UN MÉTODO DE OPTIMIZACIÓN POR REDUCCIÓN DEL HESSIANO PARA CAMPOS ESCALARES Y SU APLICACIÓN A LAS GRAMÁTICAS LIBRES DE CONTEXTO

Esther Sofía Mamián López

Universidad del Cauca
Facultad de las Ciencias Naturales, Exactas y de la Educación
Departamento de Matemáticas
Popayán
2010

UN MÉTODO DE OPTIMIZACIÓN POR REDUCCIÓN DEL HESSIANO PARA CAMPOS ESCALARES Y SU APLICACIÓN A LAS GRAMÁTICAS LIBRES DE CONTEXTO

Esther Sofía Mamián López

Trabajo de investigación

Fredy Angel Amaya Robayo
Doctor en Ciencias de la Computación

Universidad del Cauca
Facultad de las Ciencias Naturales, Exactas y de la Educación
Departamento de Matemáticas
Popayán
2010

Notas de aceptación

_____

_____

_____

_____

_____

_____

Doctor Fredy Angel Amaya Robayo

_____

Magister Tulio Emiro López Erazo

_____

Especialista Mat. Aplicada Hebert Vivas

Popayán, 12 de Febrero de 2010

A Emiro Mamián y Aydeé López por señalar caminos de constante conocimiento y enseñar a descubrir en cada paso nuevas busquedas.

AGRADECIMIENTOS

Este documento es el resultado del trabajo colectivo de varias personas que compartieron ideas, propuestas y conocimientos.

Gracias a Fredy Amaya Robayo por su constante aporte, enseñanza compartida, eje de todo el proceso. A los Estudiantes y profesores vinculados al proyecto de investigación por todo su apoyo y acompañamiento. A Carlos Alberto Trujillo por el apoyo durante el desarrollo experimental de la investigación. A Tulio Emiro López y Hebert Vivas por sus aportes y contribuciones finales.

Índice general

Índice de tablas

Índice de figuras

PRESENTACIÓN

Dentro del desarrollo del proyecto de investigación que adelanta en la actualidad el grupo de Matemática Aplicada, Línea de Matemática Computacional (proyecto 1579 de la Vicerrectoría de Investigaciones; "NUEVAS ALTERNATIVAS PARA LA ESTIMACIÓN DE LOS PARÁMETROS EN UNA GRAMÁTICA INCONTEXTUAL PROBABILÍSTICA"), en el cual se enmarca este trabajo, se propone abordar el problema de la eficacia de la estimación de los parámetros de una gramática incontextual probabilística, desde diferentes puntos de vista, en primer lugar desde la teoría de los momentos, luego a partir del punto de vista clásico basado en el teorema de transformaciones crecientes, después con el algoritmo Inside-Outside y por último desde métodos cuasi-Newton para problemas de optimización a gran escala. Este último se analiza en este trabajo.

INTRODUCCIÓN

Una gramática formal es un objeto o modelo matemático que permite especificar un lenguaje, es decir, es el conjunto de reglas capaces de generar todas las posibles combinaciones de ese lenguaje, ya sea éste un lenguaje formal o un lenguaje natural. Las gramáticas formales han sido de gran utilidad en aplicaciones prácticas; diseño de compiladores, simulación en biología, lenguajes de programación para computadoras, reconocimiento Sintáctico de Formas, Inferencia Gramatical, como modelo sintáctico de lenguajes naturales, entre otros.

Una Gramática Incontextual Probabilística (GIP) es una extensión natural de una Gramática Incontextual (GI) que se compone básicamente de dos partes: un conjunto de reglas (gramática característica) que conforman la parte estructural de la misma y una función de distribución de probabilidad (o simplemente probabilidades) asociadas a las reglas que constituyen su parte estocástica [And99, And08].

El proceso de aprendizaje de las Gramáticas Incontextuales Probabilísticas (GIP) puede descomponerse en el aprendizaje de la gramática característica y/o el aprendizaje de las probabilidades asociadas a las reglas. Con el aprendizaje de la gramática característica se pretende recoger la información estructural presente en la muestra, mientras que con el aprendizaje de las probabilidades se pretende recoger básicamente la información estocástica [And99].

En el aprendizaje de las probabilidades de una GIP, en primer lugar se debe definir alguna función criterio a optimizar dependiente de la muestra. Esta muestra puede estar formada por cadenas pertenecientes al lenguaje, o por cadenas pertenecientes y no pertenecientes al lenguaje, en el caso particular se trabaja con muestras pertenecientes al lenguaje. Una vez definida la función objetivo, se elige un proceso para optimizarla.

El problema a tratar ahora es el de estimar los parámetros de una GIP. Dada una muestra del lenguaje bajo estudio, encontrar las probabilidades de las reglas, de tal forma que se maximice la función de verosimilitud de la muestra. Como la función de verosimilitud es un polinomio en varias variables, y el teorema de transformaciones crecientes, algoritmo clásico de estimación para encontrar el máximo de tal función, tiene un alto costo computacional [And99, Rob04], se explora el uso de métodos cuasi-Newton para la obtención de los valores de los parámetros (probabilidades de las reglas) de la gramática incontextual probabilística que maximicen la función de verosimilitud.

En éste trabajo se va a estudiar el uso del método de optimización descrito en [TJC], que esta diseñado para optimizar funciones con un número alto de variables y de restricciones. Se espera proporcionar, al proyecto investigativo, “NUEVAS ALTERNATIVAS

PARA LA ESTIMACIÓN DE LOS PARÁMETROS EN UNA GRAMÁTICA INCONTEXTUAL PROBABILÍSTICA", una buena herramienta, en cuanto a reducción de costos computacionales, como alternativa para estimar los parámetros de una gramática incontextual probabilística (GIP). También se proporcionarán resultados experimentales, con base en varias gramáticas preestablecidas, que permitirán realizar fácilmente comparaciones con resultados obtenidos con otros métodos desarrollados dentro del trabajo investigativo [Rob04].

El contenido de esta monografía se divide en cinco capítulos, en el primero se realiza una breve revisión al concepto de grámaticas incontextuales. En el segundo capítulo se expone la teoría de optimización necesaria para abordar el tema del tercer capítulo, en el cual se estudia el algoritmo propuesto en [TJC]. En el cuarto capítulo se realizan los experimentos y se detalla la metodología utilizada para tal fin. En el quinto y último capítulo se exponen los resultados obtenidos en los experimentos realizados, se presentan las conclusiones y se sugieren trabajos futuros.

Capítulo 1

Gramáticas Incontextuales

1.1. Introducción

La lingüística computacional es una rama de la informática que usa modelos matemáticos y herramientas computacionales, con el fin de mejorar los sistemas automáticos que procesan diferentes aspectos del manejo del lenguaje, tanto natural como formal. Entre los lenguajes formales utilizados tanto para procesar lenguaje natural como para otras aplicaciones, están los generados por las Gramáticas Incontextuales Probabilísticas (GIPs). La importancia de las GIPs radica en que el tipo de modelo que produce provee mayor información que otros modelos, pero en contraste, el costo computacional que requieren para la estimación de los parámetros en tareas de cierta complejidad es elevado. Hasta ahora, los modelos matemáticos y algoritmos propuestos y utilizados en la estimación de los parámetros de las GIPs, si bien producen modelos de lenguaje con una buena capacidad expresiva, su costo computacional es elevado, por lo cual se requiere buscar métodos más eficientes para realizar la estimación de los parámetros.

En lo que resta del capítulo se hará un breve estudio y estimación de las GIPs, y se planteará el problema de optimizar la función de verosimilitud asociada a las reglas de la GIP.

1.2. Cadenas, Alfabetos y Lenguajes

Definición 1.2.0.1 *Conceptos básicos sobre cadenas, alfabetos y lenguajes.*

- *Un alfabeto o vocabulario, notado como Σ, es un conjunto finito de símbolos.*
- *Una cadena (o frase) es una secuencia finita de símbolos (elementos de Σ).*

- *La longitud de una cadena x es el número de símbolos que tiene, se denota $|x|$.*
- *Una cadena vacía es aquella que no posee símbolos y se denota ϵ ($|\epsilon| = 0$).*
- *Por Σ^* se entiende el conjunto de todas las cadenas de Σ.*
- *Por Σ^+ se entiende el conjunto de todas las cadenas de símbolos de Σ que tienen longitud mayor o igual que 1.*
- *Un lenguaje L sobre Σ se define como un subconjunto del conjunto Σ^*.*

1.3. Gramáticas y Gramáticas Incontextuales

Definición 1.3.0.2 *Una gramática formal es una 4-tupla $G = (N, \Sigma, P, S)$, donde:*

N : *Conjunto finito de símbolos, denominado conjunto de variables o no terminales.*

Σ : *Es el alfabeto. N y Σ son disjuntos.*

P : *Conjunto finito de reglas de producción (o de derivación): cada producción es de la forma $\boldsymbol{\alpha \rightarrow \gamma}$, en donde $\boldsymbol{\alpha}$ (Antecedente) y $\boldsymbol{\gamma}$ (Consecuente) son cadenas de símbolos tomados de $(N \cup \Sigma)^*$.*

S : *Símbolo inicial (Axioma) de la gramática. $S \in N$.*

Notación: La expresión $\alpha \rightarrow \gamma$, donde α y γ son cadenas de $(N \cup \Sigma)^*$, significa que la cadena α deriva en la cadena γ o que α es sustituido por γ.

Notación: La expresión $\alpha \overset{i}{\Rightarrow} \beta$ significa que α deriva en β exactamente en i pasos. La expresión $\alpha_1 \overset{a_1}{\Rightarrow} \alpha_2 \overset{a_2}{\Rightarrow} \alpha_3 \overset{a_3}{\Rightarrow}, \cdots, \overset{a_{k-2}}{\Rightarrow} \alpha_{k-1} \overset{a_{k-1}}{\Rightarrow} \alpha_k$ significa que $\alpha_1 \overset{a_1}{\Rightarrow} \alpha_2$, $\alpha_2 \overset{a_2}{\Rightarrow} \alpha_3$,..., $\alpha_{k-1} \overset{a_{k-1}}{\Rightarrow} \alpha_k$. La expresión $\alpha \overset{*}{\Rightarrow} \beta$ significa que α deriva en β, mediante alguna sucesión de derivaciones de la forma $\alpha \overset{a_1}{\Rightarrow} \alpha_1 \overset{a_2}{\Rightarrow} \alpha_2 \cdots \alpha_k \overset{a_m}{\Rightarrow} \beta$.

Definición 1.3.0.3 *Una derivación a izquierda de una cadena $x \in \Sigma^+$ en G, es una sucesión de reglas de derivación $dx = (q_1, q_2, \ldots, q_m)$, $m \geq 1$, tal que: $\left(S \overset{q_1}{\Rightarrow} \alpha_1 \overset{q_2}{\Rightarrow} \alpha_2 \overset{q_3}{\Rightarrow}, \cdots, \overset{q_m}{\Rightarrow} x\right)$, donde α_1,$\alpha_i \in (N \cup \Sigma)^+$, $2 \leq i \leq m+1$ y q_i reescribe el terminal más a la izquierda de α_{i-1}*

En el siguiente ejemplo se aclara este punto:

Ejemplo 1.3.0.1 Sea $G = (N, \Sigma, P, S)$, donde $N = \{A, B, S\}$, $\Sigma = \{a, b\}$, S es el símbolo inicial y P consiste en:

$$\begin{aligned} S &\rightarrow aA \\ aB &\rightarrow bBAa \\ ABb &\rightarrow abB \end{aligned}$$

Note que la cadena $bbaBB$ se transforma en la cadena $bbbBAaB$ por la aplicación de la segunda regla.

En adelante utilizaremos las siguientes convenciones con respecto a las gramáticas:

- Las letras mayúsculas tales como A,B, C, D, E y S representan no terminales; S es el símbolo inicial.
- Las letras minúsculas como a, b, c, d, e y los dígitos serán los símbolos terminales.
- Las letras minúsculas u, v, w, s denotan cadenas de terminales.
- Las letras griegas minúsculas α, β, γ se usan para denotar cadenas de terminales y no terminales.

Definición 1.3.0.4 *Una Gramática G, para la cual el conjunto P de reglas de derivación esta constituido por reglas de la forma $A \rightarrow \alpha$ donde $A \in N$ y $\alpha \in (N \cup \Sigma)^+$, se denomina Gramática Incontextual (GI).*

Definición 1.3.0.5 *Una Gramática Incontextual G se encuentra en Forma Normal de Chomsky (FNC), si las reglas de derivación tienen la forma $A \rightarrow BC$ o $A \rightarrow v$ donde $A, B, C \in N$ y $v \in \Sigma$.*

Nota: En adelante cuando se hable de derivación, se entenderá como derivación a izquierda.

Definición 1.3.0.6 *El lenguaje generado por una gramática G es el conjunto $L(G)$, definido como:*

$$L(G) = \left\{ x \in \Sigma^+ \mid S \stackrel{*}{\Rightarrow} x \right\} \tag{1.3.1}$$

Definición 1.3.0.7 *Un lenguaje probabilístico sobre un alfabeto Σ es un par (L, Φ), con L un lenguaje formal y Φ medida de probabilidad sobre Σ^*, talque $\Phi(x) > 0$ si $x \in \Sigma^*$*

Definición 1.3.0.8 *Una Gramática Incontextual Probabilística (GIP) G_p es una pareja (G, p) tal que G es una Gramática Incontextual y p una función $p : P \rightarrow (0, 1]$ sobre las reglas de la gramática de tal forma que:*

$$\forall A \in N, \sum_{(A \rightarrow \alpha) \in P} p(A \rightarrow \alpha) = 1 \tag{1.3.2}$$

1.4. Probabilidad de una cadena de una GIP

Sea G_p una GIP. Para cada $x \in L(G)$ se denomina D_x al conjunto formado por todas las derivaciones a izquierda de la cadena x. Por $N(A \to \alpha, d_x)$ se designa el número de veces que la regla $A \to \alpha$ se ha utilizado en la derivación d_x.

Definición 1.4.0.9 *Dada una GIP G_p, la probabilidad de una derivación d_x de la cadena $x \in \Sigma^*$ se define como:*

$$\Pr(x, d_x|G_p) = \prod_{(A\to\alpha)\in P} p(A \to \alpha)^{N(A\to\alpha, d_x)} \tag{1.4.3}$$

Definición 1.4.0.10 *Dada una GIP G_p, la probabilidad de la cadena $x \in \Sigma^*$ se define como:*

$$\Pr(x|G_p) = \sum_{d_x \in D_x} \Pr(x, d_x|G_p) \tag{1.4.4}$$

Definición 1.4.0.11 *Dada una GIP G_p, Se llama probabilidad de la mejor derivación de la cadena x a:*

$$\widehat{\Pr}(x|G_p) = \max_{d_x \in D_x} \Pr(x, d_x|G_p) \tag{1.4.5}$$

Definición 1.4.0.12 *El lenguaje generado por una gramática incontextual probabilística $G_p = (G, p)$, se define como $L(G_p) = \{x \in L(G) : \Pr(x|G_p) > 0\}$. Una GIP G_p se denomina consistente, si el lenguaje generado por G_p es estocástico, es decir:*

$$\sum_{x\in L(G_p)} \Pr(x|G_p) = 1. \tag{1.4.6}$$

1.5. Estimación de los Parámetros de una GIP

El problema que concierne es determinar las probabilidades de las reglas de la gramática, o como suele llamarse, problema de estimación de los parámetros de una GIP.

El problema de la estimación de los parámetros de una GIP $G_p = (G, p)$ puede plantearse de la siguiente manera: sea (L, Φ) un lenguaje estocástico donde $L \subseteq L(G)$, y Φ es una función de probabilidad desconocida. Dada una muestra Ω de L, se tiene que estimar la función p de tal manera que mediante ella se pueda representar a Φ. El problema planteado queda de la forma:

$$\widehat{p} := \max_{p} f_p(\Omega) \tag{1.5.7}$$

donde f_p es una función a ser optimizada. De este modo, para estimar los parámetros de una GIP es necesario determinar la función a optimizar y el método de optimización que se ha de utilizar. El método de optimización de uso más generalizado para la estimación de los parámetros en una GIP, esta basado en el teorema de transformaciones crecientes.

Teorema 1.5.0.1 (Transformaciones crecientes) *Sea $P(\Theta)$, con $\Theta = \{\Theta_{ij}\}$, un polinomio homogéneo de grado d con coeficientes no negativos. Sea $\theta = \{\theta_{ij}\}$ un punto del conjunto*

$$D = \left\{ \theta_{ij} \mid \theta_{ij} \geq 0,\ \sum_{j=1}^{q_i} \theta_{ij} = 1,\ i = 1, \ldots, k\ j = 1, \ldots, q_i \right\}$$

para k, q_i enteros. Sea $Q(\Theta)$ un punto del conjunto D definido como:

$$Q(\theta_{ij}) = \frac{\theta_{ij}\left(\frac{\partial P}{\partial \Theta_{ij}}\right)_\theta}{\sum_{j=1}^{q_i} \theta_{ik}\left(\frac{\partial P}{\partial \Theta_{ik}}\right)_\theta}$$

tal que para todo i, $\sum_{j=1}^{q_i} \theta_{ik}\left(\frac{\partial P}{\partial \Theta_{ik}}\right)_\theta \neq 0$. Entonces $P(Q(\theta)) > P(\theta)$ para $Q(\theta) \neq \theta$.

La aplicación del teorema anterior permite obtener un máximo local del polinomio P en el espacio de búsqueda definido por D haciendo uso del siguiente algoritmo de estimación:

Algoritmo de estimación

Entrada	$P(\Theta)$
Salida	Θ
Método	$\Theta =$ `valores iniciales`
	repetir `calcular` $Q(\Theta)$ `utilizando` $P(\Theta)$
	$\Theta = Q(\Theta)$
	Hasta `Converger.`

Dado un polinomio que cumple las condiciones del teorema 1 y un punto inicial $\theta \in D$, el problema de la estimación se plantea como la aplicación repetida de la transformación hasta alcanzar un máximo local del polinomio definido, ó hacer la estimación hasta que la diferencia entre 2 iteraciones consecutivas esté por debajo de un umbral. Note que las probabilidades de las reglas de una GIP pueden ser puntos del dominio definido en el teorema 1 para una GIP G_p:

$$p(A_i \to \alpha_j) = \theta_{ij} \quad i = 1, \ldots, |N|\,; \quad j = 1, \ldots, |\Gamma_{A_i}| \tag{1.5.8}$$

donde Γ_{A_i} representa el conjunto de reglas de la gramática cuyo antecedente es A_i.

1.5.1. Función de Verosimilitud

Entre las funciones objetivo a optimizar está la función de verosimilitud [And99].

Para el caso de las GIPs, la función de verosimilitud de una muestra Ω de $L(G)$ notada $\Pr(\Omega|G_p)$, está dada por la ecuación:

$$\Pr(\Omega|G_p) = \prod_{x\in\Omega} \Pr(x|G_p). \tag{1.5.9}$$

Si se utiliza la función de verosimilitud como función objetivo en (1.5.7), puede notarse que dicha función es un polinomio en varias variables, si se ordenan las reglas de alguna forma y a la regla i-ésima se le asigna la variable x_i, $i = 1, 2, \ldots, |P|$, lo cual se ilustra con el siguiente ejemplo:

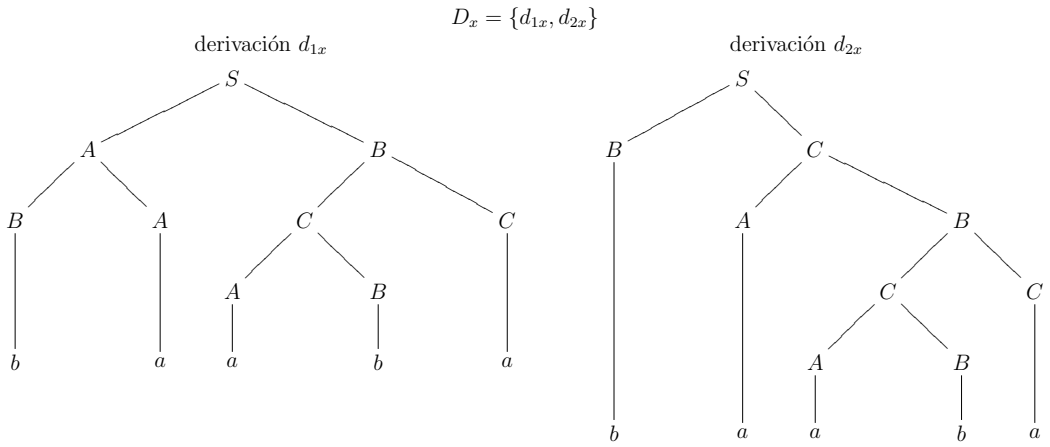
Ejemplo 1.5.1.1 Sea $G = (N, \Sigma, P, S)$, donde $N = \{A, B, C, S\}$, $\Sigma = \{a, b\}$, S es el símbolo inicial y P consiste en:

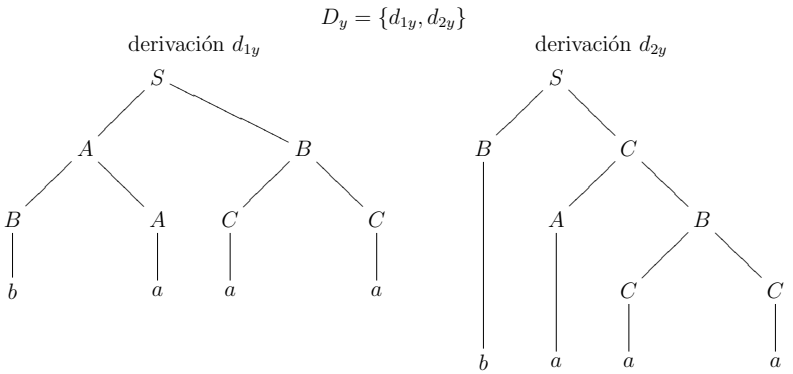
1) $S \to AB$	3) $A \to BA$	5) $B \to CC$	7) $C \to AB$
2) $S \to BC$	4) $A \to a$	6) $B \to b$	8) $C \to a$

Sea x_i la probabilidad de la regla i-ésima de la gramática, es decir:

$x_1 = p(S \to AB)$	$x_3 = p(A \to BA)$	$x_5 = p(B \to CC)$	$x_7 = p(C \to AB)$
$x_2 = p(S \to BC)$	$x_4 = p(A \to a)$	$x_6 = p(B \to b)$	$x_8 = p(C \to a)$

Sea $\Omega = \{baaba, baaa\} \subset L(G)$. Sean $x = baaba$ e $y = baaa$, los árboles de derivación para los elementos de la muestra son:





El número de veces N que se usa cada regla es:

	D_x		D_y	
Regla	$N(d_{1x})$	$N(d_{2x})$	$N(d_{1y})$	$N(d_{2y})$
$S \to AB$	1	0	1	0
$S \to BC$	0	1	0	1
$A \to BA$	1	0	1	0
$A \to a$	2	2	1	1
$B \to CC$	1	1	1	1
$B \to b$	2	2	1	1
$C \to AB$	1	2	0	1
$C \to a$	1	1	2	2

De la ecuación (1.4.3) se tiene que:

$$\Pr(x, d_{1x}|G_p) = x_1x_3x_4^2x_5x_6^2x_7x_8 \qquad \Pr(y, d_{1y}|G_p) = x_1x_3x_4x_5x_6x_8^2$$
$$\Pr(x, d_{2x}|G_p) = x_2x_4^2x_5x_6^2x_7^2x_8 \qquad \Pr(y, d_{2y}|G_p) = x_2x_4x_5x_6x_7x_8^2$$

Luego, de la ecuación (1.4.4), se tiene que:

$$\Pr(x|G_p) = \Pr(x, d_{1x}|G_p) + \Pr(x, d_{2x}|G_p) = x_1x_3x_4^2x_5x_6^2x_7x_8 + x_2x_4^2x_5x_6^2x_7^2x_8$$
$$\Pr(y|G_p) = \Pr(y, d_{1y}|G_p) + \Pr(y, d_{2y}|G_p) = x_1x_3x_4x_5x_6x_8^2 + x_2x_4x_5x_6x_7x_8^2$$

Ahora, de la ecuación (1.5.9) tenemos que la función de verosimilitud de la muestra Ω esta dada por:

$$\begin{aligned}
\Pr(\Omega|G_p) &= \Pr(x|G_p) * \Pr(y|G_p) \\
&= (x_1x_3x_4^2x_5x_6^2x_7x_8 + x_2x_4^2x_5x_6^2x_7^2x_8) * (x_1x_3x_4x_5x_6x_8^2 + x_2x_4x_5x_6x_7x_8^2) \\
&= x_1^2x_3^2x_4^3x_5^2x_6^3x_7x_8^3 + x_1x_2x_3x_4^3x_5^2x_6^3x_7^2x_8^3 \\
&\quad + x_1x_2x_3x_4^3x_5^2x_6^3x_7^2x_8^3 + x_2^2x_4^3x_5^2x_6^3x_7^3x_8^3
\end{aligned}$$

Dado que la función $\Pr(\Omega|G_p)$ es un polinomio en varias variables, se tiene que el problema de estimación de los parámetros de la GIP se convierte en el problema de optimizar un polinomio en varias variables sujeto a restricciones lineales de la forma: $0 \leq x_i \leq 1$, $i = 1, 2, \ldots, |P|$ y $\sum_{x_i \in \Psi_\alpha} x_i = 1$, donde Ψ_α es el conjunto de todas las x_i que representan la probabilidad de una regla cuyo antecedente es α.

En el año de 1995, Lorenz Biegler, Jorge Nocedal y Claudia Schmid, publicaron en la en la SIAM Journal on Optimization, un algoritmo basado en el método de reducción hessiana que optimiza polinomios en varias variables con restricciones de igualdad. Este algoritmo fue desarrollado para ser eficiente con problemas donde el número de variables n es grande y $n - m$ pequeño, donde m es el número de restricciones. En el presente trabajo se estudia y se implementa éste algoritmo para encontrar una solución al problema 1.5.7.

Capítulo 2

Optimización

2.1. Introducción

La optimización como principio en el que se basa el análisis de numerosos y complejos problemas de decisión, centrando la atención en una función objetivo diseñada para cuantificar los resultados y medir la calidad de la decisión tomada con respecto al problema. Éste objetivo es maximizado (o minimizado) sujeto a restricciones que limitan la selección de la decisión.

En la optimización se debe identificar primero la *función objetivo*, una propiedad cuantitativa del sistema a estudiar. Este objetivo puede ser, ganancias, tiempo, potencial, energía o alguna cantidad o combinación de cantidades que pueden ser representadas por un solo número. La función objetivo depende de ciertas características del sistema llamadas *variables*. La finalidad, es encontrar los valores de las variables que optimizan la función objetivo. Algunas veces las variables están sujetas a *restricciones*, esto significa que no cualquier decisión es posible. El proceso para identificar el objetivo, las variables y restricciones de un problema dado, es conocido como *modelado*. Construir un modelo apropiado es el primer paso en el proceso de optimización. Así, según el modelo del problema, se asignará un algoritmo de optimización para resolver dicho problema. Después de haber aplicado el algoritmo, se debe ser capaz de reconocer si se ha tenido éxito en la tarea de encontrar la solución, para ello existe una expresión matemática conocida como *condiciones de optimalidad*.

Matemáticamente, la optimización es la minimización o maximización de una función sujeta a posibles restricciones sobre sus variables.

Para el estudio de los diferentes métodos de optimización se usará la siguiente notación:

- el vector de variables o parámetros x.
- la función objetivo f, es un campo escalar el cual se maximizará ó minimizará.
- las restricciones c_i, campos escalares que definen ciertas ecuaciones y/o inecuaciones que el vector x debe satisfacer.

Empleando esta notación, un problema de optimización puede ser escrito, en general, como sigue:

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{min} & f(x) \\ \text{Sujeto a} & c_i(x) = 0, \quad i \in \mathcal{E} \\ & c_i(x) \geq 0, \quad i \in \mathcal{I} \end{array} \tag{2.1.1}$$

donde $f : \mathbb{R}^n \to \mathbb{R}$, $c_i : \mathbb{R}^n \to \mathbb{R}^m$, con $m < n$ y $\mathcal{E}$ e $\mathcal{I}$ son conjuntos de índices para las restricciones de igualdad y desigualdad respectivamente.

Tamaño del problema

Una medida de la complejidad de un problema de programación es su tamaño, medido en términos de el número de variables desconocidas o el número de restricciones. Se reconocen tres clases de problemas: *Problemas a pequeña escala* tienen cinco o menos variables y/o restricciones; *Problemas a mediana escala* tienen de cinco a mil variables; y los *Problemas a gran escala* tienen de mil a un millón de variables y/o restricciones. Los problemas de pequeña escala pueden ser resueltos manualmente o haciendo uso de un computador. Los problemas de mediana escala son resueltos usando un computador personal con la ayuda de códigos de programación matemática. Problemas a gran escala requieren códigos sofisticados que exploten las estructuras particulares y usualmente requieren grandes computadores.

Gran parte de la teoría básica asociada con optimización, sobre todo en la programación no lineal, está orientada a alcanzar las condiciones necesarias y suficientes satisfechas por un punto solución. Esta parte de la teoría involucra principalmente el estudio de los multiplicadores de Lagrange, incluyendo el teorema de Karush-Kuhn-Tucker y sus extensiones.

Optimización con restricciones y sin restricciones

Problemas generales como (2.1.1) son clasificados de acuerdo con la naturaleza de la función objetivo y de las restricciones (linealidad, no linealidad, convexidad), el número de variables (grande o pequeño), la suavidad de las funciones (diferenciables o no diferenciables).

- Problemas de *optimización sin restricciones*, para el cual $\mathcal{E} = \mathcal{I} = \emptyset$ en (2.1.1). Estos aparecen en muchas aplicaciones o como reformulaciones de problemas de optimización con restricciones (métodos de penalización [Rog81, GY08]).

- Problemas de *optimización con restricciones* son aquellos en los cuales las restricciones juegan un rol importante. Estas restricciones pueden ser desde simples cotas hasta desigualdades que representan complejas relaciones entre las variables.

Haciendo uso de una clasificación como la que se mencionó antes, el interés de estudio de este trabajo se enfoca en métodos para resolver grandes problemas con restricciones, donde su función objetivo sea no lineal, convexa y suave.

Convexidad

El concepto de convexidad es fundamental en optimización. El término convexo es aplicado a conjuntos y a funciones. Un conjunto $A \subseteq \mathbb{R}^n$ es un *conjunto convexo*, si para cualquier dos puntos $x, y \in A$, se tiene que $\alpha x + (1-\alpha)y \in A$, para todo $\alpha \in [0, 1]$. La función f es una *función convexa* si su dominio es un conjunto convexo y además si para cualquiera dos puntos $x, y \in A$, donde A es el dominio, la siguiente propiedad se cumple:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), \quad \text{para todo } \alpha \in [0, 1] \tag{2.1.2}$$

Se dice que f es *estrictamente convexa* si la desigualdad (2.1.2) es estricta, esto es, $x \neq y$ y $\alpha \in (0, 1)$.

Si la función objetivo, la *región factible* y el conjunto de puntos que satisfacen todas las restricciones, en un problema de optimización (2.1.1), son convexas, entonces una solución local del problema (2.1.1) es también una solución global. El termino *programación convexa*, es usado para describir un caso especial del problema general de optimización con restricciones (2.1.1) en el cual:

- La función objetivo es convexa

- Las restricciones de igualdad c_i $i \in \mathcal{E}$, son lineales

- Las restricciones de desigualdad c_i $i \in \mathcal{E}$ son cóncavas, esto es, si $-c_i$ es convexa.

Optimización global y local

Muchos algoritmos para optimización no lineal buscan únicamente una solución local, esto es, un punto en el cual la función objetivo es mas pequeña que en los otros puntos

en alguna de sus vecindades. Estos algoritmos no siempre encuentran soluciones globales, es decir el punto donde el valor de la función objetivo es el mínimo con respecto a todos los puntos factibles.

Algoritmos de optimización

Usualmente los algoritmos de optimización son iterativos. Empiezan en un punto inicial factible $x_0 \in \mathbb{R}^n$ y generan una secuencia de estimaciones por medio de alguna regla, tal que las iteraciones x_k se mueven regularmente hacia una vecindad de un minimizador local x^*, y entonces rápidamente convergen a la solución. La estrategia utilizada para moverse de una iteración a la siguiente, es lo que diferencia un algoritmo de otro. Para ello muchos algoritmos hacen uso de los valores de la función objetivo f, de las restricciones y posiblemente de sus primeras y/o segundas derivadas. Algunos algoritmos acumulan información ganada en las iteraciones previas, mientras otros usan solo la información local obtenida en la iteración actual.

Los "buenos" algoritmos deben cumplir las siguientes características:

- Robusto: Buen desempeño en variados problemas de la misma clase con buenos valores iniciales.
- Eficaz: Tiempo de ejecución corto y poco almacenamiento.
- Preciso: Capaces de encontrar una solución con precision, sin ser demasiado sensibles a errores en los datos o de redondeo cuando se ejecutan en un computador.

Convergencia

La teoría de los algoritmos iterativos puede ser dividida en tres aspectos. El primero concierne a la creación de los algoritmos. El segundo aspecto, la verificación de que un algoritmo dado generará una secuencia que converge al punto solución. Éste aspecto es llamado *Análisis de convergencia global*. El tercer aspecto hace referencia al *Análisis de convergencia local* o *Análisis de complejidad*, esto hace mención al radio con el cual la secuencia de puntos generados, converge hacia la solución.

Un problema de optimización no se puede considerar como resuelto, simplemente porque el algoritmo converge a la solución, puesto que es posible que éste requiera una cantidad excesiva de tiempo para reducir el error de convergencia a una aceptable tolerancia.

Se dará una breve descripción asociada con diferentes tipos de radios de convergencia.

Sea $\{x_k\}$ una secuencia iterativa generada por algún algoritmo que converge a x^* con alguna norma, por ejemplo

$$\lim_{k\to\infty} \|x_k - x^*\| = 0. \tag{2.1.3}$$

Si existe un número real $\alpha \geq 1$, el cual se denomina Q-orden, y una constante positiva β, denominada Q-factor, independiente de el número de iteraciones k, tal que

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^\alpha} = \beta, \tag{2.1.4}$$

se dice que $\{x_k\}$ tiene radio Q-convergente de orden α. En particular,

- Cuando $\alpha = 1$ y $\beta \in (0, 1)$, la secuencia $\{x_k\}$ es llamada convergencia Q-lineal;
- Cuando $\alpha = 1$ y $\beta = 0$, o $1 < \alpha < 2$ y $\beta > 0$, la secuencia $\{x_k\}$ es llamada convergencia Q-superlineal;
- Cuando $\alpha = 2$, se dice que $\{x_k\}$ tiene radio de convergencia Q-cuadratico.

La motivación principal para introducir el radio Q-convergencia, es comparar la velocidad de convergencia de diferentes iteraciones. No es difícil mostrar que el radio de convergencia depende de α y de β. Supongase que existen dos secuencias $\{x_k\}$ y $\{x'_k\}$ y que su Q-orden y Q-factor son respectivamente $\{\alpha, \beta\}$ y $\{\alpha', \beta'\}$. Si $\alpha > \alpha'$, entonces la secuencia con $Q - \alpha$ orden converge mas rápido que la secuencia con $Q - \alpha'$ orden. Por ejemplo, una secuencia cuadráticamente convergente, convergerá mas rápido que las secuencias lineal y superlinalmente convergente. Cuando $\alpha = \alpha'$ su radio de convergencia, Q-orden es el mismo, si $\beta < \beta'$, entonces la secuencia $\{x_k\}$ converge mas rápido que $\{x'_k\}$.
Usualmente, si el radio de convergencia de un algoritmo es Q-superlineal o Q-cuadrática, se dice que tiene un rápido radio de convergencia. Por ejemplo los métodos cuasi-Newton convergen Q-superlinealmente, y los métodos de Newton convergen Q-cuadraticamente.

Otra medida de radio de convergencia es la llamada R-convergente (radio raiz convergencia)
sea $\{x_k\} \subset \mathbb{R}^n$ una secuencia que converge a x^*. Sea

$$R_p = \begin{cases} \limsup_{k\to\infty} \|x_k - x^*\|^{1/k}, & \text{si } p = 1; \\ \limsup_{k\to\infty} \|x_k - x^*\|^{1/p^k}, & \text{si } p > 1. \end{cases} \tag{2.1.5}$$

- Si $R_1 = 0$, $\{x_k\}$ es llamado convergencia R-superlineal hacia x^*.
- Si $0 < R_1 < 1$, $\{x_k\}$ es llamado convergencia R-lineal hacia x^*.
- Si $R_1 = 1$, $\{x_k\}$ es llamado convergencia sublinealmente hacia x^*.

Similarmente, si $R_2 = 0$, $0 < R_2 < 1$, $R_2 \geq 1$, entonces $\{x_k\}$ se dice que es R-supercuadrático, R-cuadrático, R-subcuadrático convergente hacia x^*. respectivamente.

Usualmente un algoritmo con radio cuadrático o super lineal se considera bastante conveniente. Sin embargo, se debe tener en cuenta que los resultados teóricos de la convergencia y el radio de convergencia no son garantía de un buen desarrollo.

2.2. Método de descenso

El método de descenso es uno de los métodos mas simples y fundamentales de la teoría de minimización para optimización sin restricciones. Hay una estructura fundamental subyacente para todos los algoritmos descendentes que se analizan. Se comienza en un punto inicial, se determina, de acuerdo con una regla fija, una dirección de movimiento y después se sigue esta dirección hacia un mínimo (relativo) de la función objetivo. En el nuevo punto se determina una nueva dirección y se repite el proceso. Las diferencias fundamentales entre los diferentes algoritmos de descenso, radican en la regla mediante la cual se seleccionan las direcciones sucesivas del movimiento.

En cada iteración el método de búsqueda lineal calcula una *dirección de búsqueda* d_k y decide que cantidad se mueve a lo largo de ésta dirección. Una nueva iteración es dada por

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2.2.6}$$

donde el escalar positivo α_k es denominado *longitud de paso*. Muchos de los algoritmos de búsqueda lineal requieren que d_k sea una *dirección descendente*, esto es $d_k^T \nabla f_k < 0$, ya que esta propiedad garantiza que la función f pueda ser reducida a lo largo de ésta dirección. Generalmente la dirección de búsqueda tiene la forma

$$d_k = -B_k^{-1} \nabla f_k, \tag{2.2.7}$$

donde B_k es una matriz no singular y simétrica. En el método de descenso la matriz B_k es la matriz identidad, mientras que en el método de Newton es el Hessiano $\nabla^2 f_k$ y en

el método cuasi-Newton, B_k es una aproximación del Hessiano. Cuando d_k es definido por medio de (2.2.7) y B_k es definido positivo, se obtiene

$$d_k^T \nabla f_k = -\nabla f_k^T B_k^{-1} \nabla f_k < 0, \tag{2.2.8}$$

y por lo tanto d_k es una dirección de descenso.

La estrategia básica de un método de descenso puede ser como sigue:

Algoritmo 2.2.0.1 (Estrategia básica)

Paso 1 *Definir un punto inicial x_0 y la tolerancia $\epsilon > 0$.*

Paso 2 *(Criterio de parada) Si $\|\nabla f(x_k)\| \leq \epsilon$, para.*

Paso 3 *(Encontrar el paso de descenso) De acuerdo con la estrategia, encontrar d_k tal que sea un paso de descenso.*

Paso 4 *(Búsqueda lineal) Determinar el tamaño α_k tal que el valor de la función objetivo decrezca.*

Paso 5 *Establecer $x_{k+1} = x_k + \alpha_k d_k$, k:=k+1 e ir al paso 2.*

El éxito del método de búsqueda lineal depende de la efectiva elección de la dirección d_k y el *longitud de paso* α_k.

longitud de paso

En el cálculo de la *longitud de paso* α_k, se deberá elegir para que éste dé una reducción sustancial de f, pero sin gastar mucho tiempo en su elección. La selección ideal deberá ser el minimizador global de la función unidimensional $\phi(\cdot)$ definida por:

$$\phi(\alpha) = f(x_k + \alpha d_k), \qquad \alpha > 0, \tag{2.2.9}$$

en general es bastante costoso computacionalmente, identificar este valor, pues usualmente se necesitan demasiadas evaluaciones de la función objetivo f y en algunos casos de el gradiente ∇f. Por ello muchas estrategias desarrollan una búsqueda *lineal inexacta* para seleccionar α de tal manera que logre una adecuada reducción en f con un mínimo costo.

Una condición es impuesta sobre α_k, como ésta debe garantizar una reducción en f, es decir, $f(x_k + \alpha_k d_k) < f(x_k)$. Esta condición no es suficiente para garantizar la convergencia hacia el mínimo x^*, pues en algunos casos pueden generar una reducción

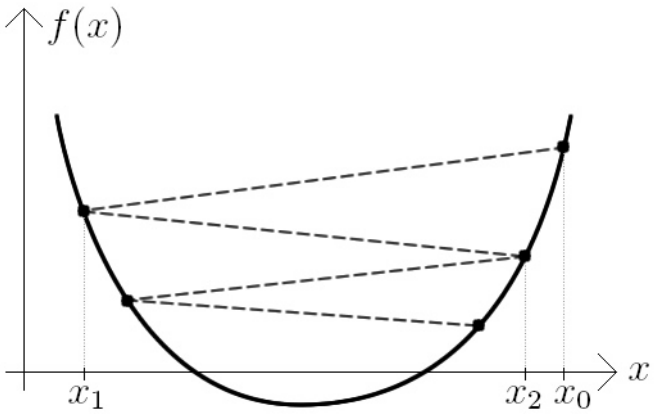


Figura 2.1: Reducción insuficiente en f

insuficiente en f la cual conduce a fallar la convergencia hacia el mínimo. Por ello es necesario forzar una condición de *decrecimiento suficiente*, la cual se estudiará a continuación.

Condiciones de Wolfe

Una búsqueda lineal inexacta condiciona que α_k, debe primero que todo, dar un decrecimiento suficiente en la función objetivo f, es decir:

$$f(x_k + \alpha d_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T d_k, \tag{2.2.10}$$

para alguna constante $c_1 \in (0, 1)$. En otras palabras, la reducción en f debe ser proporcional a la *longitud de paso* α y a la derivada direccional $\nabla f_k^T d_k$. La desigualdad (2.2.10) es denominada *Condición de Armijo*.

Cuando en la practica, c_1 es elegido para ser tan pequeño como $c_1 = 10^{-4}$, la Condición de armijo (2.2.10) no es suficiente para asegurar que el algoritmo haga un progreso razonable, en el sentido de lograr una rápida convergencia. Por ello es necesario un segundo requerimiento, llamado *Condición de curvatura*, el cual requiere que α_k satisfaga

$$\nabla f(x_k + \alpha_k d_k)^T d_k \geq c_2 \nabla f_k^T d_k, \tag{2.2.11}$$

para alguna constante $c_2 \in (c_1, 1)$, donde c_1 es la constante de (2.2.10).

La Condición de Armijo y la Condición de curvatura, son conocidas como las *Condiciones de Wolfe*.

Un α puede satisfacer las Condiciones de Wolfe sin la particularidad de ser cercano al minimizador de ϕ. Sin embargo se puede modificar la condición de curvatura para forzar que α este en al menos una amplia vecindad de un minimizador local o de un punto estacionario de ϕ. Las *Condiciones fuertes de Wolfe* requieren que α_k satisfaga

$$\begin{aligned} f(x_k + \alpha_k d_k) &\leq f(x_k) + c_1 \alpha_k \nabla f_k^T d_k \quad (2.2.12) \\ |\nabla f(x_k + \alpha_k d_k)^T d_k| &\leq c_2 |\nabla f_k^T d_k|, \quad (2.2.13) \end{aligned}$$

con $0 < c_1 < c_2 < 1$.

Es posible probar que existen pasos de longitud α que satisfacen las Condiciones de Wolfe para cada función f que sea suave y acotada inferiormente. Ver [JJ06].

Lema 2.2.0.1 *Supongase que $f : \mathbb{R}^n \to \mathbb{R}$ es continuamente diferenciable. Sea d_k una dirección de descenso en x_k y asuma que f es acotada inferiormente a lo largo del segmento de recta $\{x_k + \alpha d_k : \alpha > 0\}$. Entonces si $0 < c_1 < c_2 < 1$, existen intervalos de pasos de longitud α que cumplen las Condiciones de Wolfe y las Condiciones fuertes de Wolfe.*

2.3. Método de Newton

La idea básica del método de Newton para optimización sin restricciones, es usar iterativamente una aproximación de Taylor de grado dos $q^{(k)}$ de la función objetivo f en la iteración actual k y minimizar la aproximación $q^{(k)}$.

Sea $f(x) : \mathbb{R}^n \to \mathbb{R}$, con $f \in C^2$ y el Hessiano $\nabla^2 f(x_k)$ definido positivo. Sea $q^{(k)}$ una aproximación de Taylor de grado dos de f en la iteración k,

$$f(x_k + s) \approx q^{(k)}(s) = f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T \nabla^2 f(x_k) s, \quad (2.3.14)$$

donde $s = x - x_k$. Minimizando $q^{(k)}(s)$ se tiene

$$x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) \quad (2.3.15)$$

la cual es la *fórmula de Newton*. Sea

$$G_k = \nabla^2 f(x_k), \quad g_k = \nabla f(x_k). \quad (2.3.16)$$

Usando (2.3.15) en (2.3.16), se obtiene

$$x_{k+1} = x_k - G_k^{-1} g_k, \quad (2.3.17)$$

donde $s_k = x_{k+1} - x_k = -G_k^{-1} g_k$ es una dirección de Newton. Se comprueba que la dirección de Newton es una dirección descendente porque esta satisface $g_k^T s_k = -g_k^T G_k^{-1} g_k < 0$, si G_k es definida positiva.

Para funciones cuadráticas definidas positivas, el método de Newton puede alcanzar el mínimo en una iteración. Sin embargo para funciones no cuadráticas generales, no es seguro que el método de Newton alcance el mínimo en un número finito de iteraciones. Pero ya que la función objetivo es aproximada por una función cuadrática cerca del mínimo, entonces, si el punto inicial es cercano al mínimo, el método de Newton convergerá rápidamente. El siguiente teorema muestra la convergencia local y el radio de convergencia cuadrático del método de Newton.

Teorema 2.3.0.1 (Convergencia del método de Newton) *Sea $f \in C^2$ y x_k suficientemente cercano a la solución x^* del problema, con $g(x^*) = 0$. Si el Hessiano $G(x^*)$ es definido positivo y $G(x)$ satisface las condiciones de Lipschitz*

$$|G_{ij}(x) - G_{ij}(y)| \leq \beta \|x - y\|, \text{ para algún } \beta, \text{ para todo } i, j \tag{2.3.18}$$

donde $G_{ij}(x)$ es la (i,j)-elemento de $G(x)$, entonces para todo k, la iteración de Newton (2.3.15) está bien definida; la secuencia generada $\{x_k\}$ converge a x^* con radio cuadrático.

Note que el método de Newton es un método local. Cuando el punto inicial esta demasiado lejos de la solución, no hay seguridad de que G_k sea definido positivo y que la dirección de Newton d_k sea una dirección descendente. Por lo tanto, la convergencia no esta garantizada. Pero como se indicó antes, la búsqueda lineal es una estrategia global, se puede emplear el método de Newton con búsqueda lineal para garantizar la convergencia global. Sin embargo, hay que tener en cuenta que solo cuando la secuencia de la longitud del paso $\{\alpha_k\}$ converge a 1, el método de Newton es convergente con radio cuadrático. La iteración de Newton con búsqueda lineal es como sigue

$$d_k = -G_k^{-1} g_k \tag{2.3.19}$$
$$x_{k+1} = x_k + \alpha_k d_k \tag{2.3.20}$$

donde α_k es la longitud del paso. Las dos fórmulas anteriores corresponden al siguiente algoritmo:

Algoritmo 2.3.0.2 (Método de Newton con búsqueda lineal)

Paso 1 *Dado un $x_0 \in \mathbb{R}^n$, $\epsilon > 0$, establecer $k := 0$.*

Paso 2 *Calcular g_k. Si $\|g_k\| \leq \epsilon$, parar y retornar x_k, en otro caso pasar al paso 3*

Paso 3 *Resolver $G_k d_k = -g_k$ para d_k*

Paso 4 *Paso de búsqueda lineal: Encuentra un α_k tal que*

$$f(x_k + \alpha_k d_k) = \min_{\alpha \geq 0} f(x_k + \alpha d_k). \tag{2.3.21}$$

Paso 5 *Sea $x_{k+1} = x_k + \alpha_k d_k$, $k := k + 1$, ir al paso 2.*

El siguiente teorema prueba que el algoritmo (2.3.0.2) es globalmente convergente.

Teorema 2.3.0.2 *Sea $f : \mathbb{R}^n \to \mathbb{R}$ dos veces continuamente diferenciable sobre un conjunto abierto convexo $D \subset \mathbb{R}^n$. Asuma que para algún $x_0 \in D$ existe una constante $m > 0$ tal que $f(x)$ satisface*

$$u^T \nabla^2 f(x) u \geq m\|u\|^2, \quad \text{para todo } u \in \mathbb{R}^n, \ \ x \in L(x_0) \tag{2.3.22}$$

donde $L(x_0) = \{x | f(x) \leq f(x_0)\}$. Entonces la secuencia $\{x_k\}$ generada por el algoritmo (2.3.0.2) satisface

1. *cuando $\{x_k\}$ es una secuencia finita, entonces $g_k = 0$ para algún k;*
2. *cuando $\{x_k\}$ es una secuencia infinita, entonces $\{x_k\}$ converge a un mínimo único x^* de f.*

2.4. Métodos Cuasi-Newton

Teniendo en cuenta la hipótesis de que la evaluación y utilización de la Matriz Hessiana, en el método de Newton (sección anterior), no es práctica o su costo computacional es elevado [GY08, JJ06, SY06], se utiliza una aproximación B_k de la inversa del Hessiano, en lugar de la inversa verdadera que requiere el método de Newton. Así, los métodos cuasi-Newton son de la clase de métodos que no necesitan calcular el Hessiano, pero generan una serie de aproximaciones Hessianas y al mismo tiempo mantienen un rápido radio de convergencia. De esta manera varían los métodos, desde los que durante todo su proceso mantienen fija la aproximación de la inversa del Hessiano hasta los que en cada iteración "actualizan" la aproximación de la inversa del Hessiano.

Como en lugar de calcular el Hessiano G_k, se construye una aproximación B_k de la inversa del Hessiano. Se espera que la secuencia $\{B_k\}$ sea definida positiva, que la dirección de descenso generado por el método sea una buena dirección, en el sentido de una rápida convergencia, como el método de Newton. Entonces ¿Que condiciones

debe satisfacer $\{B_k\}$?.

Sea $f : \mathbb{R}^n \to \mathbb{R}$ dos veces continuamente diferenciable sobre un conjunto abierto $D \subset \mathbb{R}^n$. La aproximación de Taylor de grado dos de f en x_{k+1} es:

$$f(x) \approx f(x_{k+1}) + g_{k+1}^T(x - x_{k+1}) + \frac{1}{2}(x - x_{k+1})^T G_{k+1}(x - x_{k+1}) \tag{2.4.23}$$

donde $g_{k+1} \doteq \nabla f(x_{k+1})$ y $G_{k+1} \doteq \nabla^2 f(x_{k+1})$. Hallando la derivada de la aproximación de Taylor

$$g(x) \approx g_{k+1} + G_{k+1}(x - x_{k+1}) \tag{2.4.24}$$

estableciendo $x = x_k$, $s_k = x_{k+1} - x_k$ y $y_k = g_{k+1} - g_k$, se tiene:

$$G_{k+1}^{-1} y_k \approx s_k. \tag{2.4.25}$$

Note que (2.4.25) es una igualdad cuando f es una función cuadrática

$$G_{k+1}^{-1} y_k = s_k. \tag{2.4.26}$$

Así la aproximación de la inversa del Hessiano H_{k+1} debe satisfacer la relación

$$H_{k+1} y_k = s_k, \tag{2.4.27}$$

la cual es llamada ecuación cuasi-Newton o condición cuasi-Newton.

Multiplicando en (2.4.27) por s_k^T a ambos lados de la igualdad, se obtiene

$$s_k^T B_{k+1} s_k = s_k^T y_k. \tag{2.4.28}$$

esto significa que si

$$s_k^t y_k > 0, \tag{2.4.29}$$

la matriz H_{k+1} es definida positiva. Usualmente (2.4.29) es llamada condición de curvatura.

Lo anterior nos indica que la clave del método cuasi-Newton es construir B_{k+1} de tal manera que la ecuación cuasi-Newton (2.4.27) se cumpla. En general B_{k+1} es construido como una actualización de B_k, el cual es el siguiente tema a estudiar.

2.4.1. La familia de Broyden

La familia de Broyden [A.01, GY08] provee el método a utilizar en el "paso de Newton" como fórmula de actualización de la inversa de la aproximación de la Matriz Hessiana.

Como se vio antes, la matriz que aproxima la inversa del Hessiano debe cumplir la ecuación cuasi-Newton (2.4.27), esto es

$$B_{k+1}q_k = d_k. \tag{2.4.30}$$

Suponga que en cada iteración se actualiza la matriz B_{k+1} tomando la matriz B_k y adicionándole un termino de "corrección"C_k, entonces se tendría:

$$(B_k + C_k)q_k = d_k \tag{2.4.31}$$

de donde

$$C_k q_k = d_k - B_k q_k. \tag{2.4.32}$$

El método mas popular de actualización viene de la siguiente familia de matrices[1] (familia de Broyden[A.01],[GY08])

$$C^{Br}(\xi) = \frac{pp^T}{p^T q} - \frac{Bqq^T B}{q^T Bq} + \xi v v^T \tag{2.4.33}$$

donde $v = \frac{p}{p^T q} - \frac{Bq}{\tau}$, $\tau = q^T Dq$ (por simplicidad se asume que $B = B_k, d = d_k, q = q_k$).

La elección de el escalar $\xi \in [0, 1]$, el cual parametriza la familia de matrices C, da lugar a diferentes fórmulas de actualización, en particular

- Tomando $\xi = 0$ en cada iteración, obtenemos la llamada actualización de Davidson-Fletcher-Powell (DFP):

$$C^{DFP} = C^B(0) = \frac{pp^T}{p^T q} - \frac{Dqq^T D}{q^T Dq} \tag{2.4.34}$$

  la cual es históricamente el primer método cuasi-newton.

[1] $C^B r$ familia de matrices, denominada familia de Broyden

- Tomando $\xi = 1$ en cada iteración, obtenemos la llamada actualización de Broyden-Fletcher-Goldfarb-Shanno(BFGS):

$$C^{BFGS} = C^B(1) = \frac{pp^T}{p^Tq}[1 + \frac{q^TDq}{p^Tq}] - \frac{Dqp^T + pq^TD}{p^Tq} \tag{2.4.35}$$

- En general un miembro de la familia de Broyden (2.4.33) puede escribirse como una combinación convexa de los dos actualizaciones nombradas:

$$C^B(\xi) = (1 - \xi)C^{DFP} + \xi C^{BFGS}. \tag{2.4.36}$$

Finalmente, ¿como elegir la primera aproximación para B_0 en el algoritmo? Desafortunadamente no existe fórmula general que funcione bien en todos los casos. Es posible utilizar información especifica del problema, por ejemplo, estableciendo la aproximación de la inversa del hessiano por medio de diferencias finitas en el punto inicial x_0, De lo contrario, establecer B_0 como la matriz identidad.

2.5. Optimización con restricciones

Para el estudio de la minimización de problemas con restricciones, se empieza por estudiar las condiciones necesarias y suficientes que los puntos solución deben satisfacer. Estas condiciones, además de su valor intrínseco en la caracterización de soluciones definen los multiplicadores de Lagrange y cierta matriz Hessiana que, en conjunto, constituyen la base para el desarrollo y análisis de los algoritmos presentados en el resto del capítulo.

Una formulación general para este problema es

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{min} & f(x) & \\ \text{Sujeto a} & c_i(x) = 0, & i \in \mathcal{E} \\ & c_i(x) \geq 0, & i \in \mathcal{I} \end{array} \tag{2.5.37}$$

donde la funciones objetivo $f(x) : \mathbb{R}^n \to \mathbb{R}$ y las restricciones $c_i(x) : \mathbb{R}^n \to \mathbb{R}^m$, son funciones suaves y al menos una de ellas es no lineal, $\mathcal{E}$ e $\mathcal{I}$ son conjuntos de índices de restricciones de igualdad y desigualdad, respectivamente.

2.5.1. Condiciones de optimalidad

En esta subsección se da la caracterización matemática de las soluciones de (2.5.37), se discuten las condiciones de optimalidad de dos tipos; *Condiciones necesarias* que

debe ser satisfechas por algún punto solución. *Condiciones suficientes*, aquellas que, si se satisfacen en un punto x^*, garantizan que éste es punto solución.

Definición 2.5.1.1 *Un punto $x \in \mathbb{R}^n$, se dice, punto factible si y solo si x satisface las restricciones de igualdad y desigualdad. El conjunto de todos los puntos factibles es llamado* Conjunto factible.

Así podemos reescribir el problema (2.5.37) como

$$\min_{x \in X} f(x), \tag{2.5.38}$$

donde $X = \{x : c_i(x) = 0, i \in \mathcal{E}; c_i(x) \geq 0, i \in \mathcal{I}\}$ es el conjunto factible. Lo cual significa que la solución del problema de optimización con restricciones, es un punto x del Conjunto factible X tal que la función objetivo es minimizada.

La siguiente definición establece la minimización global y local.

Definición 2.5.1.2 *Si $x^* \in X$ y si*

$$f(x) \geq f(x^*), \forall x \in X, \tag{2.5.39}$$

entonces x^* se dice un minimizador global del problema (2.5.37). Si $x^* \in X$ y si

$$f(x) > f(x^*), \forall x \in X, x \neq x^*, \tag{2.5.40}$$

entonces x^* se dice un minimizador global estricto.

Definición 2.5.1.3 *Si $x^* \in X$ y si existe una vecindad $B(x^*, \delta)$ de x^* tal que:*

$$f(x) \geq f(x^*), \forall x \in X \cap B(x^*, \delta), \tag{2.5.41}$$

entonces x^* es llamado un minimizador local del problema (2.5.37), donde

$$B(x^*, \delta) = \{x : \|x - x^*\|_2 \leq \delta\}, \tag{2.5.42}$$

donde $\delta > 0$
Si $x^* \in X$ y si existe una vecindad $B(x^*, \delta)$ de x^* tal que:

$$f(x) > f(x^*), \forall x \in X \cap B(x^*, \delta), x \neq x^* \tag{2.5.43}$$

entonces x^* se dice un minimizador local estricto.

Definición 2.5.1.4 *Si $x^* \in X$ y si existe una vecindad $B(x^*, \delta)$ tal que x^* es el único minimizador local en $X \cap B(x^*, \delta)$, entonces x^*, es un minimizador aislado local.*

Dadas estas definiciones se hará la derivación matemática de la caracterización de las soluciones de (2.5.37).

CONDICIONES DE OPTIMALIDAD DE PRIMER ORDEN

Se establecen las condiciones necesarias de primer orden, ya que las direcciones factibles juegan un papel importante en la derivación de las condiciones de optimalidad, se comienza dando la definición de *direcciones factibles.*

Definición 2.5.1.5 *Sea $x^* \in X, d \neq 0, d \in \mathbb{R}^n$, si existe un $\delta > 0$ tal que*

$$x^* + td \in X, \qquad \text{para todo } t \in [0, \delta], \tag{2.5.44}$$

d es llamado una dirección factible de $x^* \in X$. El conjunto de todas las direcciones factibles de $x^* \in X$ es

$$FD(x^*, X) = \{d : x^* + td \in X, \forall t \in [0, \delta]\} \tag{2.5.45}$$

Definición 2.5.1.6 *Sea $x^* \in X$ y $d \in \mathbb{R}^n$. Si*

$$\begin{array}{ll} d^T \nabla c_i(x^*) = 0, & i \in \mathcal{E} \\ d^T \nabla c_i(x^*) \geq 0, & i \in \mathcal{I}(x^*) \end{array} \tag{2.5.46}$$

donde $I(x^*) = \{i | c_i(x^*) = 0, i \in \mathcal{I}\}$ es el conjunto de indices de las restricciones activas del problema de optimización en x^*. Entonces d se llama dirección factible linealizada en $x^* \in X$. El conjunto de todos las direcciones factibles linealizadas en $x^* \in X$ es:

$$LFD(x^*, X) = \left\{ d : \begin{array}{ll} d^T \nabla c_i(x^*) = 0, & i \in \mathcal{E}; \\ d^T \nabla c_i(x^*) \geq 0, & i \in \mathcal{I}(x^*). \end{array} \right\}. \tag{2.5.47}$$

Definición 2.5.1.7 *Sea $x^* \in X$ y $d \in \mathbb{R}^n$ (con $d \neq 0$), si exiten las secuencias d_k, $k = 1, 2, 3, ...$ y $\delta_k > 0$, $k = 1, 2, 3, ...$, tal que $x^* + \delta_k d_k \in X$, para todo d_k y $d_k \to d$, cuando $\delta_k \to 0$, entonces el límite de la dirección d_k es llamada la dirección factible secuencial de $x^* \in X$. El conjunto de todas las direcciones factibles secuenciales de $x^* \in X$ es:*

$$SFD(x^*, X) = \left\{ d : \begin{array}{ll} x^* + \delta d_k \in X, & \forall k \\ d_k \to d, \delta_k \to 0 & \end{array} \right\}. \tag{2.5.48}$$

Definiendo $x_k = x^* + \delta_k d_k$, entonces $\{x_k\}$ es una secuencia de puntos factibles que satisface

1. $x_k \neq x^*$, $\forall k$;
2. $\lim\limits_{k \to \infty} x_k = x^*$;

3. $x_k \in X$ para todo k suficientemente grande.

Si $\delta_k = \|x_k - x^*\|$, entonces se tiene:

$$d_k = \frac{x_k - x^*}{\|x_k - x^*\|} \to d,$$

esto significa que $x_k = x^* + \delta_k d_k$ es una secuencia de puntos factibles con dirección factible d.

Antes de establecer las condiciones necesarias, se define la Función de Lagrange para el problema (2.5.37)

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(x), \tag{2.5.49}$$

y se establece la siguiente notación ha usarse en el resto de la presente monografía; sea $\nabla_x \mathcal{L}(x, \lambda)$ y $\nabla^2_{xx} \mathcal{L}(x, \lambda)$ el Gradiente y el Hessiano de la Función de Lagrange con respecto, únicamente, a la variable x.

Teorema 2.5.1.1 (Condiciones necesarias de primer orden)
Suponga que x^* es una solución local de (2.5.37), que las restricciones necesarias (CQ, por su siglas en inglés)

$$SFD(x^*, X) = LFD(x^*, X) \tag{2.5.50}$$

se cumplen, entonces existen los multiplicadores de Lagrange λ_i^* tales que las siguientes condiciones se satisfacen:

$$\begin{aligned}
\nabla_x \mathcal{L}(x^*, \lambda^*) &= 0 && && (2.5.51a)\\
c_i(x^*) &= 0 && \forall i \in \mathcal{E}, && (2.5.51b)\\
c_i(x^*) &\geq 0 && \forall i \in \mathcal{I}, && (2.5.51c)\\
\lambda_i^* &\geq 0 && \forall i \in \mathcal{I}, && (2.5.51d)\\
\lambda_i^* c_i(x^*) &= 0 && \forall i \in \mathcal{I}. && (2.5.51e)
\end{aligned}$$

Estas condiciones son conocidas como las Condiciones de Karush - Kuhn - Tucker o condiciones KKT. Un punto que satisface estas condiciones es denominado *punto KKT*.

CONDICIONES DE OPTIMALIDAD DE SEGUNDO ORDEN
Las condiciones de primer orden son utilizadas para determinar si un punto x^* es un minimizador local de un problema de optimización, para ello se usa la información únicamente de las primeras derivadas de las funciones que en el problema intervienen.

Sin embargo en algunos casos (i.e. [SY06]) la información de las segundas derivadas son necesarias.

Se inicia dando las siguientes definiciones:

Definición 2.5.1.8 *Sea x^* un punto KKT del problema (2.5.37) y λ^* su correspondiente multiplicador de Lagrange, defina el conjunto de las restricciones fuertemente activas como*

$$\mathcal{I}_+(x^*) = \{i | i \in \mathcal{I}(x^*), \lambda_i^* > 0\} \tag{2.5.52}$$

Definición 2.5.1.9 *Sea x^* un punto KKT del problema (2.5.37) y λ^* su correspondiente multiplicador de Lagrange. Si existen sucesiones d_k y δ_k ($k = 1, 2, 3...$) tales que $x^* + \delta_k d_k \in X$ satisfacen*

$$\begin{array}{ll} c_i(x_k) = 0, & i \in \mathcal{E} \cup \mathcal{I}_+(x^*) \\ c_i(x_k) \geq 0, & i \in \mathcal{I}(x^*) \setminus \mathcal{I}_+(x^*) \end{array} \tag{2.5.53}$$

y $d_k \to d$ y $\delta_k \to 0$, entonces d es una sucesión de direcciones con restricciones nulas hacia x^*. El conjunto de todas las sucesiones de direcciones con restricciones nulas se define como:

$$S(x^*, \lambda^*) = \left\{ d : \begin{array}{ll} x^* + \delta d_k \in X, & \delta_k > 0,\ \delta_k \to 0,\ d_k \to d \\ c_i(x_k) = 0, & i \in \mathcal{E} \cup \mathcal{I}_+(x^*) \\ c_i(x_k) \geq 0, & i \in \mathcal{I}(x^*) \setminus \mathcal{I}_+(x^*) \end{array} \right\} \tag{2.5.54}$$

o equivalentemente

$$S(x^*, \lambda^*) = \left\{ d : \begin{array}{l} d \in SFD(x^*, X); \\ \sum_{i=1}^m \lambda_i^* c_i(x_k) = 0 \end{array} \right\}. \tag{2.5.55}$$

Similar a la dirección factible linealizada, definida en la anterior subsección, se tiene la siguiente definición

Definición 2.5.1.10 *Sea x^* un punto KKT de (2.5.37), y λ^* su correspondiente multiplicador de Lagrange, si d es una dirección factible linealizada en x^* y $\lambda_i^* d^T \nabla c_i(x^*) = 0$, para todo $i \in \mathcal{I}(x^*)$, entonces d, es un dirección con restricciones nulas linealizada. El conjunto de todas las direcciones con restricciones nulas linealizadas se define como:*

$$G(x^*, \lambda^*) = \left\{ d : \begin{array}{ll} d \neq 0 & \\ d^T \nabla c_i(x^*) = 0 & i \in \mathcal{E} \cup \mathcal{I}_+(x^*) \\ d^T \nabla c_i(x^*) \geq 0 & i \in \mathcal{I}(x^*) \setminus \mathcal{I}_+(x^*) \end{array} \right\} \tag{2.5.56}$$

equivalentemente

$$G(x^*, \lambda^*) = \left\{ d : \begin{array}{ll} d \in LFD(x^*, \lambda^*) & \\ d^T \nabla c_i(x^*) = 0 & i \in \mathcal{I}_+(x^*) \end{array} \right\} \tag{2.5.57}$$

Ahora; se establece el resultado principal de esta subsección

Teorema 2.5.1.2 (Condiciones necesarias de segundo orden) *Sea x^* un minimizador local de (2.5.37), si las restricciones necesarias (2.5.50) se cumplen, entonces*

$$d^t \nabla^2_{xx} \mathcal{L}(x^*, \lambda^*) d \geq 0, \forall d \in S(x^*, \lambda^*), \tag{2.5.58}$$

donde $\mathcal{L}(x.\lambda)$ es la función de Lagrange.

Teorema 2.5.1.3 (Condiciones suficientes de segundo orden) *Sea x^* un punto KKT de (2.5.37), si*

$$d^t \nabla^2_{xx} \mathcal{L}(x^*, \lambda^*) d > 0, \forall d \in G(x^*, \lambda^*), \tag{2.5.59}$$

entonces x^* es un minimizador local estricto

2.5.2. Función de mérito

La función de mérito es una función escalar que indica si una nueva iteración es mejor o peor que la iteración actual, en el sentido de avanzar hacia el mínimo de el problema a resolver. En la optimización sin restricciones, la función objetivo f es una elección natural para la función de mérito [JJ06, SY06]. Muchos de los métodos de optimización sin restricciones, requieren que f decrezca en cada paso. En los métodos factibles para optimización con restricciones en los cuales el punto inicial y todas la iteraciones subsecuentes satisfacen todas las restricciones del problema, la función objetivo es una función de mérito apropiada. Por otro lado, aquellos algoritmos que permiten violar las restricciones, requieren algún medio para evaluar la calidad de los pasos y de las iteraciones.

Una elección popular de la función de mérito para problemas de programación no lineal, es la *función penalty* ℓ_i definida por:

$$\phi_1(x; \mu) = f(x) + \mu \sum_{i \in \mathcal{E}} |c_i(x)| + \mu \sum_{i \in I} [c_i(x)]^-, \tag{2.5.60}$$

donde $[z]^- = max\{0, -z\}$. El escalar positivo μ es el *parámetro penalty*, el cual determina la tolerancia que se asigna para que el cumplimiento de las restricciones sea relativa a la minimización de la función objetivo.

Definición 2.5.2.1 *Una función de mérito $\phi_1(x; \mu)$ es exacta si existe un escalar positivo μ^* tal que para algún $\mu > \mu^*$, alguna solución local del problema de programación no lineal (2.5.37) es un minimizador local de $\phi_1(x; \mu)$.*

Para un estudio mas amplio de las diferentes funciones de mérito tales como ℓ_2 o lagrange aumentado de Fletcher y su elección, se recomienda [JJ06, SY06, GY08].

2.5.3. Efecto Maratos

Algunos algoritmos basados en funciones de mérito, rechazan pasos que hacen buenos progresos hacia la solución y pueden fallar hacia una rápida convergencia, fenómeno denominado *efecto Maratos*.

Las estrategias para evitar el efecto Maratos son:

1. Usar una función de mérito que no sufra el efecto Maratos. Un ejemplo es la función Lagrangiana aumentada de Fletcher.
2. Emplear una corrección de segundo orden.
3. Permitir que la función de mérito ϕ se incremente en ciertas iteraciones, esto es, usar estrategias monotonas.

Sobre esta última discutiremos en la siguiente subsección.

2.5.4. Estrategia no monótona

La deficiencia causada por el efecto Maratos, puede ser evitada aceptando ocasionalmente pasos que incrementen la función de mérito, tales pasos son llamados *pasos relaxed*. Si una suficiente reducción de la función de mérito no ha sido obtenida con cierto número de iteraciones de *paso relaxed* $\hat{i}$, entonces retornamos a la iteración anterior al *pasos relaxed* y desarrollamos una iteración normal, usando una búsqueda lineal o alguna otra técnica que force la reducción de la función de mérito.

Se describirá una instancia particular de las técnicas no monótonas llamada, *búsqueda lineal no monótona o estrategia Watchdog*. Establecemos $\hat{i} = 1$, para que permita a la función de mérito incrementarse sobre un único paso, antes de insistir sobre un decrecimiento suficiente en la función de mérito. El centro de la discusión es sobre un algoritmo de búsqueda lineal que usa una función de mérito no suave ϕ. Se asume que el parámetro penalty μ no cambia a lo largo de la iteración. Para simplificar la notación, se omite la dependencia de ϕ sobre μ, así la función de mérito se escribe $\phi(x)$ y la derivada direccional como $D(\phi(x); d_k)$.

Algoritmo 2.5.4.1 (Watchdog)

Elegir una constante $\eta \in (0, 0{,}5)$ *y un punto inicial* x_0*;*
Establecer $k \leftarrow 0$*,* $S \leftarrow 0$*;*
repetir *hasta satisfacer el test*

Calcular el paso d_k*;*
Establecer $x_{k+1} \leftarrow x_k + d_k$*;*
si $\phi(x_{k+1}) \leq \phi(x_k) + \eta D(\phi(x_k); d_k)$
$k \leftarrow k+1$, $S \leftarrow S \cup k$*;*
sino
Calcular la dirección de búsqueda d_{k+1} *en* x_{k+1}*;*
Encontrar α_{k+1} *tal que*
$\phi(x_{k+2}) \leq \phi_(x_{k+1}) + \eta\alpha_{k+1} D(\phi(x_{k+1}); d_{k+1})$*;*
Establecer $x_{k+2} \leftarrow x_{k+1} + \alpha_{k+1} d_{k+1}$*;*
si $\phi(x_{k+1}) \leq \phi(x_k)$ *o* $\phi(x_{k+2}) \leq \phi(x_k) + \eta D(\phi(x_k); d_k)$
$k \leftarrow k+2$, $S \leftarrow S \cup k$;
si $\phi(x_{k+2}) > \phi(x_k)$
(retornar a s_k *y buscar a lo largo de* d_k*)*
Encontrar α_k *tal que* $\phi(x_{k+3}) \leq \phi(x_k) + \eta D(\phi(x_k); d_k)$*;*
Calcular $x_{k+3} = x_k + \alpha_k d_k$*;*
$k \leftarrow k+3$, $S \leftarrow S \cup k$*;*
sino
calcular una dirección d_k *en* x_{k+2}*;*
Encontrar α_{k+2} *tal que*
$\phi(x_{k+3}) \leq \phi(x_{k+2}) + \eta\alpha_{k+2} D(\phi(x_{k+2}); d_{k+2})$*;*
Establecer $x_{k+3} \leftarrow x_{k+2} + \alpha_{k+2} d_{k+2}$*;*
$k \leftarrow k+3$, $S \leftarrow S \cup k$*;*
fin
fin
fin

El conjunto S solo es requerido para identificar las iteraciones en las cuales una suficiente reducción de la función de mérito fue obtenida. Note que al menos un tercio de las iteraciones tiene sus índices en S. Usando este hecho, se puede mostrar que métodos de optimización con restricciones que usan la técnica Watchdog, son globalmente convergentes, ver [JJ06]. También se puede mostrar que para todo k suficientemente grande, la longitud del paso $\alpha_k = 1$ y el radio de convergencia es superlineal, ver [JJ06].

2.5.5. Eliminación de variables

Cuando se trata con problemas de optimización con restricciones es natural intentar usar las restricciones para eliminar algunas variables del problema y obtener un problema simple con pocos grados de libertad. Las técnicas de eliminación deben usarse con cuidado porque pueden alterar el problema o introducir un mal condicionamiento.

Eliminación simple usando restricciones lineales
Considere la minimización de una función no lineal sujeta a un conjunto de restricciones lineales de igualdad,

$$\begin{array}{ll} \min & f(x) \\ \text{Sujeto a} & Ax = b \end{array} \tag{2.5.61}$$

donde $A \in \mathbb{R}^{mxn}$ es una matriz con $m \leq n$. Suponga que A tiene rango completo por filas, bajo esta suposición se puede encontrar un subconjunto de m columnas de A linealmente independientes. Si reunimos estas columnas en una matriz B de tamaño m x m y definimos una permutación $P \in \mathbb{R}^{nxn}$ tal que permute las m columnas LI en las primeras m columnas de A, es decir

$$AP = [B|N], \tag{2.5.62}$$

donde N denota las $n-m$ restantes columnas de A. Se definen los subvectores $x_B \in \mathbb{R}^m$ y $x_N \in \mathbb{R}^{n-m}$:

$$\begin{pmatrix} x_B \\ x_N \end{pmatrix} = P^T x \tag{2.5.63}$$

x_B se denomina *variable básica* y B *matriz básica*. Note que $PP^T = I$, entonces la restricción $Ax = b$ se puede reescribir como

$$b = Ax = AP(P^T x) = Bx_B + Nx_N, \tag{2.5.64}$$

reacomodando esta fórmula, se deduce que la variable básica puede ser expresada como

$$x_B = B^{-1}b - B^{-1}Nx_N; \tag{2.5.65}$$

por tanto, se puede calcular un punto factible de las restricciones $Ax = b$, eligiendo un valor de x_N y estableciendo x_B de acuerdo con la fórmula (2.5.65), así, el problema (2.5.61) es equivalente al problema sin restricciones

$$\min_{x_N} h(x_N) = f\left(P\begin{pmatrix} B^{-1}b - B^{-1}Nx_N \\ x_N \end{pmatrix}\right). \tag{2.5.66}$$

La sustitución (2.5.65) es denominada *simple eliminación de variables*.

Existe una interpretación interesante de la simple eliminación de variables. Para simplificar la notación, se asumirá que la submatriz B cuyas columnas son LI, son también las primeras columnas de la matriz A, esto es, $P = I$.

A partir de (2.5.63) y de (2.5.65), se ha visto que un punto factible para las restricciones lineales (2.5.61), pueden ser escritas como

$$\begin{pmatrix} x_B \\ x_N \end{pmatrix} = x = Yb + Zx_N \tag{2.5.67}$$

donde

$$Y = \begin{pmatrix} B^{-1} \\ 0 \end{pmatrix}, Z = \begin{pmatrix} -B^{-1}N \\ I \end{pmatrix}. \tag{2.5.68}$$

Note que Z tiene n-m columnas linealmente independientes, y que satisface

$$AZ = \begin{pmatrix} B & N \end{pmatrix} \begin{pmatrix} -B^{-1}N \\ I \end{pmatrix} = -IN + N = 0.$$

Por tanto Z es una base para el espacio nulo de A, además las columnas de Y y de Z forman un conjunto LI. Note también que de (2.5.68) y (2.5.62), Yb es una solución particular de las restricciones lineales $Ax = b$. En otras palabras, la técnica de eliminación simple expresa puntos factibles como la suma de soluciones particulares de $Ax = b$ (el primer termino en (2.5.67)) mas un desplazamiento a lo largo del espacio nulo de las restricciones (segundo termino en (2.5.67)).

Estrategia general de reducción para restricciones lineales
Para generalizar (2.5.67) y (2.5.68), se eligen matrices $Y \in \mathbb{R}^{nxm}$ y $Z \in \mathbb{R}^{nx(n-m)}$ con las siguientes propiedades:

$$\begin{array}{ll} [Y|Z] \in \mathbb{R}^{nxn} & \text{es no singular,} \\ AZ = 0. & \end{array}$$

Estas propiedades indican que, como en (2.5.68), las columnas de Z son una base para el espacio nulo de A. Como A tiene rango fila completa, entonces $A[Y|Z] = [AY|0]$, por lo tanto la matriz $AY \in \mathbb{R}^{mxm}$ es no singular, y se expresa una solución de las restricciones lineales $Ax = b$ como

$$x = Yx_y + Zx_z \tag{2.5.69}$$

para algunos vectores $x_y \in \mathbb{R}^m$ y $x_z \in \mathbb{R}^{n-m}$. Sustituyendo (2.5.69) en las restricciones $Ax = b$, se obtiene

$$Ax = (AY)x_y = b; \tag{2.5.70}$$

y a partir de la no singularidad de AY, x_y puede ser escrito explícitamente como

$$x_y = (AY)^{-1}b. \tag{2.5.71}$$

Sustituyendo esta expresión en (2.5.69), se concluye que algún vector x de la forma

$$x = Y(AY)^{-1}b + Zx_z \tag{2.5.72}$$

satisface las restricciones $Ax = b$ para algún $x_z \in \mathbb{R}^{n-m}$. Por lo tanto, el problema (2.5.61) puede ser reescrito equivalentemente como un problema sin restricciones

$$\min_{x_z} f(Y(AY)^{-1}b + Zx_z). \tag{2.5.73}$$

2.6. Programación cuadrática

La programación cuadrática es uno de los algoritmos más simples de optimización no lineal con restricciones. Trata con una clase especial del problema de optimización de la forma presentada en (2.5.37) donde la función objetivo $f(x)$ es cuadrática y las restricciones $c_i(x)$ $(i \in E \cup I)$ son lineales. Este tipo de problemas surgen como subproblemas en métodos para problemas de optimización con restricciones, tales como, programación cuadrática sucesiva, métodos de lagrange aumentado y métodos de punto interior, entre otros.

2.6.1. Programación cuadrática con restricciones de igualdad

Un problema de programación cuadrática (QP, por sus siglas en inglés) con restricciones de igualdad, se puede escribir como:

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & Q(x) = g^T x + \frac{1}{2} x^T G x \\ \text{Sujeto a} & A^T x = b \end{array} \tag{2.6.74}$$

donde $g \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{nxm}$ una matriz simétrica, donde $m \leq n$) $G \in \mathbb{R}^{nxn}$. Sin perdida de generalidad se asume que A tiene rango completo por filas $(rank(A) = m)$, para que las restricciones sean consistentes. Para estudiar el caso en el cual A no tiene rango completo por columnas se recomienda la sección 16.8 de [JJ06].

Las condiciones necesarias de primer orden (2.5.1.1) para que x^* sea una solución de (2.6.74), establece que existe un vector λ^* tal que el siguiente sistema de ecuaciones se satisface

$$\begin{pmatrix} G & -A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = \begin{pmatrix} -g \\ b \end{pmatrix}. \tag{2.6.75}$$

El sistema (2.6.75), se replantea usando $x^* = x + d$, donde x, es una estimación de la solución y d el paso de descenso deseado:

$$\begin{pmatrix} G & -A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} -p \\ \lambda^* \end{pmatrix} = \begin{pmatrix} c \\ h \end{pmatrix}, \tag{2.6.76}$$

donde $h = Ax - b$, $c = g + Gx$, $p = x^* - x$.

La matriz (2.6.76) es llamada la matriz de Karush-Kuhn-Tucker (KKT), y el siguiente resultado da las condiciones bajo la cual es no singular.

Sea Z la matriz de tamaño n x $(n - m)$, cuyas columnas son una base para el espacio nulo de A, esto es, Z tiene rango completo y satisface $AZ = 0$.

Lema 2.6.1.1 *Sea A de rango completo, y asuma que la matriz Z^TGZ es definida positiva. Entonces la matriz KKT*

$$K = \begin{pmatrix} G & -A^T \\ A & 0 \end{pmatrix} \tag{2.6.77}$$

es no singular, y por lo tanto existe un único vector (x^*, λ^*) que satisface (2.6.75).

Se estableció que cuando las condiciones del Lema 2.6.1.1 se satisfacen, existen un único vector par (x^*, λ^*) que satisfacen las condiciones necesarias de primer orden para (2.6.74). De hecho las condiciones suficientes de segundo orden son también satisfechas en (x^*, λ^*), por tanto x^* es un minimizador estrictamente local de (2.6.74). Así se usa un argumento directo para mostrar que x^* es una solución global de (2.6.74).

Teorema 2.6.1.1 *Sea A de rango completo por filas y suponga que la matriz Z^TGZ es definida positiva. Entonces el vector x^* que satisface (2.6.75), es la única solución global de (2.6.74)*

Cuando la matriz de reducción Hessiana Z^TGZ es semidefinida positiva con valores propios iguales a cero, el vector x^* que satisface (2.6.76) es un minimizador local, pero no un minimizador estricto local. Si la matriz Hessiana reducida tiene valores propios negativos, entonces x^* es solo un punto estacionario, y no un minimizador local.

Existen diferentes métodos para resolver el sistema KKT (2.6.75), a través de métodos de solución directa tales como método de complemento-schur, método de los espacios nulos, método de lagrange entre otros [JJ06]. A continuación se analizará el método de los espacios nulos para resolver el sistema (2.6.76).

2.6.2. Método de los espacios nulos

El método de los espacios nulos no requiere la no singularidad de G, éste asume únicamente las condiciones del lema (2.6.1.1), sin embargo requiere conocer la matriz básica de los espacios nulos Z.

Suponga que se particiona el vector p de (2.6.76) en dos componentes

$$d = Yd_y + Zd_z, \tag{2.6.78}$$

donde $Z \in \mathbb{R}^{n\text{x}(n-m)}$ es la matriz de los espacios nulos, $Y \in \mathbb{R}^{n\text{x}m}$ es una matriz tal que $[Y|Z]$ es no singular, d_y es un m-vector y d_z es un $(n-m)$-vector. Las matrices Y y Z fueron discutidas en la sección 1.4 (Optimización con restricciones), donde se analizó que Yx_y es una solución particular de $Ax = b$, mientras que Zx_z es un desplazamiento a lo largo de estas restricciones.

Sustituyendo p en (2.6.76) y teniendo en cuenta que $AZ = 0$, se obtiene

$$(AY)d_y = -h \tag{2.6.79}$$

Como A tiene rango m y $[Y|Z]_{nxn}$ es no singular, el producto $A[Y|Z] = [AY|0]$ tiene rango m. Por lo tanto $AY \in \mathbb{R}^{m\text{x}m}$ es una matriz no singular, y d_y esta bien determinado por la ecuación (2.6.79).

Sustituyendo (2.6.78) en (2.6.76), se obtiene

$$-GYd_y - GZd_z + A^T\lambda^* \tag{2.6.80}$$

y multiplicando por Z^T se obtiene

$$(Z^TGZ)d_z = -Z^TGYd_y - Z^Tg. \tag{2.6.81}$$

Este sistema puede ser resuelto desarrollando una factorización de Cholesky de la matriz Hessiana reducida Z^TGZ para determinar d_Z. Y así poder calcular $d = Yd_y + Zd_z$. Para obtener el multiplicador de lagrange, se multiplica el primer bloque columna de (2.6.76) por Y^T para obtener el sistema lineal

$$(AY)^T\lambda^* = Y^T(g + Gp), \tag{2.6.82}$$

el cual se resuelve para λ^*.

El método de los espacios nulos puede ser bastante efectivo cuando el número de grados de libertad $n - m$ es pequeño. Su limitación principal radica en la matriz de los espacios nulos Z, pues en grandes problemas su cálculo,en términos computacionales, es demasiado costoso, ver [JJ06]. Se debe tener en cuenta que la matriz Z no es única, por lo tanto debe hacerse una buena elección de ella para que no con lleve a un mal condicionamiento.

2.7. Programación cuadrática sucesiva

La programación cuadrática sucesiva (SQP, por su sigla en inglés) es una de las técnicas más eficaces para la solución de problemas de programación no lineal restrictiva. La técnica es apropiada para pequeños o grandes problemas y es efectiva cuando el problema a resolver, tiene un número significante de restricciones no lineales.

Considere un problema de optimización con restricciones de igualdad:

$$\begin{array}{c} \underset{x\in\mathbb{R}^n}{min}\, f(x) \\ \text{sujeto a } c(x)=0 \end{array} \tag{2.7.83}$$

donde $f:\mathbb{R}^n\to\mathbb{R}$ y $c:\mathbb{R}^n\to\mathbb{R}^m$ son funciones suaves. El método SQP genera, en cada iteración, una aproximación del problema (2.7.83) en un sub problema cuadrático, y usa el minimizador de este sub problema para construir una nueva iteración x_{k+1}. La derivación mas simple del método SQP, se puede ver como una aplicación del método de Newton a las condiciones de optimalidad KKT de el problema (2.7.83).

Por (2.5.49), la función de Lagrange para el problema (2.7.83) es $\mathcal{L}(x,\lambda)=f(x)-\lambda^T c(x)$. $A(x)$ denota el jacobiano de la matriz de las restricciones, esto es:

$$A(x)=[\nabla c_1(x),\nabla c_2(x),...,\nabla c_m(x)],$$

donde c_i es la i-ésima componente del vector $c(x)$. Las condiciones de primer orden (2.5.1.1), de un problema con restricciones de igualdad (2.7.83), puede ser visto como un sistema de $n+m$ ecuaciones en las $n+m$ desconocidas x y λ:

$$F(x,\lambda)=\begin{pmatrix} f(x)-A(x)^T\lambda \\ c(x) \end{pmatrix}=0 \tag{2.7.84}$$

Una solución (x^*,λ^*) del problema (2.7.83), para el cual $A(x)$ tiene rango completo satisface (2.7.84). Una sugerencia es resolver las ecuaciones no lineales (2.7.84) usando el Método de Newton;

El jacobiano de (2.7.84) con respecto a x y a λ es:

$$F'(x,\lambda)=\begin{pmatrix} \nabla^2_{xx}\mathcal{L}(x,\lambda) & -A(x)^T \\ A(x) & 0 \end{pmatrix}=0 \tag{2.7.85}$$

El paso de Newton en la iteración (x_k,λ_k) esta dado por:

$$\begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \end{pmatrix}=\begin{pmatrix} x_k \\ \lambda_k \end{pmatrix}+\begin{pmatrix} d_k \\ d_\lambda \end{pmatrix} \tag{2.7.86}$$

donde d_k y d_λ resuelven el sistema Newton - KKT

$$\begin{pmatrix} \nabla^2_{xx}\mathcal{L}(x,\lambda) & -A(x)^T \\ A(x) & 0 \end{pmatrix} \begin{pmatrix} d_k \\ d_\lambda \end{pmatrix} = \begin{pmatrix} -f(x) + A(x)^T\lambda \\ -c(x) \end{pmatrix} \tag{2.7.87}$$

Esta iteración de Newton esta bien definida, cuando la matriz KKT (2.7.87), es no singular. Esta matriz es no singular, esto es, si se cumplen las condiciones del lema (2.6.1.1).

Existe otra alternativa de ver la iteración (2.7.87). Suponga que en la iteración (x_k, λ_k), el problema (2.7.83) se aproxima al modelo cuadrático:

$$\begin{gathered} \min_d f_k(x) + \nabla f_k^T d + \tfrac{1}{2} d^T \nabla^2_{xx}\mathcal{L}_k d \\ \text{sujeto a } A_k d + c_k = 0 \end{gathered} \tag{2.7.88}$$

Si el lema 2.6.1.1 se cumple, el problema (2.7.88) tiene una única solución (d_k, l_k) que satisface:

$$\nabla^2_{xx}\mathcal{L}_k d_k + \nabla f(x) - A_k^T l_k = 0 \tag{2.7.89}$$

$$A_k d_k + c_k = 0. \tag{2.7.90}$$

Los vectores d_k y l_k pueden ser identificados como las soluciones de la ecuación de Newton (2.7.87). Si se resta $A_k^T\lambda_k$ de ambos lados de la primera ecuación en (2.7.87), se obtiene:

$$\begin{pmatrix} \nabla^2_{xx}L_x & -A_k^T \\ A_k & 0 \end{pmatrix} \begin{pmatrix} d_k \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} -\nabla f_k \\ -c_k \end{pmatrix}. \tag{2.7.91}$$

De ahí, por la no singularidad de la matriz de coeficientes, se tiene que $\lambda_{k+1} = l_k$ y que d_k es solución de (2.7.87), (2.7.89) y (2.7.90).

Así la nueva iteración (x_{k+1}, λ_{k+1}), puede ser definida como la solución de un problema cuadrático o como la iteración generada por el método de Newton aplicado a las condiciones de optimalidad del problema.

2.7.1. Métodos de reducción hessiana

Los métodos de reduccion Hessiana son un caso especial de la programación cuadrática sucesiva (SQP), estos métodos, al igual que SQP, generan en la iteración x_k una dirección de búsqueda resolviendo un problema cuadrático.

La idea fundamental del método de reducción Hessiana , es que únicamente la parte de la matriz Hessiana de la función de lagrange es usada para que el método requiera menos almacenamientos y costo computacional en cada iteración.

Para hacer la derivación del método de reducción Hessiana, considerese la ecuación (2.7.91). Cuando se estudió el método de los espacios nulos, se definieron las matrices Y_k y Z_k, las cuales son el espacio rango de A_k^T y el espacio nulo de A_k, respectivamente. En dicho método se estudia la siguiente ecuación

$$d_k = Y_k d_y + Z_k d_z, \tag{2.7.92}$$

sustituyendo en (2.7.91) se obtiene un sistema lineal independiente de λ

$$\begin{aligned} (A_k Y_k) d_y &= -c_k, & (2.7.93)\\ (Z_k^T \nabla_{xx}^2 L_k Z_k) d_z &= -Z_K^T \nabla_{xx}^2 L_k Y_k d_y - Z_k^T \nabla f_k. & (2.7.94) \end{aligned}$$

Desde el primer bloque de ecuaciones de (2.7.91) el multiplicador de lagrange λ_{k+1} se obtiene resolviendo

$$(A_k Y_k)\lambda_{k+1} = Y_k^T(\nabla f_k + \nabla_{xx}^2 L_k d_k). \tag{2.7.95}$$

Para evitar cálculos costosos, en la resolución de las ecuaciones (2.7.94) y (2.7.95), el método de reducción Hessiana hace una aproximación del hessiano $\nabla_{xx}^2 \mathcal{L}_k$ y del término cruzado $Z_k^T \nabla_{xx}^2 \mathcal{L}_k Z_k$.

Para ampliar las diferentes sugerencias de las aproximaciones, se recomienda leer, [Kla87, JJ06].

En el siguiente capítulo se estudiará en detalle una variación del método de reducción Hessiana.

Capítulo 3

Un Método de reducción hessiana para optimización con restricciones de dimensión grande

En este capitulo se estudia el algoritmo de reducción Hessiana propuesto por Lorenz T Biegler, Jorge Nocedal y Claudia Schmid, publicado en SIAM Journal on Optimization en 1996, [TJC].

Este método de reducción Hessiana está diseñado para resolver problemas de optimización no lineales,de grandes dimensiones, con restricciones de igualdad y pocos grados de libertad. El algoritmo incorpora un *vector de corrección* que aproxima el termino cruzado $Z^T W Y p_y$. Además establece las condiciones bajo las cuales se produce una convergencia local y superlineal.

3.1. Introducción

Considere un problema de optimización no lineal

$$\min_{x \in \mathbb{R}^n} f(x) \tag{3.1.1}$$

$$\text{Sujeto a } c(x) = 0 \tag{3.1.2}$$

donde $f : \mathbb{R}^n \to \mathbb{R}$ y $c : \mathbb{R}^n \to \mathbb{R}^m$ son funciones suaves. Este algoritmo es una variación del método de programación cuadrática sucesiva y es desarrollado para ser eficiente en el caso particular cuando el número de variables n es grande. Se asume que existen las primeras derivadas de las funciones f y c.

Aplicando el método de programación cuadrática sucesiva para resolver (3.1.1), se genera en cada iteración k, una dirección de búsqueda d_k a partir de la solución del problema

$$\min_{d\in\mathbb{R}^n} \quad g(x_k)^T d + \tfrac{1}{2} d^t W(x_k) d \tag{3.1.3}$$

$$\text{sujeto a} \quad c(x_k) + A(x_k)^t d = 0 \tag{3.1.4}$$

donde g es el gradiente de la función objetivo f, W es el Hessiano de la función de Lagrange $L(x,\lambda) = f(x) + \lambda^T c(x)$ y $A(x) = [\nabla c_1(x), ..., \nabla c_m(x)]$ es la matriz, de los gradientes de las restricciones, de tamaño n x m.

Una nueva iteración de del algoritmo es calculada como:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{3.1.5}$$

donde α_k es un *paso de longitud* elegido para reducir el valor de la función de mérito. En éste caso se utiliza la función de mérito ℓ_1 definida por

$$\phi_\mu(x) = f(x) + \mu \|c(x)\|_1, \tag{3.1.6}$$

donde μ es un *parámetro de penalización.*

La solución del problema cuadrático (3.1.3) puede ser escrito de una forma simple si se eligen bases convenientes de $\mathbb{R}^n$ para representar la dirección de búsqueda d_k. Para ello, se establece una matriz no singular de dimensión n:

$$[Y_k | Z_k], \tag{3.1.7}$$

donde $Y_k \in \mathbb{R}^{n\text{x}m}$ y $Z_k \in \mathbb{R}^{n\text{x}(n-m)}$, y se asume que

$$A_k^T Z_k = 0. \tag{3.1.8}$$

Así Z_k es una base para el espacio tangente de las restricciones. Se puede expresar d_k, la solución de (3.1.3), como

$$d_k = Y_k p_y + Z_k p_z, \tag{3.1.9}$$

para un par de vectores $p_y \in \mathbb{R}^m$ y $p_z \in \mathbb{R}^{(n-m)}$. Debido a (3.1.8) las restricciones lineales (3.1.4) se pueden reescribir como

$$c_k + A_k^T Y_k p_y = 0. \tag{3.1.10}$$

Si se asume que A_k tiene rango completo por columnas, entonces la no singularidad de $[Y_k|Z_k]$ y la ecuación (3.1.8) implican que la matriz $A_k^T Y_k$ es no singular, y así p_y es determinado a partir de (3.1.10)

$$p_y = -[A_k^T Y_k]^{-1} c_k. \tag{3.1.11}$$

Sustituyendo lo anterior en (3.1.9), se obtiene

$$d_k = -Y_k [A_k^T Y_k]^{-1} c_k + Z_k p_z. \tag{3.1.12}$$

Note que

$$Y_k [A_k^T Y_k]^{-1} \tag{3.1.13}$$

es la inversa derecha de A_k^T y que el primer término de (3.1.12) es una solución particular de las restricciones lineales (3.1.4).

Luego el tamaño del subproblema SQP se reduce a un problema que puede ser expresado exclusivamente en términos de las variables p_z, así sustituyendo (3.1.9) en (3.1.3), considerando $Y_k p_y$ como constante e ignorando los términos constantes, se obtiene un problema cuadrático sin restricciones

$$\underset{p_z \in \mathbb{R}^{n-m}}{min} (Z_k^T g_k + Z_k^T W_k Y_k p_y)^T p_z + \frac{1}{2} p_z^T (Z_k^T W_k Z_k) p_z. \tag{3.1.14}$$

Si se asume que $Z_k^T W_k Z_k$ es definida positiva, la solución de (3.1.14), aplicando el paso de Newton, es

$$p_z = -(Z_k^T W_k Z_k)^{-1} [Z_k^T g_k + Z_k^T W_k Y_k p_y]. \tag{3.1.15}$$

Esto determina la dirección de búsqueda d_k.

En el caso particular donde el número de variables n es grande, pero $n-m$ es pequeño, es práctico aproximar $Z_k^T W_k Z_k$ usando el método métrica variable [C.91], ya que calcularla resulta bastante costoso computacionalmente, cuando m es grande. Por esta razón algunos autores ignoran el *termino cruzado* $Z_k^T W_k Z_k p_y$ en (3.1.15) y calculan solo una aproximación del hessiano reducido $(Z_k^T W_k Z_k)$ [TJC]. Esta sugerencia es bastante adecuada cuando las matrices básicas Y_k y Z_k en (3.1.7) se eligen para ser ortonormales. Sin embargo para grandes problemas, calcular las bases ortogonales es poco eficiente, por ello es mejor obtener Y_k y Z_k usando el método simple eliminación de variables; desafortunadamente en este caso, ignorando el termino cruzado $Z_k^T W_k Z_k$ conduce al algoritmo a ser ineficiente [HJ88] . Por lo tanto se sugiere aproximar el termino cruzado a un vector w_k,

$$[Z_k W_k Y_k] p_y \approx w_k, \tag{3.1.16}$$

así la ecuación (3.1.15) queda

$$p_z = -(Z_k^T W_k Z_k)^{-1}[Z_k^T g_k + \zeta_k w_k], \tag{3.1.17}$$

donde $0 < \zeta_k \leq 1$ es un *factor de salto* que será discutido posteriormente.

Se consideran dos opciones para calcular w_k; la primera aproxima la matriz $[Z_k W_k Y_k]$ usando la actualización de Broyden. La segunda genera w_k usando el método de diferencias finitas.

Para describir la primera estrategia considere el método cuasi-Newton en el cual la matriz rectangular $Z_k^T W_k$ es aproximada a una matriz S_k, usando el método de Broyden. Y se obtiene el vector w_k multiplicando S_k por $Y_k p_y$

$$w_k = S_k Y_k p_y. \tag{3.1.18}$$

Para actualizar S_k; como $W_{k+1} = \nabla^2_{xx} L(x_{k+1}, \lambda_{k+1})$, usando la aproximación de la secante se obtiene

$$W_{k+1}(x_{k+1} - x_k) \approx \nabla_x L(x_k, \lambda_k) - \nabla_x L(x_{k+1}, \lambda_{k+1}), \tag{3.1.19}$$

cuando x_{k+1} es cercano a x_k. Multiplicando por Z_k^T a ambos lados de la ecuación se obtiene la siguiente relación

$$S_{k+1}(x_{k+1} - x_k) = Z_k^T[\nabla_x L(x_k, \lambda_k) - \nabla_x L(x_{k+1}, \lambda_{k+1})]. \tag{3.1.20}$$

Note que en (3.1.19) la multiplicación a ambos lados de la igualdad se realiza con Z_k^T y no con Z_{k+1}^T, obteniéndose una inconsistencia de índices, pero haciendo la multiplicación con este factor se logra un algoritmo con todas las propiedades que se desean, esta estrategia es usual en estudios de métodos SQP [TJC]. Además al usar Z_k en lugar de Z_{k+1}, permite actualizar S_{k+1} y B_{k+1} antes de crear Z_{k+1} en la nueva iteración.

Para aproximar la matriz Hessiana reducida $Z_k^T W_k Z_k$; utilizando (3.1.5) y (3.1.9) en (3.1.20), se obtiene

$$[S_{k+1} Z_k]\alpha_k p_z = -\alpha_k S_{k+1}(Y_k p_y) + Z_k^T[\nabla_x L(x_{k+1}, \lambda_{k+1}) - \nabla_x L(x_k, \lambda_{k+1})]. \tag{3.1.21}$$

Como S_{k+1} es una aproximación de $Z_k^T W_k$, esto sugiere la siguiente ecuación secante para B_{k+1}, la aproximación cuasi-Newton para la matriz Hessiana reducido $Z_k^T W_k Z_k$

$$B_{k+1} s_k = y_k, \tag{3.1.22}$$

donde $s_k = \alpha_k p_z$ y

$$y_k = Z_k^T[\nabla_x L(x_{k+1}, \lambda_{k+1}) - \nabla_x L(x_k, \lambda_{k+1})] - \bar{w_k}, \tag{3.1.23}$$

con

$$\bar{w_k} = \alpha_k S_{k+1}(Y_k p_y). \tag{3.1.24}$$

Luego B_k se actualiza por medio de la fórmula BFGS

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}, \tag{3.1.25}$$

donde $s_k^T y_k$ es suficientemente positivo.

El multiplicador de lagrange λ_k necesaria en la definición (3.1.23) de y_k está definida por

$$\lambda_k = -[Y_k^T A_k]^{-1} Y_k^T g_k. \tag{3.1.26}$$

Antes de continuar se dará un pequeño bosquejo del algoritmo SQP que se estudia en este trabajo. En las secciones posteriores se discutirán todos los aspectos en detalle.

Algoritmo

1. Se eligen constante $\eta \in (0, \frac{1}{2})$ y τ, τ' con $0 < \tau < \tau' < 1$. Establecer $k := 1$ y elegir un punto inicial x_1 una matriz inicial B_1 de tamaño $n-m$ x $n-m$ simétrica y definida positiva.

2. Evaluar f_k, g_k, c_k, A_k y calcular Y_k y Z_k.

$$(A_k^T Y_k) p_y = -c_k \tag{3.1.27}$$

3. Calcular la aproximación $w_k = (Z_k W_k Y_k) p_y$

4. Elegir el parámetro de salto $\zeta_k \in (0, 1]$ y calcular p_z desde

$$B_k p_z = -[Z_k^T g_k + \zeta_k w_k]. \tag{3.1.28}$$

   Defina la dirección de búsqueda desde

$$d_k = Y_k p_y + Z_k p_z \tag{3.1.29}$$

5. Sea $\alpha_k = 1$, y elija un escalar μ_k para la función de mérito (3.1.6).

6. Se comprueba la condición de búsqueda lineal

$$\phi_{\mu_k}(x_k + \alpha_k d_k) \leq \phi_{\mu_k}(x_k) + \eta\alpha_k D\phi_{\mu_k}(x_k : d_k), \tag{3.1.30}$$

donde $D\phi_{\mu_k}(x_k : d_k)$ es la derivada direccional de la función de mérito ϕ en d_k.

7. Si la condición (3.1.30) no se cumple, se elige un nuevo $\alpha_k \in [\tau\alpha_k, \tau'\alpha_k]$ y regresar al paso 7; en otro caso establecer

$$x_{k+1} = x_k + \alpha_k d_k \tag{3.1.31}$$

8. Calcular f_{k+1}, g_{k+1}, c_{k+1}, A_{k+1} y calcular Y_{k+1} y Z_{k+1}.

9. Calcular el multiplicador de lagrange definido como

$$\lambda_{k+1} = -[Y_{k+1}^T A_{k+1}]^{-1} Y_{k+1}^T g_{k+1}. \tag{3.1.32}$$

Definir $\overline{w}_k$ (como lo veremos en la sección 1.2) y calcular

$$s_k = \alpha_k p_z \tag{3.1.33}$$

y

$$y_k = Z_k^T[\nabla_x L(x_{k+1}, \lambda_{k+1}) - \nabla_x L(x_k, \lambda_k)] - \overline{w}_{k+1}. \tag{3.1.34}$$

Si el criterio de actualización (será discutido posteriormente) se satisface, calcular B_{k+1} por la fórmula BFGS; en caso contrario $B_{k+1} = B_k$.

10. Establecer $k := k + 1$ e ir al paso 3.

3.2. Matrices básicas

A continuación se analizará la elección de las matrices básicas Y_k y Z_k. Como Z_k representa el espacio nulo de A_k^T, y $[Y_k|Z_k]$ es no singular, la elección de Y_k y Z_k es arbitraria. Sin embargo buscando la estabilidad numérica del algoritmo es deseable definir Y_k y Z_k para ser ortonormales, es decir

$$Z(x)^T Z(x) = I_{n-m}$$

$$Y(x)^T Y(x) = I_m$$

$$Y(x)^T Z(x) = 0.$$

Una manera de obtener estas matrices es a partir de la factorización QR de A. Sin embargo para grandes problemas, calcular esta factorización es poco eficiente. Por ello se considera otro método para elegir las bases Y y Z. Por ejemplo, si se redefine x

de tal manera que sus primeras m componentes son *variables dependientes* o *variables básicas* y las $n-m$ *variables de control* o *variables no básicas*. Y así inducir la partición

$$A(x)^T = [C(x)N(x)], \tag{3.2.35}$$

donde $C(x)$ debe ser no singular. Se define $Z(x)$ y $Y(x)$ como

$$Z(x) = \begin{pmatrix} -C(x)^{-1}N(x) \\ I \end{pmatrix}; \quad Y(x) = \begin{pmatrix} I \\ 0 \end{pmatrix}. \tag{3.2.36}$$

3.3. Calculando w_k y $\overline{w}_k$

En esta sección se analizará el cálculo de los vectores de aproximación w_k y $\overline{w}_k$, los cuales se utilizan en la definición de la dirección de búsqueda p_z y en la actualización de B_k. También se discutirá cuándo no realizar la actualización BFGS de la aproximación de reducción hessiana y la selección de los parámetros ζ_k y μ_k.

Para calcular la aproximación del termino cruzado $(Z^TWY)p_y$, se proponen dos métodos; primero, se considera la aproximación de $Z_k^TW_k$ a través del método de diferencias finitas a lo largo de la dirección Y_kp_y. Esta opción requiere una evaluación adicional del gradiente reducido en cada iteración, lo cual origina buen paso de descenso. El segundo, mas económico sugiere definir w_k y $\overline{w}_k$ en términos de la aproximación de Broyden para $Z_k^TW_k$, esta no requiere evaluaciones adicionales de la función gradiente. El algoritmo RCH usualmente utiliza la segunda propuesta, pero en algunos casos, como se verá mas adelante, es necesario usar la primera opción.

3.3.1. Calculando w_k y $\overline{w}_k$ usando el método de diferencias finitas

Primero se calcula p_y en x_k a través de la ecuación (3.1.27). Luego se calcula el gradiente de la función de Lagrange en $x_k + Y_kpy$ y se define

$$w_k = Z_k^T[\nabla L(x_k + Y_kp_y, \lambda_k) - \nabla L(x_k, \lambda_k)]. \tag{3.3.37}$$

Después; en una nueva iteración x_{k+1}, se define

$$\overline{w}_k = Z_k^T[\nabla L(x_k + \alpha_kY_kp_y, \lambda_{k+1}) - \nabla L(x_k, \lambda_{k+1})], \tag{3.3.38}$$

la cual requiere una nueva evaluación del gradiente cuando $\alpha_k \neq 1$.

3.3.2. Usando el Método de Broyden para calcular w_k y $\overline{w}_k$

La matriz rectangular $Z_k^T W_k$ se aproxima a una matriz S_k actualizada por medio del método de Broyden, y se define w_k y $\overline{w}_k$ multiplicando esta matriz por $Y_k p_y$ o un múltiplo de este vector. Como se vió antes, se impone una ecuación secante (3.1.20) sobre esta aproximación de Broyden, la cual por tanto puede ser actualizada por la fórmula

$$S_{k+1} = S_k + \frac{(\overline{y}_k - S_k \overline{s}_k)\overline{s}_k^T}{\overline{s}_k^T \overline{s}_k}, \tag{3.3.39}$$

donde

$$\overline{y}_k = Z_k^T [\nabla L(x_{k+1}, \lambda_{k+1}) - \nabla L(x_k, \lambda_{k+1})] \tag{3.3.40}$$

y

$$\overline{s}_k = x_{k+1} - x_k. \tag{3.3.41}$$

Ahora se define

$$w_k = S_k Y_k p_y \tag{3.3.42}$$

$$\overline{w}_k = \alpha_k S_{k+1} Y_k p_y. \tag{3.3.43}$$

Al comienzo del algoritmo se elige una constante positiva Γ para definir

$$w_k := \begin{cases} w_k, & \text{si } \|w_k\| \leq \frac{\Gamma}{\|p_y\|^{1/2}} \|p_y\|; \\ w_k \frac{\Gamma \|p_y\|^{1/2}}{\|w_k\|}, & \text{en otro caso.} \end{cases} \tag{3.3.44}$$

Para el vector de corrección $\overline{w}_k$, se elige una secuencia de números positivos γ_k tal que $\sum_{k=1}^{\infty} \gamma_k < \infty$, y se establece

$$\overline{w}_k := \begin{cases} \overline{w}_k, & \text{si } \|\overline{w}_k\| \leq \alpha_k \|p_y\| / \gamma_k; \\ w_k \frac{\alpha_k \|p_y\|}{\gamma_k \|\overline{w}_k\|}, & \text{En otro caso.} \end{cases} \tag{3.3.45}$$

Como la sucesión de las iteraciones converge a la solución, $p_y \to 0$

3.3.3. Criterio de actualización

La actualización de Broyden se dice que está bien definida solo si la condición de curvatura $s_k^T y_k > 0$ se satisfacer [TJC, TJCD00]. En el caso de un problema con restricciones la condición de curvatura es difícil de satisfacer [JJ06]. Se propone omitir la actualización BFGS en algunos casos, y se presenta la siguiente estrategia para decidir cuando hacerlo. Antes de continuar, se define $e_k = x_k - x_*$ y $\sigma_k = max\{\|e_k\|, \|e_{k+1}\|\}$, donde x_* es una solución al problema (3.1.1) y note que σ_k converge a cero si las iteraciones convergen a x_*.

Criterio de Actualización 3.3.3.1 *Elija una constante $\gamma_{fd} > 0$ y una secuencia de números positivos $\{\gamma_k\}$ tal que $\sum_{k=1}^{\infty} \gamma_k < \infty$ (se usa la misma secuencia empleada en (3.3.45)).*

- *Si $\overline{w}_k$ es calculado a través del método de Broyden y si $s_k^T y_k > 0$ y*

$$\|p_y\| \leq \gamma_k^2 \|p_z\| \tag{3.3.46}$$

  se cumplen en cada iteración k, entonces la matriz B_k es actualizada usando la fórmula BFGS. En otro caso, establecer $B_{k+1} = B_k$.

- *Si $\overline{w}_k$ es calculado usando diferencias finitas y si $s_k^T y_k > 0$ y*

$$\|p_y\| \leq \gamma_{fd} \|p_z\| / \sigma_k^{1/2} \tag{3.3.47}$$

  se cumplen en cada iteración k, entonces la matriz B_k se actualiza por medio de la fórmula BFGS. En otro caso, se establece $B_{k+1} = B_k$.

Note que σ_k necesita conocer el vector solución x^* y por tanto no se puede calcular. Sin embargo como se analizará mas adelante, σ_k puede ser reemplazada por una cantidad que es del mismo orden que el error e_k, como por ejemplo las condiciones de optimalidad ($\|Z_k^T g_k\| + \|c_k\|$).

A continuación se analizan las propiedades de la matriz BFGS B_k, cuando el Criterio de Actualización 3.3.3.1 es utilizado. Defina

$$\cos\theta_k = \frac{s_k^T B_k s_k}{\|s_k\| \|B_k s_k\|}, \tag{3.3.48}$$

la cual, como se verá luego, es una medida de lo óptimo del paso de los espacios nulos $Z_k p_z$.

Se comienza reformulando un teorema de Byrd y Nocedal [HJ88] que trata el comportamiento del $\cos\theta_k$ cuando la matriz B_k es actualizada por la fórmula BFGS.

Teorema 3.3.3.1 *Sea la secuencia $\{B_k\}$ generada por la fórmula BFGS donde, para todo $k \geq 1$, $s_k \neq 0$ y*

$$\frac{y_k^T s_k}{s_k^T s_k} \geq m > 0 \tag{3.3.49}$$

$$\|y_k^2\| \leq M. \tag{3.3.50}$$

Entonces, existen constantes β_1, β_2, $\beta_3 > 0$ *tal que, para algún* $k \geq 1$, *las relaciones*

$$\cos\theta_j \geq \beta_1 \tag{3.3.51}$$

$$\beta_2 \leq \frac{\|B_j s_j\|}{\|s_j\|} \leq \beta_3 \tag{3.3.52}$$

se cumplen para los últimos $[\frac{1}{k}]$ *valores de* $j \in [1, k]$.

Este teorema hace referencia a las iteraciones para las cuales la actualización BFGS toma lugar, pero para el resto de iteraciones se establece que $B_{k+1} = B_k$, el teorema caracteriza todas la secuencia de matrices $\{B_k\}$.

El teorema anterior establece que si $s_k^T y_k$ es siempre suficientemente positivo, en el sentido en el que las condiciones (3.3.49) y (3.3.50) se cumplen, entonces la mitad de las últimas iteraciones en la cual la actualización de BFGS toma lugar es tal que $\cos\theta_j$ es acotado siempre por cero y $B_j s_j = O(\|s_j\|)$. A partir de lo anterior se define lo siguiente

Definición 3.3.3.1 *Se define* J *como el conjunto de las iteraciones para la cual (3.3.49) y (3.3.50) se cumplen. Se denomina* J *como el conjunto de las* buenas iteraciones *y se define* $J_k = J \cap \{1, 2, ..., k\}$

Lema 3.3.3.1 *En una vecindad de un punto solución* x_*, *y siempre y cuando la actualización BFGS tome lugar, tal como lo determina el criterio de actualización (3.3.3.1),* $s_k^T y_k$ *es suficientemente positivo en el sentido en que (3.3.49) y (3.3.50) se cumplen.*

3.3.4. Eligiendo μ_k y ζ_k

Se estudiará cómo elegir apropiadamente el *parámetro de penalización* μ_k y el parámetro de salto ζ_k para w_k. La dirección de búsqueda generada por el algoritmo I es siempre una dirección descendente para la función de mérito. Además, para las buenas iteraciones J, la dirección de búsqueda es fuertemente descendente.

Como d_k satisface las restricciones (3.1.4), Byrd y Nocedal demuestran en [HJ88] que la derivada direccional de la función de mérito ℓ_1 en la dirección d_k es

$$D\phi_{\mu_k}(x_k, d_k) = g_k^T d_k - \mu_k \|c_k\|_1 \tag{3.3.53}$$

Como la inversa derecha de A_k^T es usada en (3.1.27) y (3.1.32), esto implica que

$$g_k^T Y_k P_Y = \lambda_k^T c_k. \tag{3.3.54}$$

Usando (3.1.29) y (3.3.54) se obtiene

$$\begin{aligned} D\phi_{\mu_k}(x_k, d_k) &= g_k^T Z_k p_Z - \mu_k \|c_k\|_1 + \lambda_k^T c_k \\ &= (Z_k^T g_k + \zeta_k w_k)^T p_z - \zeta_k w_k^T p_z - \mu_k \|c_k\|_1 + \lambda_k^T c_k. \end{aligned} \tag{3.3.55}$$

Ahora, desde (3.1.33) y (3.3.54) se tiene

$$B_k s_k = -\alpha_k (Z_k^T g_k + \zeta_k w_k). \tag{3.3.56}$$

Sustituyendo lo anterior en (3.3.48), se obtiene

$$\cos\theta_k = \frac{-(Z_k^T g_k + \zeta_k w_k)^T p_z}{\|Z_k^T g_k + \zeta_k w_k\| \|p_z\|}. \tag{3.3.57}$$

Desde la desigualdad $\lambda_k^T c_k \leq \|\lambda_k\|_\infty \|c_k\|_1$ y usando (3.3.56) en (3.3.55), para todo k se obtiene

$$D\phi_{\mu_k}(x_k, d_k) \leq -\|Z_k^T g_k + \zeta_k w_k\| \|p_z\| \cos\theta_k - \zeta_k w_k^T p_z - (\mu_k - \|\lambda_k\|_\infty)\|c_k\|. \tag{3.3.58}$$

Note también, que desde (3.3.56) y (3.1.33) se obtiene

$$\frac{\|s_k\|}{\|B_k s_k\|} = \frac{\|p_z\|}{\|Z_k^T g_k + \zeta_k w_k\|}. \tag{3.3.59}$$

Ahora, teniendo en cuenta las buenas iteraciones J; si $j \in J$, desde (3.3.59) y (3.3.52), se obtiene que

$$\frac{1}{\beta_3}\|Z_j^T g_j + \zeta_j w_j\| \leq \|p_z^{(j)}\| \leq \frac{1}{\beta_2}\|Z_j^T g_j + \zeta_j w_j\|. \tag{3.3.60}$$

Usando lo anterior y (3.3.51) en (3.3.58), se obtiene para todo $j \in J$

$$\begin{aligned} D\phi_{\mu_k}(x_k, d_k) &\leq -\tfrac{1}{\beta_3}\|Z_j^T g_j + \zeta_j w_j\|^2 \cos\theta_j - \zeta_k w_j^T p_z^{(j)} - (\mu_j - \|\lambda_j\|_\infty)\|c_j\|_1 \\ &\leq -\tfrac{\beta_1}{\beta_3}\|Z_j^T g_j\|^2 - \tfrac{2\zeta_j \cos\theta_j}{\beta_3}(g_j^T Z_j w_j) - \zeta_j w_j^T p_z^{(j)} - (\mu_j - \|\lambda_j\|_\infty)\|c_j\|_1, \end{aligned} \tag{3.3.61}$$

donde se ha evitado el término no positivo $\zeta_j^2 \cos\theta_j \|w_j\|^2/\beta_3$. Como se asume que $\beta_3 > 1$, se obtiene

$$D\phi_{\mu_k}(x_k, d_k) \leq -\frac{1}{\beta_3}\|Z_j^T g_j\|^2 + [2\zeta_j \cos\theta_j |g_j^T Z_j w_j| - \zeta_j w_j^T p_z^{(j)}] - (\mu_j - \|\lambda_j\|_\infty)\|c_j\|_1. \tag{3.3.62}$$

De lo anterior se obtiene que

$$2\zeta_j \cos\theta_j |g_j^T Z_j w_j| - \zeta_j w_j^T p_z^{(j)} \leq \rho \|c_j\|_1, \tag{3.3.63}$$

para alguna constante ρ, y si

$$\mu_j \geq \|\lambda_j\|_\infty + 2\rho, \tag{3.3.64}$$

entonces para todo $j \in J$

$$D\phi_{\mu_k}(x_j; d_j) \leq -\frac{\beta_1}{\beta_3}\|Z_j^T g_j\|^2 - \rho\|c_j\|_1. \tag{3.3.65}$$

Esto significa que si (3.3.63) y (3.3.64) se cumplen, entonces para las buenas iteraciones $j \in J$, la dirección de búsqueda d_j es una fuerte dirección de descenso para la función de mérito ℓ_1, en el sentido de que la reducción de primer orden es proporcional al error KKT.

Se elige ζ_k para que (3.3.63) se cumpla para todas las iteraciones. Para mostrar esto, note que desde (3.1.28)

$$p_z = -B_k^{-1} Z_k^T g_k - \zeta_k B_k^{-1} w_k, \tag{3.3.66}$$

así, para $j = k$, la ecuación (3.3.63) se puede reescribir

$$\zeta_k[2\cos\theta_k |g_k^T Z_k w_k| + w_k^T B_k^{-1} Z_k^T g_k + \zeta_k w_k^T B_k^{-1} w_k] \leq \rho\|c_k\|_1. \tag{3.3.67}$$

Note que esta condición es satisfecha para un valor ζ_k positivo y suficientemente pequeño. Específicamente, en el comienzo del algoritmo se elige una constante $\rho > 0$ y, en cada iteración k, se define

$$\zeta_k = min\{1, \widehat{\zeta}_k\}, \tag{3.3.68}$$

donde $\widehat{\zeta}_k$ es el valor mas grande que satisface (3.3.68) como una igualdad.

El parámetro μ_k debe satisfacer (3.3.64), y entonces se define en cada iteración del algoritmo

$$\mu_k = \begin{cases} \mu_{k-1}, & \text{si } \mu_{k-1} \geq \|\lambda_k\|_\infty + 2\rho; \\ \|\lambda_k\|_\infty, & \text{en otro caso.} \end{cases} \tag{3.3.69}$$

El *factor de salto* ζ_k y la fórmula de actualización para el *parámetro de penalización* μ_k, han sido definidos para generar un paso fuertemente descendente en las buenas iteraciones J. Y ellos aseguran que la dirección de búsqueda es también una dirección de descenso,pero no necesariamente fuertemente descendente, para las otras iteraciones $j \notin J$. Como (3.3.63) se cumple para todas las iteraciones, la elección particular de ζ_k

$$-\zeta_k w_k^T p_z \leq \rho\|c_k\|_1. \tag{3.3.70}$$

Usando lo anterior y (3.3.69) en (3.3.58)se obtiene

$$D\phi_{\mu_k}(x_j; d_j) \leq -\|Z_k^T g_K + \zeta_k w_k\|\|p_z\|\cos\theta_k - \rho_k\|c_k\|_1. \tag{3.3.71}$$

Se concluye que la derivada direccional es no positiva.

3.4. El algoritmo

Antes de dar una completa descripción del algoritmo, se analizará en que casos se aplica diferencias finitas y cuándo usar el método de Broyden para aproximar el término cruzado. La idea es considerar el tamaño relativo de p_y y p_z. El criterio de actualización (3.3.3.1) genera tres regiones; R_1 , R_2 y R_3 ilustradas en la figura **??**. El algoritmo empieza calculando w_k usando el método de Newton y calculando p_y y p_z. Si la dirección de búsqueda no está en R_1 o en R_3, nuevamente se calcula w_k usando diferencias finitas, y a partir de este nuevo valor calcular nuevamente p_z. La razón para utilizar diferencias finitas de esta manera es que en la región media el método de Broyden no es suficientemente bueno o la convergencia no es suficientemente tangencial para que el paso sea superlineal. Por lo tanto se debe recurrir a las diferencias finitas para obtener una buena estimación de w_k.

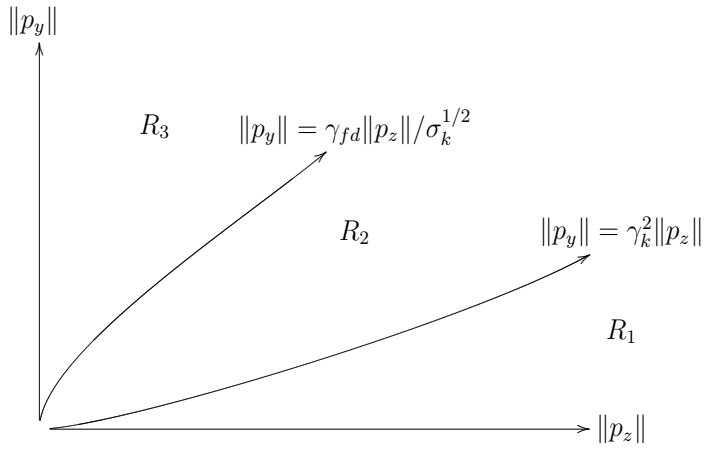


Figura 3.1: Regiones

Note que a partir del criterio (3.3.3.1), la actualización BFGS de la matriz B_k es omitida si la dirección de búsqueda esta en la región R_3. A continuación se da una completa descripción del algoritmo.

Algoritmo II

1. Elegir constantes $\eta \in (0, 1/2)$, $\rho > 0$ y τ, τ' con $0 < \tau < \tau' < 1$, y constantes positivas Γ y γ_{fd} para las condiciones (3.3.44) y (3.3.47), respectivamente. Para las condiciones (3.3.45) y (3.3.46) elegir una secuencia de números positivos $\{\gamma_k\}$. Establecer $k := 1$ y elegir un punto inicial x_1 y un valor inicial para el *parámetro de penalización* μ_1, una matriz inicial $B_1 \in \mathbb{R}^{(n-m)\text{x}(n-m)}$definida positiva y simétrica y por último una matriz inicial $Z_1 \in \mathbb{R}^{(n-m)\text{x}n}$.

2. Evaluar f_k, g_k, c_k y A_k, calcular Y_k y Z_k.

3. Establecer la variable $findiff = false$ y calcular p_y resolviendo el sistema

$$(A_k^T Y_k) p_y = -c_k \tag{3.4.72}$$

4. Calcular w_k usando el método de Broyden.

5. Elegir el *parámetro de salto* ζ_k desde las ecuaciones (3.3.67) y (3.3.68), y calcular p_z desde

$$B_k p_z = -[Z_k^T g_k + \zeta_k w_k] \tag{3.4.73}$$

6. Si (3.3.47) se satisface y (3.3.46) no se satisface, establecer $findiff = true$ y nuevamente calcular w_k desde la ecuación (3.3.37).

7. Si $findiff = true$, use este nuevo valor de w_k para elegir el "parámetro de salto" ζ_k desde las ecuaciones (3.3.67) y (3.3.68) y calcular p_z desde la ecuación (3.4.73).

8. Defina la dirección de búsqueda

$$d_k = Y_k p_y + Z_k P_z, \tag{3.4.74}$$

   donde $\alpha_k = 1$.

9. Condición de búsqueda lineal

$$\phi_{\mu k}(x_k + \alpha_k d_k) \leq \phi_{\mu k}(x_k) + \eta \alpha_k D\phi_{\mu k}(x_k; d_k). \tag{3.4.75}$$

10. Si (3.4.75) no se satisface, elegir un nuevo $\alpha_k \in [\tau\alpha_k, \tau'\alpha_k]$ y volver al paso 9, en otro caso definir

$$x_{k+1} = x_k + \alpha_k d_k. \tag{3.4.76}$$

11. Evaluar f_{k+1}, g_{k+1}, c_{k+1}, A_{k+1} y calcular $Y_{k+1}yZ_{k+1}$.

12. Calcular una estimación del multiplicador de Lagrange

$$\lambda_{k+1} = -[Y_{k+1}^T A_{k+1}]^{-1} Y_{k+1}^T g_{k+1}, \tag{3.4.77}$$

y actualizar μ_k para satisfacer (3.3.69).

13. Actualizar S_{k+1} usando las ecuaciones (3.3.39) en (3.3.41). Si $findiff = false$, calcular $\overline{w}_k$ utilizando el método de Broyden. En otro caso calcular $\overline{w}_k$ por medio de (3.3.38)).

14. Si ($s_k^T y_k \leq 0$) o si ($findiff = true$ y si (3.3.47) no se satisface) o si ($findiff = false$ y (3.3.46) no se satisface), establecer $B_{k+1} = B_k$. Si no, calcular

$$s_k = \alpha_k p_z, \tag{3.4.78}$$

$$y_k = Z_k^T [\nabla L(x_{k+1}, \lambda_{k+1}) - \nabla L(x_k, \lambda_{k+1})] - \overline{w}_k, \tag{3.4.79}$$

y calcular B_{k+1} utilizando la fórmula BFGS.

15. Establecer $k := k + 1$, e ir al paso 3.

3.5. Implementación del Algoritmo

En este capítulo se discutirán detalles importantes de la implementación del algoritmo analizado en las secciones anteriores; el estudio para determinar cuando elegir una nueva matriz C, la estrategia para el caso cuando la matriz C es cambiada, el procedimiento para calcular el parámetro de salto ζ_k.

El punto principal de la discusión será el Algoritmo II. Inicialmente se describirán algunas modificaciones y simplificaciones para el Algoritmo II, para mejorar su eficiencia en la práctica.

La fórmula (3.4.72) para calcular el multiplicador de Lagrange, dada la construcción (3.2.36) de Y_k, toma la siguiente forma

$$\lambda_k = -C_k^{-T} g_k^c, \tag{3.5.80}$$

donde g^c denota las primeras m componentes del vector g. Usando las bases Y y Z se puede simplificar la fórmula de actualización cuasi-Newton [TJCD00], la cual hace uso

de los vectores

$$y_k = Z_k^T[\nabla L(x_{k+1}, \lambda_{k+1}) - \nabla L(x_k, \lambda_{k+1})] - \bar{w}_k \quad (3.5.81)$$

$$\bar{y}_k = Z_k^T[\nabla L(x_{k+1}, \lambda_{k+1}) - \nabla L(x_k, \lambda_{k+1})]; \quad (3.5.82)$$

como lo analizó Orozco [TJCD00], las ecuaciones $\nabla L(x, \lambda) = g(x) + A(x)\lambda$, $A(x)^T Z(x) = 0$, (3.2.36) y (3.5.80) implican que para cualquier par de puntos $\bar{x}$, $\tilde{x}$ para los cuales $C(\bar{x})$ y $C(\tilde{x})$ son no singular,

$$Z(\tilde{x})^T \nabla L(\bar{x}, \lambda(\bar{x})) = Z(\bar{x})^T g(\bar{x}). \quad (3.5.83)$$

Esto y la ecuación $Z_k^T A_k = 0$ permite reescribir (3.5.81) y (3.5.82) como

$$y_k = Z_{k+1}^T g_{k+1} - Z_k^T g_k - \bar{w}_k, \qquad \bar{y}_k = Z_{k+1}^T g_{k+1} - Z_k^T g_k. \quad (3.5.84)$$

La siguiente observación corresponde al cálculo del término de corrección $\bar{w}_k$ usando diferencias finitas. Después de que una nueva iteración x_{k+1} ha sido calculada, se define

$$\bar{w}_k = Z_k^T[\nabla L(x_k + \alpha_k Y_k p_y, \lambda_{k+1}) - \nabla L(x_k, \lambda_{k+1})], \quad (3.5.85)$$

donde α_k es el *paso de longitud* usado para asegurar una suficiente reducción en la función de mérito. Recuerde que esta fórmula requiere una evaluación adicional del gradiente de Lagrange si $\alpha_k \neq 1$. Para evitar este costo, se redefine la ecuación (3.5.85) como

$$\bar{w}_k = \alpha_k Z_k^T[\nabla L(x_k + Y_k p_y, \lambda_{k+1}) - \nabla L(x_k, \lambda_{k+1})]. \quad (3.5.86)$$

Las expresiones (3.5.85) y (3.5.86) no son equivalentes, pero es posible mostrar que en una vecindad del punto solución, ellas tienen muchos efectos similares sobre el algoritmo [TJCD00].
Se hace énfasis en el cálculo de la variable p_z, la cual según (3.4.73), es

$$p_z = -B_k^{-1}[Z_k^T g_k + \zeta_k w_k], \quad (3.5.87)$$

donde "parámetro de salto" $0 < \zeta_k \leq 1$ debe asegurar que la dirección de búsqueda d_k sea siempre una dirección de descenso para la función de mérito. En el análisis de la convergencia superlineal para el algoritmo, realizado por Biegler y Nocedal [TJC], ζ_k toma el valor de 1 para que (3.5.87) se reduzca a (3.1.15) asintóticamente.

3.5.1. Selección de las bases

El algoritmo requiere la partición de las variables en variables dependientes e independientes. Esto determina la elección de la matriz básica C_k y por tanto la definición de

Z_k y Y_k.

Una vez la matriz básica ha sido determinada, se intenta mantener la misma elección de las variables básicas durante todas las iteraciones siguientes, pues la fórmula cuasi-Newton que actualiza B_k y S_k requiere que Z cambie suavemente. Para ello es necesario monitorear en cada iteración la matriz C_k, lo cual con lleva a un exceso de cálculos cuando k es demasiado grande. Por esta razón Biegler y Nocedal [TJCD00], proponen usar en cada iteración el siguiente criterio

Dado

$$\beta_k = \underset{i,j}{min}\{|(C_k^{-1}N_k)_{i,j}|\}, \tag{3.5.88}$$

el algoritmo necesita redefinir la matriz básica C_k si

$$\beta_k > 10\beta_{k-1} \tag{3.5.89}$$

o si

$$\alpha_{k-1} < 10^{-3} \quad \text{y} \quad \beta_k > \beta_{k-1} \tag{3.5.90}$$

Estas condiciones intentan determinar si la inversa de C_k está creciendo rápidamente o si una mala elección está produciendo una búsqueda de descenso excesivamente grande que obliga a generar un pequeño *paso de longitud* α_k.

3.5.2. Ajustando las matrices Cuasi-Newton

Cuando en una iteración ha sido necesario construir una nueva partición de las variables dependientes e independientes, esta nueva partición afecta la definición 3.2.36 de las matrices básicas Z y Y, por lo cual estas cambian bruscamente de x_k a x_{k+1}. Esto tendrá un efecto perjudicial en las fórmulas de actualización cuasi-Newton las cuales dependen de la matriz Z.

Biegler y Nocedal en [TJCD00] desarrollaron la siguiente estrategia para modificar las matrices B_k y S_k antes de que la actualización B_{k+1} y S_{k+1} sea desarrollada. Recuerde que la matriz jacobiana de las restricciones es escrita como $A^T = [C|N]$. Suponga que se elige una nueva partición de las variables, de tal manera que en las primeras m posiciones, son nuevamente las variables básicas. Con esta nueva elección de las variables se redefine la siguiente expresión $\bar{A}^T = [\bar{C}|\bar{N}]$, donde $\bar{A}$ es calculada desde la nueva partición. Lo anterior se resume en

$$\bar{A}^T = A^T P \tag{3.5.91}$$

donde P es una matriz de permutación. La anterior y la nueva elección de la matriz básica de los espacios nulos esta dada por

$$Z = \begin{pmatrix} -C^{-1}N \\ I \end{pmatrix} \quad \text{y} \quad \bar{Z} = \begin{pmatrix} -\bar{C}^{-1}\bar{N} \\ I \end{pmatrix} \tag{3.5.92}$$

en consecuencia $A^T Z = 0$ y $\bar{A}^T \bar{Z} = 0$. Usando 3.5.91 se obtiene

$$\bar{A}^T \bar{Z} = A^T (P\bar{Z}) = 0, \tag{3.5.93}$$

lo cual indica que las columnas de la matriz $P\bar{Z}$ también se encuentran en el espacio nulo de A^T. Por lo tanto existe una matriz cuadrada no singular R de dimension $n-m$ tal que

$$P\bar{Z} = ZR. \tag{3.5.94}$$

Cuando la selección de las variables dependientes e independientes cambia, el hessiano de la función de Lagrange también refleja este cambio. El nuevo hessiano se denota por

$$\bar{W} = P^T W P. \tag{3.5.95}$$

Como en la iteración actual la matriz de reducción Hessiana $Z^T W Z$ es aproximada a una matriz B, es necesario encontrar una nueva matriz $\bar{B}$ que aproxime $\bar{Z}^T \bar{W} \bar{Z}$. Para ello usamos la ecuación 3.5.94

$$\begin{aligned} \bar{B} \approx \bar{Z}^T \bar{W} \bar{Z} \quad &= \bar{Z}^T P^T W P \bar{Z} \\ &= R^T Z^T W Z R. \end{aligned} \tag{3.5.96}$$

Lo anterior sugiere que $\bar{B}$ sea definido como $\bar{B} = R^T B R$. No obstante calcular R resulta demasiado costoso, por lo tanto se realizan las siguientes operaciones; sea $T = [0\ \mathrm{I}]$ de tamaño $(n-m)$ x n, particionada de tal manera que $TZ = \mathrm{I}$. Usando 3.5.94 y 3.5.96 se obtiene

$$\begin{aligned} \bar{Z}^T \bar{W} \bar{Z} \quad &= R^T Z^T T^T Z^T W Z T Z R \\ &= (ZR)^T T^T (Z^T W Z) T (ZR) \\ &\quad \bar{Z}^T P^T T^T (Z^T W Z) T P \bar{Z}. \end{aligned} \tag{3.5.97}$$

De lo cual se obtiene

$$\bar{B} = \bar{Z}^T P^T T^T B T P \bar{Z} \tag{3.5.98}$$

para calcular la aproximación de la matriz de reducción hessiana usando la nueva matriz básica $\bar{Z}$.

De una manera similar se hace la derivación para

$$\bar{S} = \bar{Z}^T P^T T^T S P. \tag{3.5.99}$$

Redefinidas las matrices B_k y S_k en $\bar{B}_k$ y $\bar{S}_k$, se puede aplicar sin ningún inconveniente las fórmulas cuasi-Newton para obtener las matrices B_{k+1} y S_{k+1}.

A continuación se detallará completo el algoritmo, el cual es una modificación del Algoritmo II.

Algoritmo RCH. (Algoritmo de Reducción Hessiana con Término Cruzado).

1. Elegir constantes Γ, γ_{fd}, Δ y las secuencias γ_k y $\bar{\gamma}_k$. Establecer $k = 1$, elegir un punto inicial x_1 e inicializar el *parámetro de penalización* de la función de mérito $\mu_1 = 1$. Definir la matriz de aproximación Hessiana inicial $B_1 = I$ e inicializar la aproximación de Broyden para el término cruzado como $S_1 = [0|I]$.

2. Evaluar f_1, g_1 y c_1 y A_1 en x_1, calcular Y_1 y Z_1.

3. Establecer la variable $findiff = false$ y calcular p_y resolviendo el sistema

$$(A_k^T Y_k) p_y = -c_k \tag{3.5.100}$$

4. Calcular w_k usando el método de Broyden

$$w_k = S_k Y_k p_y, \tag{3.5.101}$$

y entonces definir

$$w_k := \begin{cases} w_k, & \text{si } \|w_k\| \leq \Gamma \|p_y\|^{1/2}; \\ w_k \frac{\Gamma \|p_y\|^{1/2}}{\|w_k\|}, & \text{en otro caso.} \end{cases} \tag{3.5.102}$$

5. Elegir el "parámetro de salto" ζ_k desde

$$\zeta_k = \begin{cases} 1, & \text{si } g_k^T Z_k B_k^{-1} w_k \geq 0; \\ min(\frac{-0{,}1 g_k^T Z_k B_k^{-1} Z_k^T g_k}{g_k^T Z_k B_k^{-1} w_k}, 1) & \text{en otro caso.} \end{cases} \tag{3.5.103}$$

y calcular p_z desde

$$B_k p_z = -[Z_k^T g_k + \zeta_k w_k]. \tag{3.5.104}$$

6. Calcular $\sigma_k = \|Z_k^T g_k\| + \|c_k\|$. Si $\tilde{\sigma} = max\{\|Z_k^T g_k\|_\infty, \|c_k\|_\infty\} \leq \Delta$ y si

$$\|p_y\| \leq \frac{\gamma_{fd\|p_z\|}}{\sigma_k^{1/2}} \tag{3.5.105}$$

y

$$\|p_y\| > \gamma_k^2 \|p_z\| \tag{3.5.106}$$

se cumplen, entonces establecer $finddiff = true$ y calcular nuevamente w_k desde la ecuación

$$w_k = Z_k^T[\nabla L(x_k + Y_k p_y, \lambda_k) - g(x_k)]. \tag{3.5.107}$$

7. Si $finddiff = true$ use este nuevo valor de w_k para elegir el "parámetro de salto" ζ_k desde la ecuación (3.5.103) y calcular nuevamente p_z desde la ecuación (3.5.104).

8. Defina la dirección de búsqueda desde

$$d_k = Y_k p_y + Z_k p_z \tag{3.5.108}$$

y establecer $\alpha_k = 1$.

9. El test de búsqueda lineal (condición de Armijo)

$$\phi_{\mu k}(x_k + \alpha_k d_k) \leq \phi_{\mu k}(x_k) + 0{,}1\alpha_k D\phi_{\mu k}(x_k; d_k), \tag{3.5.109}$$

donde

$$\phi_{\mu k}(x_k) = f_k + \mu_k \|c_k\|_1. \tag{3.5.110}$$

10. Si (3.5.109) no se satisface, elegir un nuevo α_k desde

$$\alpha_k = max\{\frac{-0{,}5D\phi_{\mu k}(x_k; d_k)\alpha_k^2}{\phi_{\mu k}(x_k + \alpha_k d_k) - \phi_{\mu k}(x_k) - \alpha_k D\phi_{\mu k}(x_k; d_k)}, 0{,}1\} \tag{3.5.111}$$

e ir al paso 9; en otro caso establecer

$$x_{k+1} = x_k + \alpha_k d_k. \tag{3.5.112}$$

11. Evaluar f_{k+1}, g_{k+1} y c_{k+1} y A_{k+1} en x_{k+1}, calcular Y_{k+1} y Z_{k+1}.

12. Calcular el multiplicador de Lagrange

$$\lambda_{k+1} = -C_{k+1}^{-T} g_{k+1}^c \tag{3.5.113}$$

y actualizar μ_k

$$\mu_{k+1} = max(1{,}001 + \|\lambda_{k+1}\|_\infty, (3\mu_k + \|\lambda_{k+1}\|_\infty)/4, 10^{-6}) \tag{3.5.114}$$

13. Actualizar S_{k+1} usando la actualización de Broyden

$$s_{k+1} = S_k + \frac{(\bar{y}_k - S_k\bar{s}_k)\bar{s}_k^T}{\bar{s}_k^T\bar{s}_k} \tag{3.5.115}$$

donde

$$\bar{y}_k = Z_{k+1}^T g_{k+1} - Z_k^T g_k, \qquad \bar{s}_k = x_{k+1} - x_k \tag{3.5.116}$$

Si $findiff = false$ calcular $\bar{w}_k$ usando el método de Broyden

$$\bar{w}_k = \alpha_k S_{k+1} Y_k p_y \tag{3.5.117}$$

y establecer

$$\bar{w}_k := \begin{cases} \bar{w}_k, & \text{si } \|\bar{w}_k\| \leq \alpha_k\|p_y\|/\gamma_k; \\ \bar{w}_k \frac{\alpha_k\|p_y\|}{\gamma_k\|\bar{w}_k\|}, & \text{en otro caso.} \end{cases} \tag{3.5.118}$$

Si $findiff = true$ calcular $\bar{w}_k$ por medio de

$$\bar{w}_k = \alpha_k Z_k^T[\nabla L(x_k + Y_k p_y, \lambda_{k+1}) - g(x_k)] \tag{3.5.119}$$

y

$$\bar{w}_k := \begin{cases} \bar{w}_k, & \text{si } \|\bar{w}_k\| \leq \alpha_k\|p_y\|/\bar{\gamma}_k; \\ \bar{w}_k \frac{\alpha_k\|p_y\|}{\bar{\gamma}_k\|\bar{w}_k\|}, & \text{en otro caso.} \end{cases} \tag{3.5.120}$$

14. Si $s_k^T y_k \leq 0$ o si (3.5.105) no se cumple, definir$B_{k+1} = B_k$. Si no calcular

$$s_k = \alpha_k p_z \tag{3.5.121}$$

$$y_k = Z_{k+1}^T g_{k+1} - Z_k^T g_k - \bar{w}_k \tag{3.5.122}$$

y actualizar B_{k+1} por medio de la fórmula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k S_k} + \frac{y_k y_k^T}{y_k^T s_k}. \tag{3.5.123}$$

15. Si $\tilde{\sigma}_k = max\|Z_k^T g_k\|_\infty, \|c_k\|_\infty \leq tol$, para; en caso contrario establecer $k = k + 1$ e ir al paso 3.

La tolerancia de convergencia usada para terminar el algoritmo es establecido por el usuario. Los valores numéricos de los otros parámetros usados están dados en la tabla 3.1. El test numérico desarrollado en [TJCD00], sugiere dichos valores para un buen desarrollo.

Parámetros	Referencia	Valor sugerido
Γ	(3.5.102)	20
γ_{fd}	(3.5.105)	10
γ_k	(3.5.118)	$0{,}1(n-m)^{0,25}k^{-1,1}$
$\bar{\gamma}_k$	(3.5.120)	$0{,}01(n-m)^{0,25}k^{-1,1}$
Δ	Paso 6	0.1

Tabla 3.1: Constantes sugeridas

3.6. Convergencia superlineal

Sin los términos de corrección w_k y $\overline{w}_k$ y con un criterio de actualización apropiado, Nocedal y Overton (1985) probaron que el Algoritmo RCH es 2-pasos Q-superlinealmente [1] convergente, asumiendo que Y_k y Z_k son bases ortogonales y que una buena matriz inicial B_1 es usada. El anterior resultado fue ampliado por Xie (1991) para bases mas generales y para una matriz inicial $B_1 > 0$. En el articulo [TJC] Biegler y Nocedal demuestran que incorporando el término de corrección usado en el Algoritmo RCH, éste tiene un radio de convergencia 1-paso Q-superlineal [2]. Este resultado es posible gracias al Criterio de Actualización I y por uso de la aproximación de las diferencias finitas, las cuales le permiten a la actualización BFGS ocurrir con mayor frecuencia.

Para estabilizar la convergencia superlineal, se necesita asegurar que el *paso de longitud* $\alpha_k = 1$, cuando k es suficientemente grande. En este trabajo se usa la función de mérito ℓ_1, no diferenciable, la cual puede rechazar pasos α_k unitarios aun cercanos a la solución, esto es denominado efecto Maratos y requiere que el algoritmo sea modificado para permitir *pasos de longitud* unitarios y así lograr un rápido radio de convergencia. En la Sección 2.5.3 se hace un estudio sobre este efecto y se detalla el algoritmo de la técnica que se utiliza para sortear el inconveniente causado por el efecto Maratos, esta técnica es llamada *Watchdog* y el algoritmo mencionado se utilizó en el programa desarrollado en este trabajo. Para estudiar en detalle la convergencia 1-paso Q-superlineal se recomienda los trabajos de Biegler, Nocedal y Byrd [TJC, HJ88].

[1] Se dice que un algoritmo converge dos pasos en el sentido de la siguiente definición $\lim_{k\to\infty} \frac{\|x_{k+p}-x^*\|}{\|x_k-x^*\|} = 0$, con $p = 2$.

[2] Se dice que un algoritmo converge a un paso cuando en la ecuación de la nota anterior p es igual a uno.

Capítulo 4

Experimentos

Se efectuó un estudio experimental del **Algoritmo RCH** descrito en el capítulo anterior. Los valores de constantes de decisión muy pequeños son aproximados a un parámetro "Tol", el cual es definido por el usuario. En este trabajo se realizaron los experimentos con variados valores del parámetro "Tol" esperando obtener los "mejores" valores de dicho parámetro así como también obtener una estimación del valor para la condición de parada del algoritmo, para que éste, en términos de rendimiento computacional sea óptimo. De igual manera se espera lograr la estimación de la tolerancia de activación y parada de la técnica *Watchdog*.

El **Algoritmo RCH** está implementado sobre la plataforma de alto nivel Scilab 5.0.3 (última versión estable liberada). Scilab es un software de cálculo científico orientado a la computación numérica, su base la constituyen las librerías que contienen cientos de rutinas de álgebra lineal, matrices esparcidas, polinomios, sistemas lineales, análisis numérico, entre otros; por lo cual se constituye como una completa herramienta para la aplicación del algoritmo propuesto por Nocedal y colaboradores [TJCD00].

4.1. Marco experimental

Los experimentos se dividieron en dos grupos, el primero de ellos se realizó con base en diversos problemas para programación no lineal propuestos por Klaus [Kla87], donde las soluciones numéricas fueron calculadas por su autor usando el código de programación no lineal: programación cuadrática sucesiva NLPQL de Schittkowski [Kla87]. Estos problemas resueltos permitieron comparar el **Algoritmo RCH**, ajustarlo y verificar su buena implementación. Un segundo grupo de problemas corresponden a polinomios de verosimilitud, generados en el estudio realizado por [JA09] a partir de un *corpus* de lenguaje de una gramática en *Forma Normal de*

Chomsky (FNC) [And08, And99], estos *corpus* se obtuvieron con el software realizado por Murillo [And08]. En ésta segunda serie de experimentos se estudia la aplicación del **Algoritmo RCH** en la estimación de los parámetros de una GIP.

4.1.1. Recursos de máquina

Se empleó una máquina con 4095MB de RAM, 500Gb de Disco duro y Procesador AMD Phenom(tm) 9650 Quad-Core 2.30Ghz, sobre los sistemas operativos Windows Vista Ultimate y Ubuntu 8.10 de 64 bits.

4.2. Primer grupo de experimentos

4.2.1. Metodología

Las condiciones experimentales del primer grupo de experimentos se muestran en las siguientes tablas. En la primera de ellas se detallan los problemas elegidos de la literatura [Kla87] y en la tabla 4.2 los puntos iniciales y mínimos de cada uno de los problemas nombrados, así como el valor de la función objetivo evaluada en dichos puntos. Estos valores son obtenidos por Klaus en [Kla87].

Problema	Función objetivo	Restricciones
Problema 1	$f(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2$	$g(x) = \quad x_1(x_1 - 4) - 2x_2 + 12$
Problema 2	$f(x) = \quad (x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2$	$g_1(x) = x_1 + 3x_2$ $g_2(x) = x_3 + x_4 - 2x_5$ $g_3(x) = x_2 - x_5$
Problema 3	$f(x) = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^4$	$g(x) = x_1(1 + x_2^2) + x_3^4 - 4 - 3\sqrt{2}$
Problema 4	$f(x) = -\sum_{i=1}^{10} x_i^2$	Para $j = 1:8$ y $i = 1:10$ $g_j(x) = (\sum_{i=1}^{10} x_i / e_{ij}) - 1$ $g_9(x) = (\sum_{i=1}^{10} x_i^2 / a_i^2) - 4$ $e_{ij} = \begin{cases} 1, & i \neq j; \\ 2, & i = j. \end{cases}$

Tabla 4.1: Primer grupo de experimentos

	Problema 1	Problema 2	Problema 3	Problema 4
Punto inicial x_0	$\begin{pmatrix} -1{,}2 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 2 \\ 2 \\ 2 \\ 2 \\ 2 \end{pmatrix}$	$\begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$
Punto mínimo x^*	$\begin{pmatrix} 2 \\ 4 \end{pmatrix}$	$\begin{pmatrix} -0{,}7674 \\ 0{,}2558 \\ 0{,}6279 \\ -0{,}1163 \\ 0{,}2558 \end{pmatrix}$	$\begin{pmatrix} 1{,}105 \\ 1{,}197 \\ 1{,}535 \end{pmatrix}$	$\begin{pmatrix} 0{,}1664 \\ 0{,}1664 \\ 0{,}1664 \\ 0{,}1664 \\ 0{,}1664 \\ 0{,}1664 \\ 0{,}1664 \\ 0{,}1664 \\ 2{,}843 \\ -2{,}843 \end{pmatrix}$
$f(x_0)$	24,2	6	1	10
$f(x^*)$	1	4,09302	0,0325	15,16

Tabla 4.2: Valores iniciales y puntos mínimos

4.2.2. Resultados utilizando el Algoritmo RCH

Las constantes utilizadas en el **Algoritmo RCH**, para el primer grupo de experimentos se observan en la tabla 4.3, éstas son sugeridas en el estudio realizado por Bigler, Nocedal y colaboradores [TJC].

Parámetros	Referencia	Valor sugerido
Γ	(3.5.102)	20
γ_{fd}	(3.5.105)	10
γ_k	(3.5.118)	$0{,}1(n-m)^{0{,}25}k^{-1{,}1}$
$\bar{\gamma}_k$	(3.5.120)	$0{,}01(n-m)^{0{,}25}k^{-1{,}1}$
Δ	Paso 6	0.1

Tabla 4.3: Constantes sugeridas

La tabla 4.4 muestra los resultados del grupo de experimentos asociados a la tabla 4.1. En la primera columna se indica el mínimizador de la función objetivo, en la segunda columna se muestra el número de iteraciones que empleó el **Algoritmo RCH** para encontrar el mínimo. En la tercera y cuarta están los valores de la función objetivo y las restricciones evaluadas en el mínimo, respectivamente. En la quinta columna se observa el tiempo, con una precisión de cien nanosegundos, tiempo de maquina que emplea el **Algoritmo RCH** para alcanzar el mínimo. Para determinar este tiempo se usó la herramienta *timer* disponible en la librería del software Scilab, la cual retorna el tiempo de CPU, es decir el número de ciclos del procesador utilizados para un cálculo. Este tiempo es independiente de los procesos que podrían ralentizar el equipo.

Mínimo (x^*)	Número de iteraciones	$f(x^*)$	$g(x^*)$	Tiempo (Seg)
Problema 1				
$\begin{pmatrix} 2{,}0001 \\ 4{,}0009 \end{pmatrix}$	36	1.0003	- 0,002	5.6718750
Problema 2				
$\begin{pmatrix} -0{,}7674 \\ 0{,}2558 \\ 0{,}6279 \\ -0{,}1163 \\ 0{,}2558 \end{pmatrix}$	7	4.0930	$\begin{pmatrix} 1{,}110^{-16} \\ -1{,}110^{-16} \\ 0 \end{pmatrix}$	1.296875
Problema 3				
$\begin{pmatrix} 1{,}0269 \\ 1{,}0472 \\ 1{,}2022 \end{pmatrix}$	312	$6{,}7065e^{-3}$	$-3{,}160e^{-4}$	16.6250
Problema 4[a]				
$\begin{pmatrix} -0{,}0918375 \\ -0{,}0918375 \\ -0{,}0918375 \\ -0{,}0918375 \\ -0{,}0918375 \\ -0{,}0918375 \\ -0{,}0918375 \\ -0{,}0918375 \\ -1{,}8136493 \\ 3{,}5024308 \end{pmatrix}$	60	$-15{,}6238$	$e^{-10}\begin{pmatrix} -0{,}0000355 \\ 0{,}0000178 \\ 0{,}0000311 \\ 0{,}0000222 \\ 0{,}0000355 \\ -0{,}0000622 \\ -0{,}0000355 \\ 0{,}0000355 \\ 136{,}26075 \end{pmatrix}$	9.1719

Tabla 4.4: Resultados

[a]Para este problema $\eta = 0{,}4$ en el cuadro 4.3.

Comparando los datos de la tabla 4.4, obtenidos con el **Algoritmo RCH** y los resultados (tabla 4.2) de Schittkowski [Kla87], se observaron valores similares, por lo cual se concluyó buen comportamiento [1] del **Algoritmo RCH**.

4.3. Segundo grupo de experimentos

4.3.1. Metodología

Se usaron cuatro gramáticas distintas en *Forma Normal de Chomsky (FNC)*, de cada una de estas gramáticas se generó un *corpus* de datos y a partir de él se construyó la función de verosimilitud, que es, como se observa en el capítulo 1, sección 1.5, un polinomio multivariado. En el capítulo mencionado se precisa que el problema a resolver es maximizar la función de verosimilitud. El **Algoritmo RCH** fue desarrollado para minimizar problemas, por lo tanto para realizar los experimentos se minimiza el negativo de la función, obteniendo así los valores que maximizan la función de verosimilitud.

Se desea comparar los resultados de este trabajo con los obtenidos en [JA09], tanto en calidad de modelos como en tiempo de ejecución, por ello se usan las mismas gramáticas con las que trabajaron Álvarez e Ibarra en [JA09].

En la tabla (4.5) se especifican las gramáticas usadas. El detalle de éstas se encuentra en el Anexo.

Gramática	Número de Terminales	Número de no terminales	Número de Reglas
Gramática 2	5	12	25
Gramática 3	14	13	47
Gramática 4	14	13	47
Gramática 6	2	5	8

Tabla 4.5: Gramáticas

La tabla 4.6 detalla cada problema de optimización, cuya función objetivo es la función de verosimilitud (un polinomio homogéneo), donde las restricciones son generadas a partir de la condición que establece la definición de una GIP, ecuación 1.3.2, así cada problema a minimizar es un problema de optimización con restricciones de igualdad. Las x_i son las probabilidades de las reglas numeradas en algún orden preestablecido

[1] En función de rendimiento y convergencia al mínimo.

	Gramática 2	Gramática 3	Gramática 4	Gramática 6
Número de cadenas	4000	4000	4000	4000
Grado del polinomio $f(x)$	6791	43817	44941	98956
Restricciones $g(x)$	$(\sum_{i=1}^{8} x_i) - 1$ $(\sum_{i=9}^{13} x_i) - 1$ $(\sum_{i=14}^{16} x_i) - 1$ $x_{17} - 1$ $x_{18} - 1$ $x_{19} - 1$ $x_{20} - 1$ $x_{21} - 1$ $x_{22} - 1$ $x_{23} - 1$ $x_{24} - 1$ $x_{25} - 1$	$(\sum_{i=1}^{4} x_i) - 1$ $(\sum_{i=5}^{12} x_i) - 1$ $(\sum_{i=13}^{15} x_i) - 1$ $(\sum_{i=16}^{18} x_i) - 1$ $(\sum_{i=19}^{20} x_i) - 1$ $(\sum_{i=12}^{29} x_i) - 1$ $(\sum_{i=30}^{37} x_i) - 1$ $x_{38} - 1$ $(\sum_{i=39}^{41} x_i) - 1$ $(\sum_{i=42}^{44} x_i) - 1$ $x_{45} - 1$ $x_{46} - 1$ $x_{47} - 1$	$(\sum_{i=1}^{3} x_i) - 1$ $(\sum_{i=4}^{6} x_i) - 1$ $(\sum_{i=7}^{9} x_i) - 1$ $(\sum_{i=10}^{17} x_i) - 1$ $(\sum_{i=18}^{26} x_i) - 1$ $(\sum_{i=27}^{28} x_i) - 1$ $(\sum_{i=29}^{31} x_i) - 1$ $(\sum_{i=33}^{35} x_i) - 1$ $(\sum_{i=37}^{39} x_i) - 1$ $(\sum_{i=40}^{47} x_i) - 1$	$(\sum_{i=1}^{4} x_i) - 1$ $x_5 - 1$ $x_6 - 1$ $x_7 - 1$ $x_8 - 1$

Tabla 4.6: Segundo grupo de experimentos

Los valores dados en la tabla 4.7, son las probabilidades asignadas inicialmente a cada una de las reglas, estos puntos iniciales fueron generados en el programa desarrollado por Álvarez e Ibarra [JA09]. El punto inicial para la Gramática 2 fue generado uniformemente; para el resto de las gramáticas se hace uso de las frecuencias relativas [And08, Rob04, And99].

Dichos puntos iniciales son puntos factibles.

	Gramática 2	Gramática 3	Gramática 4	Gramática 6
Vector inicial x_0	$\begin{pmatrix} 0{,}125 \\ 0{,}125 \\ 0{,}125 \\ 0{,}125 \\ 0{,}125 \\ 0{,}125 \\ 0{,}125 \\ 0{,}2 \\ 0{,}2 \\ 0{,}2 \\ 0{,}2 \\ 0{,}2 \\ 0{,}333 \\ 0{,}333 \\ 0{,}333 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0{,}216749771038974 \\ 0{,}234456090363285 \\ 0{,}334283097588277 \\ 0{,}21451104100946 \\ 0{,}105932634578203 \\ 0{,}1082731250636 \\ 0{,}223873002951053 \\ 0{,}112954106034395 \\ 0{,}120179098402361 \\ 0{,}107357280960619 \\ 0{,}106644957769411 \\ 0{,}114785794240358 \\ 0{,}208446689985749 \\ 0{,}404586086280606 \\ 0{,}386967223733644 \\ 0{,}274986093083627 \\ 0{,}272019284257371 \\ 0{,}452994622659002 \\ 0{,}458861231181994 \\ 0{,}541138768818006 \\ 0{,}329195585042957 \\ 0{,}080777815146604 \\ 0{,}084446754517381 \\ 0{,}089950163573546 \\ 0{,}079738282324884 \\ 0{,}080655517167578 \\ 0{,}083193200232366 \\ 0{,}083468370685174 \\ 0{,}088574311309505 \\ 0{,}114886179554949 \\ 0{,}114246738852417 \\ 0{,}190212294313241 \\ 0{,}113223633728366 \\ 0{,}115738767158326 \\ 0{,}119063858811493 \\ 0{,}113735186290391 \\ 0{,}118893341290818 \\ 1 \\ 0{,}318531943815134 \\ 0{,}34685092886271 \\ 0{,}334617127322157 \\ 0{,}331495296788842 \\ 0{,}331927775975781 \\ 0{,}336576927235377 \\ 1 \\ 1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0{,}359531606628957 \\ 0{,}225761635407363 \\ 0{,}213853329363898 \\ 0{,}200853428599782 \\ 0{,}109854123250968 \\ 0{,}106976282623797 \\ 0{,}116800635109656 \\ 0{,}112434256227052 \\ 0{,}227150937779101 \\ 0{,}106182395554232 \\ 0{,}110945717971619 \\ 0{,}109655651483576 \\ 0{,}195206357344271 \\ 0{,}402589079723148 \\ 0{,}402204562932581 \\ 0{,}308918277382163 \\ 0{,}444387928111224 \\ 0{,}246693794506612 \\ 0{,}452115158636898 \\ 0{,}547884841363102 \\ 0{,}0878581416798072 \\ 0{,}0890184760941358 \\ 0{,}0823539912528636 \\ 0{,}323941566748981 \\ 0{,}082086221772634 \\ 0{,}0843176341078813 \\ 0{,}0813721698253548 \\ 0{,}0831572996935527 \\ 0{,}0858944988247895 \\ 0{,}190087100788055 \\ 0{,}11281625881377 \\ 0{,}113065118208212 \\ 0{,}118871837411862 \\ 0{,}121443384487764 \\ 0{,}114309415180423 \\ 0{,}114848610535048 \\ 0{,}114558274574865 \\ 1 \\ 0{,}33625180095312 \\ 0{,}324282389449185 \\ 0{,}339465809597695 \\ 0{,}349015893063888 \\ 0{,}320387327649721 \\ 0{,}330596779286391 \\ 1 \\ 1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0{,}422665993239831 \\ 0{,}421927813823381 \\ 0{,}0745949726096585 \\ 0{,}08081122032713 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$

Tabla 4.7: Valores iniciales

En la tabla 4.8 se indican los valores de la función objetivo asociado a cada problema de

optimización del segundo grupo de experimentos, evaluados en los puntos iniciales de la tabla 4.7.

	Gramática 2	Gramática 3	Gramática 4	Gramática 6
$f(x_0)$	$-1{,}97967336977072320e^{-7}$	$-1{,}72325454368015370e^{-6}$	$-1{,}00344676859202520e^{-6}$	$-2{,}84763996588850450e^{-9}$

Tabla 4.8: Función objetivo evaluada en x_0

Los experimentos, para cada una de las gramáticas, se realizaron con distintos valores del parámetro "Tol", con el fin de observar el comportamiento del algoritmo para estimar buenos valores [a] de éste parámetro.

En la siguiente tabla se dan los valores del parámetro "Tol" con los cuales se realizaron los experimentos.

	Gramática 2	Gramática 3	Gramática 4	Gramática 6
Tol 1	10^{-5}	10^{-5}	10^{-2}	10^{-4}
Tol 2	10^{-6}	10^{-6}	10^{-3}	10^{-5}
Tol 3	10^{-7}	10^{-7}	10^{-4}	10^{-6}
Tol 4		10^{-8}		

Tabla 4.9: Parámetro "Tol" para cada una de las gramáticas

4.3.2. Resultados

En esta parte se muestran los resultados del segundo grupo de experimentos, para los cuales se utilizaron los mismos valores de las constantes indicadas en la tabla (4.3).

Se dan cuatro tablas, una para cada gramática, las cuales contienen los valores del número de iteraciones, el tiempo que tarda el algoritmo en alcanzar el mínimo y el valor de $\tilde{\sigma_k} = max\{\|Z_k^T g_k\|_\infty, \|c_k\|_\infty\}$, el cual es sugerido por Bigler y Nocedal [TJC], como tolerancia de parada para la activación de la técnica *watchdog*.

[a] Buenos valores en el sentido de menor tiempo de convergencia y mejor mínimo.

Los puntos mínimos obtenidos por el **Algoritmo RCH** se muestran en las tablas 4.14 a **??**, con sus respectivos parámetros "Tol".

Tol / Datos	10^{-5}	10^{-6}	10^{-7}
Número de iteraciones	3	16	69
$f(x^*)$	$-2{,}0596777356729288e^{-7}$	$-2{,}0390603226596105e^{-7}$	$-2{,}03701207559255583168e^{-7}$
Tiempo	3387.5370	64404.8610	431258.6200
$\tilde{\sigma_k}$	$2{,}2154757310932638e^{-6}$	$2{,}4581501876230760217 4e^{-6}$	$2{,}43392721355206730e^{-6}$

Tabla 4.10: Resultados gramática 2 - Otros valores

Tol / Datos	10^{-5}	10^{-6}	10^{-7}	10^{-8}
Número de iteraciones	16	14	35	270
$f(x^*)$	$-2{,}2302803207602115e^{-6}$	$-1{,}7240934396509376e^{-6}$	$-1{,}723584515336602792 67e^{-6}$	$-1{,}723669509690174e^{-6}$
Tiempo	15058.6406	13019.2031	17826.8130	128144.9550
$\tilde{\sigma_k}$	$9{,}45756535042363570e^{-6}$	$1{,}48833280388949160e^{-6}$	$6{,}6446882973636434144 3e^{-7}$	$6{,}13335615851084980e^{-7}$

Tabla 4.11: Resultados gramática 3 - Otros valores

Tol / Datos	10^{-2}	10^{-3}	10^{-4}
Número de iteraciones	134	140	165
$f(x^*)$	$-7{,}6124144809821583e^{-5}$	$-5{,}90437006352583374836e^{-5}$	$-6{,}3631614815485931e^{-5}$
Tiempo	57478.6875	61909.9263	91361.4848
$\tilde{\sigma_k}$	$8{,}11145363781018420e^{-4}$	$8{,}027325135351315310 57e^{-4}$	$9{,}3791361910089410e^{-4}$

Tabla 4.12: Resultados gramática 4 - Otros valores

Tol / Datos	10^{-4}	10^{-5}	10^{-6}
Número de iteraciones	39	38	171
$f(x^*)$	$-2{,}9495246552753322e^{-9}$	$-2{,}9453251713484649e^{-9}$	$-2{,}9387398422248857240 2e^{-9}$
Tiempo	411.3125	593.8593	6580.0900
$\tilde{\sigma_k}$	$2{,}8136188201521861e^{-5}$	$2{,}6973864670987169e^{-5}$	$2{,}63546008754111937833e-05$

Tabla 4.13: Resultados gramática 6 - Otros valores

	Parámetro "Tol"		
	10^{-4}	10^{-5}	10^{-6}
Mínimo x^*	$\begin{pmatrix} 0{,}12499971642612076 \\ 0{,}12500104278525193 \\ 0{,}12500130384822722 \\ 0{,}12500015632474681 \\ 0{,}12499992473375565 \\ 0{,}12500009920637356 \\ 0{,}12500005327713903 \\ 0{,}12500013732559867 \\ 0{,}20000015056049952 \\ 0{,}20000015820489520 \\ 0{,}20000021865396861 \\ 0{,}20000022580904137 \\ 0{,}19999982142697936 \\ 0{,}33333300216632894 \\ 0{,}33333304573938405 \\ 0{,}33333307929412426 \\ 0{,}99999970298298113 \\ 1{,}00000014300586430 \\ 0{,}99999981150789019 \\ 0{,}99999989877343576 \\ 0{,}99999973908136253 \\ 0{,}99999971393505094 \\ 0{,}99999969941069911 \\ 0{,}99999958455745963 \\ 0{,}99999957096282144 \end{pmatrix}$	$\begin{pmatrix} 0{,}12499971642612076 \\ 0{,}12500104278525193 \\ 0{,}12500130384822722 \\ 0{,}12500015632474681 \\ 0{,}12499992473375565 \\ 0{,}12500009920637356 \\ 0{,}12500005327713903 \\ 0{,}12500013732559867 \\ 0{,}20000015056049952 \\ 0{,}20000015820489520 \\ 0{,}20000021865396861 \\ 0{,}20000022580904137 \\ 0{,}19999982142697936 \\ 0{,}33333300216632894 \\ 0{,}33333304573938405 \\ 0{,}33333307929412426 \\ 0{,}99999970298298113 \\ 1{,}00000014300586430 \\ 0{,}99999981150789019 \\ 0{,}99999989877343576 \\ 0{,}99999973908136253 \\ 0{,}99999971393505094 \\ 0{,}99999969941069911 \\ 0{,}99999958455745963 \\ 0{,}99999957096282144 \end{pmatrix}$	$\begin{pmatrix} 0{,}12499968240467663 \\ 0{,}12500034119875730 \\ 0{,}12500142318358820 \\ 0{,}12500030608220231 \\ 0{,}12499991262040959 \\ 0{,}12500021520315951 \\ 0{,}12500014170484330 \\ 0{,}12500019307809432 \\ 0{,}20000017036711876 \\ 0{,}20000020384206027 \\ 0{,}20000028546841861 \\ 0{,}20000030862567367 \\ 0{,}19999984292615913 \\ 0{,}33333295420472636 \\ 0{,}33333299635626001 \\ 0{,}33333302881627763 \\ 0{,}99999959428414376 \\ 1{,}00000007750904300 \\ 0{,}99999970136043326 \\ 0{,}99999978828986447 \\ 0{,}99999979248535809 \\ 0{,}99999985364370125 \\ 0{,}99999979745242684 \\ 0{,}99999971271508037 \\ 0{,}99999967617752350 \end{pmatrix}$

Tabla 4.14: Resultados gramática 2 - Puntos mínimos

	Parámetro "Tol"			
	10^{-5}	10^{-6}	10^{-7}	10^{-8}
Mínimo x^*	0,21680796698423535 0,23438786726573019 0,33426593030515050 0,21454552928427853 0,10596885327978722 0,10827121157556645 0,22384367937098282 0,11295285964786114 0,12017463491527369 0,10734807811831448 0,10664536919791356 0,11478585632895021 0,20844880173924807 0,40458603931399822 0,38696819944683514 0,27498604875747285 0,27201589642168517 0,45299126456737698 0,45885754382977095 0,54113503708209831 0,32919177916845677 0,08077523204071291 0,08444437773383764 0,08995108583936037 0,07973895304670163 0,08065639316661774 0,08319407535342392 0,08347026813043250 0,08857639014847982 0,11488713053304292 0,11424757946884920 0,19021337868077418 0,11322462111973637 0,11573987761302833 0,11906500139387705 0,11373608721495082 0,11889468591042965 1,00000068685204010 0,31853258305231130 0,34685159327189524 0,33461776722337172 0,33149642183778161 0,33192898725272907 0,33657764760319403 1,00000041550458560 1,00000045265540670 0,99999986075144343	0,21674821820261972 0,23445889552068891 0,33428259102268304 0,21451178358681225 0,10593349893221445 0,10827252179432116 0,22387232324377493 0,11295412379493945 0,12017913311374191 0,10735725373243059 0,10664502921159974 0,11478585195970969 0,20844678320809409 0,40458607618474751 0,38696737716477281 0,27498608426451088 0,27201874088705741 0,45299408404397667 0,45886066153107546 0,54113819207160530 0,32919500031587229 0,08077742593858937 0,08444639829936382 0,08995033479738718 0,07973841332766669 0,08065568099401217 0,08319336391866798 0,08346859847411151 0,08857456810692065 0,11488624940119072 0,11424679105224315 0,19021238548948988 0,11322370939851271 0,11573886250612200 0,11906395929575134 0,11373524813319899 0,11889347408417524 1,00000003602224120 0,31853197437440234 0,34685096344629074 0,33461715798758351 0,33149540142963141 0,33192789440387438 0,33657696716505164 0,99999998599047168 0,99999999659868222 0,99999993557711930	0,21674958554000404 0,23445702358572512 0,33428302394922021 0,21451103139388028 0,10593263727646185 0,10827287362419827 0,22387278469605298 0,11295410342253351 0,12017906701794241 0,10735726639612972 0,10664495791390401 0,11478579382645139 0,20844669326690549 0,40458608422900161 0,38696725982634067 0,27498609122954348 0,27201915771370966 0,45299449722456014 0,45886109597412711 0,54113863195504353 0,32919544586431843 0,08077772157474560 0,08444666864075991 0,08995020072819065 0,07973831009757569 0,08065555259670380 0,08319323562880247 0,08346843267767569 0,08857438006853978 0,11488620547993626 0,11424676066121127 0,19021232521367229 0,11322366101184285 0,11573879903181231 0,11906389188310108 0,11373521034849997 0,11889338189886946 1,00000001704944741 0,31853195933817363 0,34685094532446270 0,33461714286995947 0,33149533001428144 0,33192781241724229 0,33657694536629229 1,00000000615778939 1,00000000808508283 0,99999998990927375	0,21674961728522782 0,23445692659868331 0,33428303986067481 0,21451102959102991 0,10593262559964786 0,10827286498432646 0,22387276733992398 0,11295410204093517 0,12017905544226172 0,10735725563824156 0,10664495562252425 0,11478579181248150 0,20844669122534165 0,40458608473355961 0,38696725440912566 0,27498609164192644 0,27201917748374593 0,45299451682239383 0,45886111553512848 0,54113865177299303 0,32919546576326408 0,08077773439339267 0,08444668026474678 0,08995019325197243 0,07973830407787358 0,08065554538836697 0,08319322842553938 0,08346842848646539 0,08857437482684681 0,11488620736291023 0,11424676318320995 0,19021232632422536 0,11322366268391568 0,11573879999130175 0,11906389265658614 0,11373521252130062 0,11889338150234835 1,00000001970573680 0,31853196207680862 0,34685094791736515 0,33461714560474942 0,33149533026358513 0,33192781216726930 0,33657694795896820 1,00000001098259390 1,00000001227508920 0,99999999450339216

Tabla 4.15: Resultados gramática 3 - Puntos mínimos

	Parámetro "Tol"		
	10^{-2}	10^{-3}	10^{-4}
Mínimo x^*	$\begin{pmatrix} 0{,}34029130381456929 \\ 0{,}24704720930018859 \\ 0{,}21264731616437654 \\ 0{,}20085632902997674 \\ 0{,}10976411187351660 \\ 0{,}10604751163340470 \\ 0{,}11683988079610193 \\ 0{,}11242083897204774 \\ 0{,}22711503137820699 \\ 0{,}10618890011045871 \\ 0{,}11102266797955286 \\ 0{,}10966314363760962 \\ 0{,}19526349303599305 \\ 0{,}40258817123526192 \\ 0{,}40220358268927336 \\ 0{,}30891916785041851 \\ 0{,}44438773291342992 \\ 0{,}24669107796300177 \\ 0{,}45211797259271758 \\ 0{,}54788487605942859 \\ 0{,}08786456386870364 \\ 0{,}08902571311635826 \\ 0{,}08236113775043764 \\ 0{,}32393733804947317 \\ 0{,}08209073529242646 \\ 0{,}08432428623858418 \\ 0{,}08138001641206435 \\ 0{,}08316233981354423 \\ 0{,}08590142335126676 \\ 0{,}19008688014438294 \\ 0{,}11281587290175736 \\ 0{,}11306464493621671 \\ 0{,}11887177128014530 \\ 0{,}12144337673595328 \\ 0{,}11430942262348094 \\ 0{,}11484844242082368 \\ 0{,}11455825737981433 \\ 1{,}00000021933711070 \\ 0{,}33625129647157531 \\ 0{,}32428181671682493 \\ 0{,}33946750281276483 \\ 0{,}34901594380947365 \\ 0{,}32038699269647708 \\ 0{,}33059701423585419 \\ 0{,}99999640577962334 \\ 0{,}99999897809011096 \\ 0{,}99999728870523219 \end{pmatrix}$	$\begin{pmatrix} 0{,}34002440416822760 \\ 0{,}24708918368685062 \\ 0{,}21267186577656125 \\ 0{,}20094295559549372 \\ 0{,}10987105930915386 \\ 0{,}10596393942290254 \\ 0{,}11679316047486202 \\ 0{,}11245537299409952 \\ 0{,}22721722788178023 \\ 0{,}10618304001241355 \\ 0{,}11100150653924450 \\ 0{,}10971196085200878 \\ 0{,}19525100120260924 \\ 0{,}40258987232378907 \\ 0{,}40220311861491581 \\ 0{,}30891895577919870 \\ 0{,}44438599078116797 \\ 0{,}24669029163265205 \\ 0{,}45211550089249875 \\ 0{,}54788347034845764 \\ 0{,}08786437844385991 \\ 0{,}08902488601652959 \\ 0{,}08236096275339407 \\ 0{,}32393651491036529 \\ 0{,}08209036222869144 \\ 0{,}08432470016862204 \\ 0{,}08137969518347951 \\ 0{,}08316355313999919 \\ 0{,}08590148033681066 \\ 0{,}19008756922381476 \\ 0{,}11281630195690692 \\ 0{,}11306471813590596 \\ 0{,}11887166528392153 \\ 0{,}12144350460355932 \\ 0{,}11430957996716423 \\ 0{,}11484875111502375 \\ 0{,}11455836780991416 \\ 0{,}99999907326216386 \\ 0{,}33625052901181313 \\ 0{,}32428107830911557 \\ 0{,}33946556807507133 \\ 0{,}34901599585956122 \\ 0{,}32038732461533831 \\ 0{,}33059687839208640 \\ 0{,}99999736885796520 \\ 0{,}99999744512815403 \\ 0{,}99999786892186782 \end{pmatrix}$	$\begin{pmatrix} 0{,}34015201379376497 \\ 0{,}24694628830422924 \\ 0{,}21270039303731519 \\ 0{,}20093681741763855 \\ 0{,}10986272720788681 \\ 0{,}10596829433076839 \\ 0{,}11679635892794411 \\ 0{,}11245383396279038 \\ 0{,}22721199678488926 \\ 0{,}10618410704624084 \\ 0{,}11100222781570390 \\ 0{,}10970930855999543 \\ 0{,}19525024801764390 \\ 0{,}40259013501253016 \\ 0{,}40220337918200094 \\ 0{,}30891898701952342 \\ 0{,}44438600025441582 \\ 0{,}24669025191274729 \\ 0{,}45211564614867261 \\ 0{,}54788356136051097 \\ 0{,}08786458947497838 \\ 0{,}08902511283622761 \\ 0{,}08236118800423749 \\ 0{,}32393651667363338 \\ 0{,}08209053575943277 \\ 0{,}08432491588463052 \\ 0{,}08137993411527877 \\ 0{,}08316364790678063 \\ 0{,}08590161174821379 \\ 0{,}19008756030686769 \\ 0{,}11281628972045510 \\ 0{,}11306470407404523 \\ 0{,}11887165915808856 \\ 0{,}12144349970065042 \\ 0{,}11430957536451949 \\ 0{,}11484874311552741 \\ 0{,}11455836270735680 \\ 0{,}99999906774307534 \\ 0{,}33625052225125734 \\ 0{,}32428107021925401 \\ 0{,}33946560409961102 \\ 0{,}34901598575884252 \\ 0{,}32038730702977725 \\ 0{,}33059687185235564 \\ 0{,}99999727545974537 \\ 0{,}99999742062550445 \\ 0{,}99999785231243821 \end{pmatrix}$

Tabla 4.16: Resultados gramática 4 - Puntos mínimos

En las figuras 4.1 y 4.2 se muestran las gráficas de la función objetivo de los problemas con sus parámetros "Tol". Esta representación gráfica facilitó el análisis experimental de los resultados obtenidos.

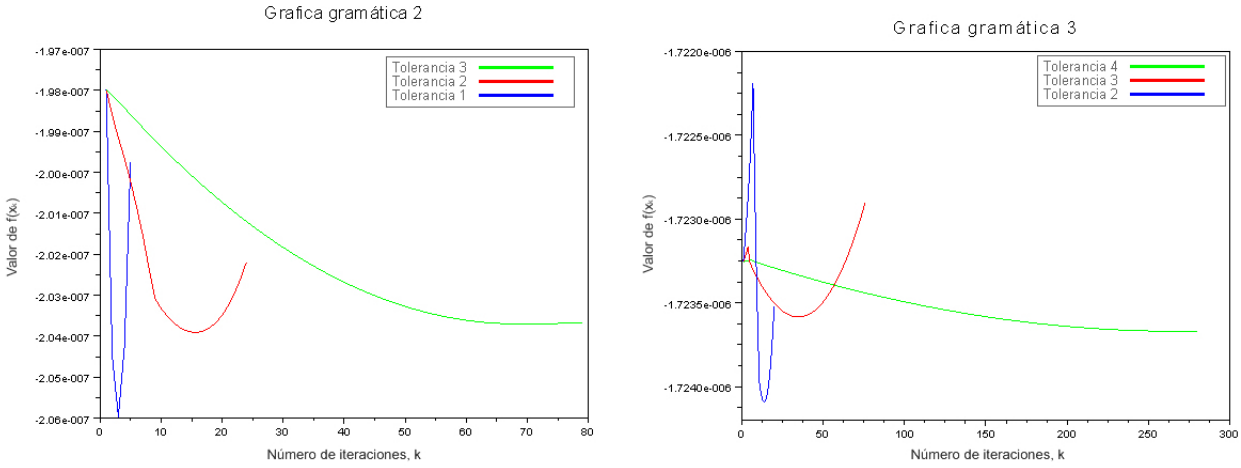


Figura 4.1: Gráficas de las gramáticas 2 y 3

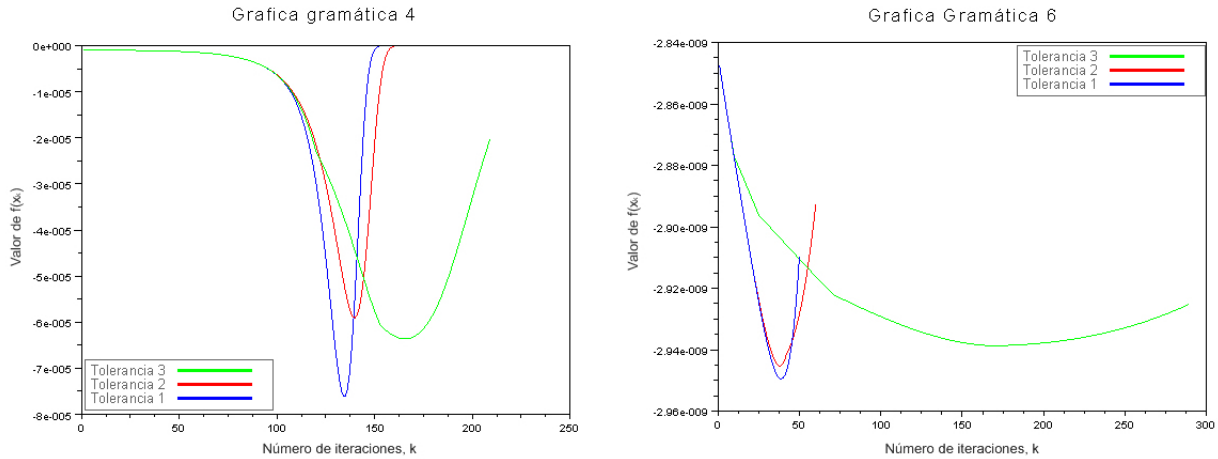


Figura 4.2: Gráficas de las gramáticas 4 y 6

Perplejidad

La evaluación de la calidad de los modelos obtenidos es un problema presente en cualquier proceso de estimación de los parámetros de un modelo de lenguaje. Usualmente, en problemas de Modelado del Lenguaje (ML) para evaluar la bondad de los modelos obtenidos a partir de las gramáticas se usa la *perplejidad* por palabra (PP). Intuitivamente, esta medida considera la capacidad del modelo para tratar los eventos que se van produciendo en el proceso de análisis, es decir, asignar probabilidades adecuadas a las frases del lenguaje. Esta medida se calcula sobre un conjunto de datos que no han sido utilizados en el proceso de entrenamiento denominado conjunto de test (Ts). Cuando el modelo es una GIP la perplejidad se define como

$$PP(Ts, G_p) = \exp\left\{-\frac{\sum_{x \in T_s} log(Pr(x|Gp))}{\sum_{x \in T_s} |x|}\right\} \tag{4.3.1}$$

Entre menor sea la perplejidad, el modelo tiene mayor capacidad expresiva.

A continuación se muestran las medidas de *perplejidad* de cada uno de los experimentos con cada una de las gramáticas.

Gramáticas	Parámetro "Tol"			
Gramática 2	10^{-4}	10^{-5}	10^{-6}	
PP=	1.68893082987088161	1.68893082987088161	1.68893124088994950	
Gramática 3	10^{-5}	10^{-6}	10^{-7}	10^{-8}
PP=	8.61185516781983651	8.61190840709166849	8.61190928438773007	8.61190929956550377
Gramática 4	10^{-2}	10^{-3}	10^{-4}	
PP=	13.8353274675043547	13.8358630745969737	13.835358963477276	
Gramática 6	10^{-4}	10^{-5}	10^{-6}	
PP=	1.24306003818340494	1.24306020442906995	1.24305946049934235	

Tabla 4.17: Pruebas de perplejidad por palabra

Capítulo 5

Conclusiones y trabajos futuros

Concluido el estudio teórico del algoritmo propuesto en [TJC], su implementación y los experimentos, nos permite establecer lo siguiente

5.1. Conclusiones

- El algoritmo implementado es efectivo para el desarrollo de problemas de optimización a gran escala con restricciones de igualdad. El tiempo de ejecución por iteración no es excesivo, obteniendo así una buena herramienta para la práctica.

- Las constantes y las tolerancias en la tabla 4.3, propuestas por Bigler, Nocedal y colaboradores [TJCD00], constituyen una buena elección para el desarrollo práctico del algoritmo.

- La herramienta elegida, Scilab 5.1.1, sobre la cual se implementó el **Algoritmo RCH**, posee las librerías necesarias para una práctica implementación del algoritmo.

- El usuario debe establecer un equilibrio entre el orden de los valores con los que se trabaja y el parámetro "Tol", para evitar inestabilidad.

- En los resultados dados en las tablas 4.10 a 4.13 se observa que a menor valor del parámetro "Tol" mejor es el valor del mínimo obtenido, a un menor número de iteraciones y por ende menor tiempo de ejecución del algoritmo. Esto se evidencia en las figuras 4.1 y 4.2.

- El valor del parámetro "Tol" se debe establecer de tal manera que permita que el valor de α tome valores cercanos a uno, para lograr que el algoritmo converja mucho mas rápido, consiguiendo así la convergencia superlineal que teóricamente tiene el algoritmo.

- En la tabla 4.17 se observa que la perplejidad por palabra en los modelos obtenidos de cada una de las gramáticas, son equivalentes. Por lo tanto el parámetro "Tol" no es determinante para los valores de PP.

- Sobre los sistemas operativos Windows Vista Ultimate y Ubuntu 8.10 de 64 bits, el algoritmo genero similares resultados con respecto al tiempo de ejecución por iteración, Por lo tanto el usuario puede usar cualquiera de los sistemas operativos nombrados y obtener resultados equivalentes.

5.2. Trabajos futuros

- Implementar y estudiar el rendimiento del **Algoritmo RCH** sobre el lenguaje de programación C y/o Fortran.

- Realizar el estudio comparativo, en términos de tiempo de ejecución y calidad de los modelos obtenidos con el **Algoritmo RCH** y los obtenidos con el algoritmo clásico de estimación.

- Estudiar la utilización del **Algoritmo RCH** para estimar parámetros de gramáticas de alta complejidad y basadas en *corpus* grandes.

ANEXOS

Gramáticas

En las siguientes tablas se presentan en detalle las cuatro gramáticas con que se realizó el segundo grupo de experimentos.

Terminales 5	No Terminales 12	Reglas 25
$t0$ $t1$ $t2$ $t3$ $t4$	$A0$ $A1$ $A2$ $A3$ $A4$ $A5$ $A6$ $A7$ $A8$ $A9$ $A10$ $A11$	$A0 - - > A3A7$ $A0 - - > A2A11$ $A0 - - > A4A10$ $A0 - - > A4A4$ $A0 - - > A5A0$ $A0 - - > A5A4$ $A0 - - > A6A6$ $A0 - - > t2$ $A1 - - > A2A11$ $A1 - - > A4A4$ $A1 - - > A5A0$ $A1 - - > A6A6$ $A1 - - > t2$ $A2 - - > A1A9$ $A2 - - > A4A8$ $A2 - - > t2$ $A3 - - > t3$ $A4 - - > t1$ $A5 - - > t2$ $A6 - - > t4$ $A7 - - > A1A0$ $A8 - - > A1A4$ $A9 - - > A2A0$ $A10 - - > A4A0$ $A11 - - > A4A1$

Tabla 5.1: Gramática 2

Terminales 5	No Terminales 12	Reglas 25
a b	S A B C D	$S - - > AC$ $S - - > BD$ $S - - > AA$ $S - - > BB$ $A - - > a$ $B - - > b$ $C - - > SA$ $D - - > SB$

Tabla 5.2: Gramática 6

Terminales 14	No Terminales 13	Reglas 47
$t0$ $t1$ $t2$ $t3$ $t4$ $t5$ $t6$ $t7$ $t8$ $t9$ $t10$ $t11$ $t12$ $t13$	$A0$ $A2$ $A4$ $A5$ $A6$ $A7$ $A8$ $A9$ $A10$ $A11$ $A12$ $A13$ $A14$	$A0 - - > A2A14$ $A0 - - > A2A4$ $A0 - - > A5A12$ $A0 - - > A5A13$ $A2 - - > t1$ $A2 - - > t2$ $A2 - - > t3$ $A2 - - > t4$ $A2 - - > t5$ $A2 - - > t6$ $A2 - - > t7$ $A2 - - > t8$ $A4 - - > A11A7$ $A4 - - > A6A9$ $A4 - - > A8A7$ $A5 - - > A11A7$ $A5 - - > A6A9$ $A5 - - > A8A7$ $A6 - - > A11A7$ $A6 - - > A8A7$ $A7 - - > A8A7$ $A7 - - > t1$ $A7 - - > t2$ $A7 - - > t3$ $A7 - - > t4$ $A7 - - > t5$ $A7 - - > t6$ $A7 - - > t7$ $A7 - - > t8$ $A8 - - > t1$ $A8 - - > t2$ $A8 - - > t3$ $A8 - - > t4$ $A8 - - > t5$ $A8 - - > t6$ $A8 - - > t7$ $A8 - - > t8$ $A9 - - > A10A6$ $A10 - - > t10$ $A10 - - > t3$ $A10 - - > t9$ $A11 - - > t11$ $A11 - - > t12$ $A11 - - > t13$ $A12 - - > A2A4$ $A13 - - > A2A9$ $A14 - - > A4A9$

Tabla 5.3: Gramática 3

Terminales 14	No Terminales 13	Reglas 47
$null$ fly $flies$ $like$ $likes$ $time$ $times$ $arrow$ $arrows$ of $with$ the a an	$Sentence$ $Verb$ $Object$ $Subject$ $NounPhrase$ $NounGroup$ $Noun$ $PrePhrase$ $Preposition$ $Article$ $D[Verb_Object]$ $D[Verb_PrePhrase]$ $D[Object_PrePhrase]$	$Sentence - - > SubjectD[Verb_Object]$ $Sentence - - > SubjectD[Verb_PrePhrase]$ $Sentence - - > VerbD[Object_PrePhrase]$ $Sentence - - > VerbObject$ $Article - - > a$ $Article - - > an$ $Article - - > the$ $D[Object_PrePhrase] - - > ObjectPrePhrase$ $D[Verb_Object] - - > VerbObject$ $D[Verb_PrePhrase] - - > VerbPrePhrase$ $Noun - - > arrow$ $Noun - - > arrows$ $Noun - - > flies$ $Noun - - > fly$ $Noun - - > like$ $Noun - - > likes$ $Noun - - > time$ $Noun - - > times$ $NounGroup - - > arrow$ $NounGroup - - > arrows$ $NounGroup - - > flies$ $NounGroup - - > fly$ $NounGroup - - > like$ $NounGroup - - > likes$ $NounGroup - - > NounNounGroup$ $NounGroup - - > time$ $NounGroup - - > times$ $NounPhrase - - > ArticleNounGroup$ $NounPhrase - - > NounNounGroup$ $Object - - > ArticleNounGroup$ $Object - - > NounNounGroup$ $Object - - > NounPhrasePrePhrase$ $PrePhrase - - > PrepositionNounPhrase$ $Preposition - - > like$ $Preposition - - > of$ $Preposition - - > with$ $Subject - - > ArticleNounGroup$ $Subject - - > NounNounGroup$ $Subject - - > NounPhrasePrePhrase$ $Verb - - > arrow$ $Verb - - > arrows$ $Verb - - > flies$ $Verb - - > fly$ $Verb - - > like$ $Verb - - > likes$ $Verb - - > time$ $Verb - - > times$

Tabla 5.4: Gramática 4

Bibliografía

[A.01] Epelman Marina A., *Continuous optimization methods*, www-personal.umich.edu, 2001.

[And99] Sánchez Peiró Joan Andreu, *Estimación de gramáticas incontextuales probabilísticas y su aplicación en modelación del lenguaje*, Ph.D. thesis, Universidad Politécnica de Valencia. Departamento de Sistemas Informáticos y computación, 1999.

[And08] Murillo Fernández Edwin Andrés, *Reduccion de gramáticas libres de contexto, aproximación computacional*, Trabajo de grado, Universidad del Cauca. Departamento de Matemáticas, 2008.

[C.91] Davidson William C., *Variable metric method for minimization*, Siam Journal On Optimization **1** (1991), 1–17.

[F.84] Coleman Thomas F., *Lecture notes in computer science*, primera ed., Springer-Verlag, 1984.

[GY08] Luenberger David G. and Yinyu Ye, *Linear and nonlinear programming*, tercera ed., Springer, 2008.

[HJ88] Byrd Richar H. and Nocedal Jorge, *An analysis of reduced hessian Methods for constrained optimization*, Reporte técnico, University of Colorado. Department of Computer Science, 1988.

[JA09] Ibarra Johnny and Alvarez Astrid, *Uso del algoritmo inside-outside en la estimación de parámetros de una gramática incontextual probabilística.*, Trabajo de grado, Universidad del Cauca. Departamento de Matemáticas, 2009.

[JET08] Flum Jörg, Grädel Erich, and Wilke Thomas, *Logic and automata, history and perspective*, primera ed., vol. 2, Amsterdam University Press, 2008.

[JJ06] Nocedal Jorge and Wright Stephen J., *Numerical optimization*, segunda ed., Springer Science, 2006.

[JRD01] Hopcroft Jhon, Motwani Rajeev, and Ullman Jeffrey D., *Introduction to automata theory, languages, and computation*, segunda ed., Addison-Wesley, 2001.

[Kla87] Schittkowski Klaus, *Lecture notes in economics and mathematical system*, segunda ed., Springer-Verlaq, 1987.

[Rob04] Freddy Angel Amaya Robayo, *Nuevas alternativas para la estimación de los parámetros en una gramática incontextual probabilística.*, Proyecto de investigación, Universidad del Cauca. Departamento de Matemáticas, 2004.

[Rog81] Fletcher Roger, *Practical methods of optimization*, primera ed., vol. 2, John Wiley and Sons, 1981.

[SY06] Wenyu Sun and Ya-Xiang Yuan, *Optimization theory and methods nonlinear programming*, primera ed., Springer Science, 2006.

[TJC] Biegler Lorenz T., Nocedal Jorge, and Schmid Claudia., *A reduced hessian method for large scale constrained optimization*, Siam Journal On Optimization **5**, no. 2, 314–347.

[TJCD00] Biegler Lorenz T., Nocedal Jorge, Schmid Claudia, and Ternet David, *Numerical experience with a reduced hessian method for large scale constrained optimization*, Computational Optimization and Applications **15** (2000), no. 1.