◇

Además, continúan verificándose los Teoremas 3.3 y 3.4 de la sección anterior.

5 Experiencias numéricas y conclusiones

En esta sección se comparan los resultados numéricos obtenidos con los algoritmos implementados cuando se aplican a un conjunto de problemas test conocidos en optimización no lineal. Los problemas test que consideramos son los siguientes:

- *Función de Wood, dimensión* $n = 4$.
- *Función de Biggs, dimensión* $n = 6$.
- *Función Rosenbrock Chained, dimensión* $n = 25$.
- *Función Rosenbrock Extendida, dimensión arbitraria con n par.*

- *Función Penalty, n arbitrario.*

Las funciones Penalty y Rosenbrock Extendida fueron corridas para $n = 1000$.

Para realizar las comparaciones se han implementado 2 algoritmos; el algoritmo *NTGC*, detallado en la sección 2, y el algoritmo de Newton Truncado usando el algoritmo *GCTM* considerando diferentes valores de b, denominado *NTGCmod-b*.

En todos los algoritmos se ha considerado el mismo criterio de terminación: $\|g_k\| < \epsilon$, con $\epsilon = 10^{-5}$.

Se consideran además las constantes $c_1 = 10^{-4}$, $c_2 = 0.7$ en el algoritmo GSRCH de Powell, y $\epsilon_c = 10^{-6}$ y $\eta_k = \min\{5\ 10^{-2},\ \|g_k\|\}$ en los algoritmos *GCT* y *GCTM*.

Para describir los resultados en las tablas se usa:

Iter: número de iteraciones del algoritmo para alcanzar el criterio de terminación.

Evalf: número de evaluaciones de la función.

Evalg: número de evaluaciones del gradiente.

Prod$_{\mathbf{Ab}}$: número de productos de matriz-vector.

Grnor: *norma*-2 del gradiente en el punto x^* que satisface el criterio de parada.

En las tablas que se presentan a continuación podrá observarse que, con respecto al algoritmo *NTGC*, el algoritmo *NTGCmod-b* produce mejoras significativas tanto en número de iteraciones como en cantidad de evaluaciones de la función y del gradiente, y de productos matriz por vector.

La siguiente tabla corresponde a los resultados obtenidos al correr los algoritmos con la función Rosenbrock Extendida.

ROS. EXT.	NTGC	NTGCmod-$\frac{1}{2}$	NTGCmod-$\frac{3}{4}$	NTGCmod-$\frac{5}{4}$
Iter	60	24	24	23
Evalf	72	31	30	28
Evalg	72	31	30	28
Prod_{Ab}	114	41	41	41
Grnor	$2.48\ 10^{-8}$	$1.85\ 10^{-6}$	$7.88\ 10^{-7}$	$2.22\ 10^{-7}$

Cuando se corrieron estos test se observó que en 35 oportunidades el algoritmo *NTGC* encontró direcciones de curvatura negativa, mientras que con los distintos valores de b propuestos el algoritmo *NTCGmod-b* sólo encontró 1 dirección de curvatura negativa. Esto nos hace pensar que la función abandona rápidamente la región en la cual no es convexa.

La siguiente tabla corresponde a los resultados obtenidos al correr los algoritmos con la función Wood.

WOOD	NTGC	NTGCmod-$\frac{1}{2}$	NTGCmod-$\frac{3}{4}$	NTGCmod-$\frac{5}{4}$
Iter	135	43	41	41
Evalf	147	72	58	56
Evalg	147	72	58	56
Prod_{Ab}	522	152	145	143
Grnor	$6.33\ 10^{-12}$	$2.25\ 10^{-12}$	$2.08\ 10^{-12}$	$7.88\ 10^{-7}$

Como en el caso anterior, se observó que la cantidad de veces en las que se encuentran direcciones de curvatura negativa es notoriamente menor al utilizar los algoritmos *NTGCmod-b* que al utilizar el algoritmo original. De 102 veces se redujo a 5 en los casos en los que el algoritmo *NTGCmod-b* propone direcciones de curvatura positiva como

nueva salida, es decir $b = 0.5$ y $b = 0.75$. Obtuvo sólo 1 vez curvatura negativa en el caso $b = 1.25$, correspondiente a usar como nueva salida una dirección de curvatura negativa.

La siguiente tabla corresponde a los resultados obtenidos al correr los algoritmos con la función Biggs.

BIGGS	**NTGC**	**NTGCmod-$\frac{1}{2}$**	**NTGCmod-$\frac{3}{4}$**	**NTGCmod-$\frac{5}{4}$**
Iter	1960	62	63	73
Evalf	9190	155	123	195
Evalg	9190	155	123	195
Prod_{Ab}	11594	254	293	324
Grnor	$6.94\ 10^{-6}$	$9.84\ 10^{-6}$	$3.73\ 10^{-6}$	$7.01\ 10^{-6}$

Puede observarse en esta tabla que usando los algoritmos *NTGCmod-b* se obtienen resultados sorprendentemente mejores que usando el algoritmo *NTGC*.

Al observar la gran diferencia entre los resultados debido a la poca estabilidad de la función Biggs y a la magnitud de las direcciones obtenidas por *GCT*, se propuso utilizar las direcciones originales de los algoritmos *GCT* y *GCTM* con la mitad de magnitud obteniéndose los siguientes resultados.

BIGGS	NTGC	NTGCmod-$\frac{1}{2}$	NTGCmod-$\frac{3}{4}$	NTGCmod-$\frac{5}{4}$
Iter	125	81	82	83
Evalf	148	91	110	103
Evalg	148	91	110	103
Prod$_{Ab}$	535	398	427	422
Grnor	$7.52\ 10^{-6}$	$6.92\ 10^{-6}$	$9.88\ 10^{-6}$	$9.92\ 10^{-6}$

Otra propuesta fue utilizar las direcciones originales con norma 1, obteniéndose los siguientes resultados.

BIGGS	NTGC	NTGCmod-$\frac{1}{2}$	NTGCmod-$\frac{3}{4}$	NTGCmod-$\frac{5}{4}$
Iter	116	71	78	64
Evalf	332	221	212	176
Evalg	332	221	212	176
Prod$_{Ab}$	525	372	373	325
Grnor	$2.77\ 10^{-7}$	$3.48\ 10^{-6}$	$2.57\ 10^{-6}$	$4.35\ 10^{-6}$

En ambas tablas puede observarse que, si bien ya no existe tanta diferencia entre los resultados, los algoritmos *NTGCmod-b* siguen mejorando todos los aspectos en comparación con el algoritmo *NTGC*. También se redujo el número de veces en las que los algoritmos encuentran direcciones de curvatura negativa.

Al probar el comportamiento de los algoritmos con la función Rosenbrock Chained, para los diferentes b propuestos, el algoritmo *NTGCmod-b* mantuvo el mismo número de iteraciones que el algoritmo *NTGC* como también el mismo número de direcciones de curvatura negativa. Sin embargo se ha notado un pequeño aumento en cuanto a la cantidad de evaluaciones de la función y del gradiente y de productos matriz por vector.

No se corrió el algoritmo modificado con la función Penalty debido a que ésta nunca encuentra direcciones de curvatura negativa.

6 Apéndice

En este apéndice se describe como Lucidi, Rochetich y Roma [2], utilizan las direcciones de curvatura negativa en la resolución del problema

$$\min_{x \in \Re^n} f(x).$$

Desde el punto de vista teórico han considerado que buenas direcciones de curvatura negativa serán aquellas que garanticen que el algoritmo pueda abandonar rápidamente las regiones donde la función es estrictamente no convexa. Estas direcciones son la base para definir algoritmos que son convergentes a puntos estacionarios con Hessiano semidefinido positivo. Para que el algoritmo cumpla esta condición es suficiente pedirle a la dirección d_k de curvatura negativa buscada que cumpla

$$d_k^t H_k d_k \to 0 \text{ implica } \min\{0, \lambda_{min}(H_k)\} \to 0 \tag{6.10}$$

donde $\lambda_{min}(H_k)$ es el menor autovalor del hessiano H_k. Esta condición garantiza que el algoritmo finalizará en un punto donde $H(x^*)$ es semidefinido positivo ya que su menor autovalor será no negativo. Cuando la dirección d_k es un autovector asociado con el menor autovalor de H_k la condición (6.10) se verifica trivialmente. Esto motivó en [2] a buscar una dirección de curvatura negativa que sea similar a un autovector de H_k asociado con el menor autovalor. Consideran que el cálculo de una dirección de curvatura negativa d_k que satisface la condición (6.10) no es necesario ni está justificado hacerlo cuando el gradiente

es grande. Debido a esto consideran una condición más débil y suficiente para calcular la dirección deseada que es la siguiente

$$\begin{cases} d_k^t H_k d_k \to 0 \\ g_k \to 0 \end{cases} \qquad \text{implica} \qquad \min\{0, \lambda_{min}(H_k)\} \to 0 \tag{6.11}$$

Es decir, la dirección d_k debería ser una dirección de descenso de curvatura negativa que tenga similitud con un autovector de H_k correspondiente al autovalor más negativo y esta similitud debería ser más fuerte cuando el gradiente es más pequeño en norma.

El método de minimización propuesto en [2] está basado en una búsqueda lineal curvilinea no monótona de la forma $x_{k+1} = x_k + \alpha_k^2 s_k + \alpha_k d_k$ donde α_k es la longitud de paso calculada por una técnica del tipo Armijo, s_k es una dirección del tipo Newton que asegura convergencia global y tasa superlineal en la convergencia local, y d_k es una dirección de curvatura negativa que garantiza convergencia hacia puntos estacionarios donde la matriz hessiana es semidefinida positiva.

Se calcula s_k resolviendo aproximadamente el sistema $H_k s = -g_k$ de manera que se satisfagan las siguientes condiciones

Condición 1: Las direcciones s_k son tales que $g_k^t s_k \leq 0$ y

$$g_k^t s_k \to 0 \;\Rightarrow g_k \to 0 \; y \; s_k \to 0.$$

Condición 2: Las direcciones s_k son tales que

$$s_k \to 0 \text{ implica } g_k \to 0.$$

Se calcula d_k de manera que se satisfaga la siguiente condición

Condición 3: Las direcciones d_k son tales que

$$g_k^t d_k \leq 0, \quad d_k^t H_k d_k \leq 0,$$

$$\begin{cases} d_k^t H_k d_k \to 0 \\ g_k \to 0 \end{cases} \text{ implica } \begin{cases} \min\{0, \lambda_{min}(H_k)\} \to 0 \\ d_k \to 0 \end{cases}$$

El algoritmo de Lanczos es el método iterativo usado en [2] para calcular tanto una dirección s_k de Newton Truncado como una dirección d_k de curvatura negativa que sea una "buena" aproximación del autovector correspondiente al menor autovalor de H_k.

A continuación se describe el esquema general del algoritmo de Lanczos [2].

Algoritmo de Lanczos

Inicio: Dado $q_1 = -g_k$

Paso 1: $i = 1$, $v_0 = 0$, $\beta_i = \|q_i\|$

Paso 2: $v_i = \frac{1}{\beta_i} q_i$, $\alpha_i = v_i^t H_k v_i$,

$$q_{i+1} = H_k v_i - \alpha_i v_i - \beta_i v_{i-1}, \ \beta_{i+1} = \|q_{i+1}\|$$

Paso 3: Si $|\beta_{i+1}| = 0$ termina; sino $i : i + 1$ y retorna al Paso 2.

En la i-ésima iteración se tienen las siguientes matrices:

$$V_i = \begin{bmatrix} v_1 & v_2 & \vdots & \vdots & v_i \end{bmatrix}, T_i = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \dots & 0 \\ \beta_2 & \alpha_2 & \ddots & 0 & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & 0 & \ddots & \alpha_{i-1} & \beta_i \\ 0 & \dots & 0 & \beta_i & \alpha_i \end{bmatrix}$$

Propiedades del Algoritmo de Lanczos

Se enuncian las siguientes propiedades que pueden ser encontradas en [2].

i) Si g_k tiene proyección no nula sobre p autovectores de H_k y si los autovalores distintos correspondientes a esos p autovectores son $m \leq p$, entonces el algoritmo termina después de m iteraciones con $\beta_{m+1} = 0$, siendo $\beta_i \neq 0$, $i = 1, \ldots, m$.

ii) Si $\tilde{y}_i$ es la solución del sistema $T_i y = -V_i^t g_k$, y si se define

$$\tilde{s}_i = V_i \tilde{y}_i$$

entonces, si $\beta_{i+1} = 0$, el vector $\tilde{s}_i$ es una solución del sistema $H_k s = -g_k$. En el caso $\beta_{i+1} \neq 0$, el vector $\tilde{s}_i$ es solución del sistema $V_i^t(H_k s + g_k) = 0$.

iii) Supongamos que H_k tiene m autovalores distintos, y que g_k tiene proyección no nula sobre m autovectores de H_k correspondientes a tales m autovalores distintos. Si μ_m es el menor autovalor de la matriz T_m producida por el algoritmo, y si w_m es el autovector correspondiente, entonces μ_m es también el menor autovalor de H_k, y su correspondiente autovector es

$$\tilde{d}_m = V_m w_m.$$

iv) Si μ_i es el menor autovalor de la matriz tridiagonal T_i, entonces existe un autovalor λ de H_k tal que $|\lambda - \mu_i| \leq \beta_{i+1}$.

v) Asumiendo que λ y μ_i son los menores autovalores de H_k y T_i respectivamente, entonces $\lambda \leq \mu_i \leq \lambda + \epsilon(i)$, donde $\epsilon(i)$ tiende rápidamente a cero.

La propiedad **i)** asegura que el algoritmo finaliza en un número finito de pasos con $\beta_{m+1} = 0$. La propiedad **ii)** muestra que el algoritmo puede obtener una aproximación de la dirección de Newton. Las propiedades **iii)**, **iv)** y **v)** muestran que el algoritmo puede usarse para obtener una aproximación al menor autovalor de H_k y al autovector correspondiente. En particular la propiedad **iii)** dice que el menor autovalor y el autovector correspondiente pueden ser obtenidos usando la matriz tridiagonal T_m, que de acuerdo a su estructura tridiagonal puede hacerse eficientemente.

En cada iteración el método de Lanczos provee dos vectores $\tilde{s}_i$ y $\tilde{d}_i = V_i w_i$ que son aproximaciones de la dirección de Newton y de un autovector correspondiente a la mínima curvatura de H_k respectivamente. El cálculo de $\tilde{d}_i$ requiere el conocimiento y guardado de la matriz V_i, lo que limita su aplicación cuando n es grande, mientras que $\tilde{s}_i$ puede obtenerse desde $\tilde{s}_{i-1}$ si se define $q_1 = -g_k$ (Paige and Saunders, Solution of sparse indefinite systems of linear equations, SIAM J. Numer. Anal., 12(1975). pp.617-629). El método SYMMLQ descripto por Paige y Saunders es usado en [2]. El límite en el guardado para V_i determina que los autores trunquen el algoritmo a pocas iteraciones L, considerando el siguiente esquema:

Algoritmo:

- *Iteraciones de Lanczos.*

  El método SYMMLQ se modifica para continuar las iteraciones hasta una i-ésima iteración tal que,

  - un criterio de terminación del método original se satisfaga, y

  - $|\beta_{i+1}| < \epsilon_k$ o si $i > L$ (donde L es el máximo de iteraciones permitida).

- *Cálculo de la dirección s_k.*

  - Si el sistema $H_k s = -g_k$ es resoluble y el vector $\tilde{s}_i$ producido por SYMMLQ satisface las Condiciones 1 y 2, entonces $s_k = \tilde{s}_i$.

  - De otra forma, se define $s_k = -g_k$.

- *Cálculo de la dirección d_k.*

  Calcula el menor autovalor μ_i de T_i,

  - Si $\mu_i < 0$, entonces calcula el correspondiente autovector w_i y define

$$d_k = -\rho_k sgn[(g_k)^t \bar{d}_i]\bar{d}_i,$$

donde $\bar{d}_i = V_i w_i$ y ρ_k es un factor de escala.

- De otra forma, define $d_k = 0$.

En sus experiencias numéricas muestran que el uso de direcciones de curvatura negativa en algoritmos monótonos da mejores resultados en términos del número de iteraciones y evaluaciones de la función y del gradiente pero no da diferencias significativas en términos de tiempo. Los beneficios del uso de direcciones de curvatura negativa son muy evidentes en el contexto no monótono, en este caso, buenos progresos en términos de tiempo fueron obtenidos.

Referencias

[1] R. Dembo, T. Steihaug, *Truncated-Newton Algorithms for Large-Scale Unconstrained Optimization*, Mathematical Programming 26, (1983), pp 190-212.

[2] S. Lucidi, F. Rochetich y M. Roma, *Curvilinear Stabilization Techniques for Truncated Newton Methods in Large Scale Unconstrained Optimization*, SIAM J. OPTIM., Vol.8 (1998), pp. 916-939.

[3] J. Nocedal, S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.

```
------------------------------------------------------------------------
      ALGORITMO DE NEWTON TRUNCADO (NT)
------------------------------------------------------------------------

      parameter(n=4,n1=n*(n+1)/2,n2=2*n)
      implicit real*8(a-h,o-z)
      dimension xk(n),gk(n),h(n,n),b(n),xk1(n),gk1(n),dk(n),ad0(n),
                d0(n),w(n2),p0(n),dk1(n),hes(n1)

------Constante utilizada por el algoritmo-----------------------------
      data eps1/1.d-5/

      nprod=0
      nfu=0
      ngra=0
      nhes=0
      call datos(n,xk)
------datos da el x0 inicial-------------------------------------------

      kf=0
      call rfunt(n,xk,rf)
------rfunt da el valor rf de f en xk----------------------------------

      nfu=nfu+1
      call gradie(n,xk,gk)
------gradie da el gradiente gk de f en xk-----------------------------

      ngra=ngra+1
100   grn=0.d0
      do i=1,n
         grn=grn+gk(i)**2
      end do
      grn=dsqrt(grn)

------Criterio de finalización-----------------------------------------
      if(grn.lt.eps1) go to 110

      kf=kf+1
      call hessia(n,n1,xk,hes,h)
------hessia da el Hessiano h de f en xk------------------------------

      nhes=nhes+1
      rk=kf
      do i=1,n
         p0(i)=0.d0
         b(i)=gk(i)
      end do
      grrn=grn


------------------------------------------------------------------------
      ALGORITMO DE GRADIENTES CONJUGADOS TRUNCADO (GCT)
------------------------------------------------------------------------
      call gct(n,h,p0,b,dk1,gk1,dk,ad0,d0,kiter,grrn,rk,ifli)

------------------------------------------------------------------------
-     dk1 es la salida de gct                                          -
```

```
-     ifli indica el tipo de salida                                          -
-        ifli= 1 (salida con curvatura positiva)                             -
-        ifli= 0 (salida con curvatura positiva próxima a cero)              -
-        ifli= -1 (salida con curvatura negativa)                            -
-----------------------------------------------------------------------------

      nprod=nprod+kiter
      do i=1,n
         dk(i)=dk1(i)
      end do

------Constantes utilizadas por el algoritmo GSRCH de Powell------------
      grhtol=0.1d0
      step=1.d0
      stmin=1.d-10
      stmax=1.d8
      nmax=50

-----------------------------------------------------------------------------
      ALGORITMO DE POWELL (GSRCH)
-----------------------------------------------------------------------------
      call gsrch(n,xk,rf,gk,dk,step,stmin,stmax,nmax,grhtol,iflag,w,
                 nfu,ngra,n2)
------xk y gk son las salidas de gsrch---------------------------------

      go to 100
110   end
```

```
------------------------------------------------------------------------
      ALGORITMO DE GRADIENTES CONJUGADOS TRUNCADO (GCT)
------------------------------------------------------------------------

      subroutine gct(n,a,x0,g0,x1,g1,d0,ad0,d,kiter,grrn,rk,iflag)
      implicit real*8(a-h,o-z)
      dimension x0(n),g0(n),a(n,n),x1(n),g1(n),d0(n),ad0(n),d(n),

------Constantes utilizadas por el algoritmo GCT-----------------------
      data epsc/1.d-6/eps2/5.d-2/

      k=0
      g0n=0.d0
      do i=1,n
         g0n=g0n+g0(i)**2
      end do
      do i=1,n
         d0(i)=-g0(i)
      end do
100   k=k+1
      call produc(n,a,d0,ad0)
------produc realiza los productos matriz por vector-------------------

      cla=0.d0
      da0=0.d0
      do i=1,n
         cla=cla+ d0(i)*d0(i)
         da0=da0+d0(i)*ad0(i)
      end do

      if(da0.lt.0.d0) then
            iflag=-1
------------d0 es una dirección de curvatura negativa------------------
            if(k.eq.1) then
                  do i=1,n
                     x1(i)=-g0(i)
                  end do
                  go to 50
------------------sale con d0 en la primera iteración------------------
            else
                  go to 50
------------------sale con el iterado previo---------------------------
            end if
      else
            if(da0.le.epsc*cla .and. da0.ge.0.d0) then
                  iflag=0
------------------d0 es una dirección de curvatura positiva próxima a----
-----------------cero-------------------------------------------------
                  if(k.gt.1) go to 50
------------------sale con el iterado previo---------------------------
                  do i=1,n
                     x1(i)=-g0(i)
                  end do
                  go to 50
------------------sale con d0 en la primera iteración------------------
            end if
      end if
```

```
      iflag=1
------d0 es una dirección de curvatura positiva-------------------------
      w12=g0n/da0
      g1n=0.d0
      do i=1,n
         x1(i)=x0(i)+w12*d0(i)
         g1(i)=g0(i)+w12*ad0(i)
         g1n=g1n+g1(i)**2
      end do
      ggg=dsqrt(g1n)/grrn


------Criterio de finalización de Gradientes Conjugados Truncado--------
      eta=dmin1(eps2,grrn)
      if(ggg.lt.eta) go to 50
------sale con curvatura positiva---------------------------------------

      beta=g1n/g0n
      do i=1,n
         d0(i)=-g1(i)+beta*d0(i)
      end do
      do i=1,n
         x0(i)=x1(i)
         g0(i)=g1(i)
      end do
      g0n=g1n
      go to 100
50    kiter=k
      return
      end
```

```
------------------------------------------------------------------------
      ALGORITMO DE GRADIENTES CONJUGADOS TRUNCADO MODIFICADO (GCTM)
------------------------------------------------------------------------

      subroutine gctm(n,a,x0,g0,x1,g1,d0,ad0,d,kiter,grrn,rk,iflag)
      implicit real*8(a-h,o-z)
      dimension x0(n),g0(n),a(n,n),x1(n),g1(n),d0(n),ad0(n),d(n),

------Constantes utilizadas en el algoritmo GCTM-----------------------
      data epsc/1.d-6/eps2/5.d-2/b/.5d0/

      k=0
      g0n=0.d0
      do i=1,n
         g0n=g0n+g0(i)**2
      end do
      do i=1,n
         d0(i)=-g0(i)
      end do
      dala=0.d0
100   k=k+1
      call produc(n,a,d0,ad0)
------produc realiza los productos matriz por vector-------------------

      cla=0.d0
      da0=0.d0
      do i=1,n
         cla=cla+d0(i)*d0(i)
         da0=da0+d0(i)*ad0(i)
      end do
      if(da0.lt.-epsc*cla) then
            iflag=-1
------------d0 es una dirección de curvatura negativa------------------
            if(k.eq.1) then
                  do i=1,n
                     x1(i)=-g0(i)
                  end do
                  go to 50
------------------sale con d0 en la primera iteración------------------
            else
                  alpha=dsqrt(dala/(-da0))
------------------alpha realiza el cambio de curvatura-----------------
                  do i=1,n
                     x1(i)=x0(i)+b*alpha*d0(i)
                  end do
                  go to 50
------------------sale con la nueva propuesta--------------------------
            end if
      end if


      if(da0.lt.0.d0 .and. da0.ge.-epsc*cla) then
            iflag=-1
------------d0 es una dirección de curvatura negativa próxima a cero----
            if(k.gt.1) go to 50
------------sale con el iterado previo---------------------------------
            do i=1,n
```

```
               x1(i)=-g0(i)
            end do
            go to 50
------------sale con d0 en la primera iteración-----------------------
      else
            if(da0.le.epsc*cla .and. da0.ge.0.d0) then
                  iflag=0
------------------d0 es una dirección de curvatura positiva próxima a----
-----------------cero-------------------------------------------------
                  if(k.gt.1) go to 50
------------------sale con el iterado previo---------------------------
                  do i=1,n
                     x1(i)=-g0(i)
                  end do
                  go to 50
------------------sale con d0 en la primera iteración------------------
            end if
      end if


      iflag=1
------d0 es una dirección de curvatura positiva-----------------------
      w12=g0n/da0
      g1n=0.d0
      do i=1,n
         x1(i)=x0(i)+w12*d0(i)
         g1(i)=g0(i)+w12*ad0(i)
         g1n=g1n+g1(i)**2
      end do
      ggg=dsqrt(g1n)/grrn
      da1a=da1a+(w12)**2*da0


------Criterio de finalización de Gradientes Conjugados Truncado--------
      eta=dmin1(eps2,grrn)
      if(ggg.lt.eta) go to 50
------sale con curvatura positiva-------------------------------------

      beta=g1n/g0n
      do i=1,n
         d0(i)=-g1(i)+beta*d0(i)
      end do
      do i=1,n
         x0(i)=x1(i)
         g0(i)=g1(i)
      end do
      g0n=g1n
      go to 100
50    kiter=k
      return
      end
```