UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS
DIVISION DE ESTUDIOS DE POSGRADO

METODOS DE OPTIMIZACION DE GRAN ESCALA Y ALGUNAS APLICACIONES A FUNCIONES PARCIALMENTE SEPARABLES

T E S I S

QUE PARA OBTENER EL GRADO ACADEMICO DE

MAESTRO EN CIENCIAS (MATEMATICAS)

Incluye Diskette

PRESENTA

IRMA DELIA GARCIA CALVILLO

DIRECTOR DE TESIS: DR. PABLO BARRERA SANCHEZ

Gracias a Dios por permitirme existir, por darme una familia tan maravillosa que en todo momento me acompañó en la elaboración de este trabajo y por poner en mi camino la gente idónea que me ha permitido desarrollarme como persona.

Gracias a Miguel Angel por su comprensión, su apoyo, su motivación, por no dejarme dar por vencida, por confiar en mí en todo momento.

Gracias a mis padres, Irma y Juvenal, por su amor, su cariño, sus enseñanzas, su amistad, por la libertad para escoger mi camino y por su preocupación constante. A mis hermanos, Norma, Olivia, Juvenal y Dariela, gracias por su apoyo, por poder contar con ustedes siempre.

Agradecimientos

Agradezco profundamente al Dr. Pablo Barrera Sánchez por haberme brindado su amistad, su apoyo, por haberme guiado durante mis estudios, por su dedicación y paciencia, por ser una persona de excelente calidad humana, gracias. Gracias por su valiosa asesoría para la realización de esta tesis.

A Guilmer González, le agradezco su amistad, su paciencia y todo el apoyo computacional brindado para la elaboración del sistema UNAMALLA.

Por la amistad, compañía y apoyo, agradezco profundamente al grupo de trabajo UNAMALLA, a Mary, Luis, Mónica y Adriana.

A Efrén, Sergio y Gerardo por su invaluable amistad, gracias.

Deseo agradecer de manera muy especial al Dr. Humberto Madrid de la Vega, por su constante apoyo, motivación y ayuda en todo momento.

Agradezco al Consejo Nacional de Ciencia y Tecnología la beca otorgada para la realización de mis estudios y al Programa de Mejoramiento al Profesorado de la SEP, el apoyo recibido para la redacción de esta tesis.

Contenido

Prólogo

La generación de mallas en regiones planas irregulares ha tenido un gran desarrollo en los últimos años, debido principalmente a su aplicación en la solución de ecuaciones diferenciales parciales. Una malla sobre una región poligonal es la subdivisión de la región en un conjunto finito de regiones simples, llamadas celdas, las cuales usualmente deben presentar alguna propiedad geométrica.

La generación de una malla que presente ciertas propiedades geométricas en una región dada, puede verse como un problema de optimización de gran escala [7]. El estudio de los métodos de optimización de gran escala efectivos, económicos y adecuados para resolver el problema de la generación de mallas, así como la implementación de estos métodos en un sistema computacional, el sistema UNAMALLA, motivó el presente trabajo.

En el primer capítulo presentamos una introducción a los métodos de optimización, presentado los conceptos básicos de la optimización sin restricciones así como algunos de los métodos de descenso mas comunes: máximo descenso y el método de Newton. En el capítulo dos se trata la teoría y algoritmos de la búsqueda lineal, encontrar un tamaño de paso adecuado en los métodos de descenso. En el tercer capítulo trabajamos el tema de gradiente conjugado lineal, visto como un método de minimización de funciones cuadráticas con un método de descenso con búsqueda lineal. En el capítulo cuatro retomamos el problema de minimizar una función cuadrática, pero ahora se busca en mínimo restringido a una región dada, éste tema será básico para presentar los métodos de región de confianza, discutidos con detalle en el capítulo cinco.

En el capítulo seis se presenta una alternativa económica y confiable al método de Newton: los métodos Quasi–Newton. Los métodos recomendables para problemas de gran escala se presentan en el capítulo siete. El capítulo ocho presenta las funciones parcialmente separables, se muestra que las funciones a optimizar en la generación de mallas resultan parcialmente separables y se resuelve el problema de generar mallas en regiones irregulares con un método eficiente de optimización, que no es muy utilizado

en la práctica, pero que resulta bastante económico ya que explota la estructura de la función objetivo; con este método hemos obtenido muy buenas configuraciones de mallas en regiones irregulares. Por último, en los apéndices, se presenta una descripción de los funcionales a optimizar en la generación de mallas, así como la presentación del sistema UNAMALLA v.2.0 para PC: Sistema para generar mallas óptimas en regiones planas irregulares, anexándose un manual para el manejo de este sistema. Actualmente se cuenta con versiones del sistema UNAMALLA bajo varias plataformas: PC, Sun Microsystems, Silicon Graphics, Linux Intel ELF y ambiente de trabajo Matlab, éstas pueden obtenerse gratuitamente en la página del grupo de trabajo UNAMALLA:

http://www.mathmoo.unam.mx/unamalla

Capítulo 1

Introducción y Conceptos Básicos

Estamos interesados en resolver el problema de minimizar una función objetivo que depende de variables reales, sin restricciones en los valores de estas variables. Esto es, queremos resolver

$$\min_x f(x)$$

donde $x \in \mathbb{R}^n$ y $f : \mathbb{R}^n \to \mathbb{R}$ una función suave. Usualmente se conoce la primera derivada de la función y en algunos casos el valor de la segunda derivada con los cuales se pueden obtener algoritmos eficientes para resolver el problema.

Los algoritmos de optimización son procesos iterativos. Se inician con un valor inicial de las variables y se genera una sucesión donde se van obteniendo mejores valores de las variables, hasta llegar a la solución buscada. Usualmente éstos son programados para que una computadora realice las iteraciones.

Los algoritmos se diferencían unos de otros en la estrategia utilizada para moverse de una iteración a la siguiente, la mayoría de las estrategias utilizan los valores de la función objetivo, de la derivada de la función y posiblemente de la segunda derivada. Algunos algoritmos acumulan información de iteraciones anteriores mientras que otros solo utilizan información de la iteración actual. Independientemente de la estrategia para pasar de una iteración a otra, los algoritmos deben contar con las propiedades de ser

- Robustos. Esto es, deben trabajar bien en una variedad de problemas para puntos iniciales razonables.
- Eficientes. Deben economizar tiempo de cómputo y memoria.
- Precisos. Deben identificar la solución con precisión, no deben ser sensibles

a errores en los datos o a errores de redondeo debidos a la aritmética de la computadora.

Esto parecerá conflictivo, por ejemplo, un método rápidamente convergente puede requerir mucho tiempo y memoria en problemas grandes. Por otro lado, un método robusto puede también ser lento. Estas relaciones entre razón de convergencia y requerimientos en memoria, y entre robustez y velocidad, son problemas centrales en optimización numérica.

La teoría matemática de optimización es utilizada tanto para caracterizar puntos óptimos, como para dar la base de la mayoría de los algoritmos prácticos. Entender la teoría es muy importante para desarrollar buenos algoritmos en la práctica.

1.1 Caracterización de la solución

La solución ideal de nuestro problema es encontrar un mínimo global de f, esto es, el punto donde la función alcanza su menor valor, la definición formal es

Un punto x^* es un mínimo global si $f(x^*) \leq f(x) \forall x \in \mathbb{R}^n$

El mínimo global es difícil de encontrar ya que usualmente solo se conoce f de manera local. La mayoría de los algoritmos pueden determinar solo un mínimo local, un punto donde se alcanza el menor valor de f en una vecindad, la definición formal es

Un punto x^* es un mínimo local si existe una vecindad $\mathcal{N}$ de x^* tal que $f(x^*) \leq f(x) \forall x \in \mathcal{N}$

Un punto que satisface esta definición es llamado un mínimo local no estricto, mientras que un mínimo local estricto se define como

Un punto x^* es un mínimo local estricto si existe una vecindad $\mathcal{N}$ de x^* tal que $f(x^*) < f(x) \forall x \in \mathcal{N}$ con $x \neq x^*$

1.2 Condiciones de Optimalidad

De las defincciones anteriores parece que la única forma de encontrar un punto x^* que sea un mínimo local es examinar todos los puntos en una vecindad de x^* para

asegurarnos que en niguno de ellos se alcanza un menor valor de f. Cuando la función es suave, sin embargo, existen formas mucho mas eficientes y prácticas de identificar un mínimo local. En particular, si f es dos veces continuamente diferenciable podemos identificar x^* como mínimo local examinando el gradiente $\nabla f(x^*)$ y la matriz hessiana $\nabla^2 f(x^*)$.

El teorema de Taylor juega un papel central en la teoría de optimización, por lo que lo enunciamos enseguida, la demostración se puede consultar en [18].

Teorema. (Teorema de Taylor) *Supongamos que $f : \mathbb{R}^n \to \mathbb{R}$ continuamente diferenciable y $p \in \mathbb{R}^n$. Entonces tenemos que*

$$f(x+p) = f(x) + \nabla f(x+tp)^t p$$

para alguna $t \in (0,1)$. Además, si f es dos veces continuamente diferenciable, se tiene que

$$\nabla f(x+p) = \nabla f(x) + \int_0^1 \nabla^2 f(x+tp)p dt$$

y

$$f(x+p) = f(x) + \nabla f(x)^t p + \frac{1}{2} p^t \nabla^2 f(x+tp)p$$

para alguna $t \in (0,1)$.

Veremos ahora las condiciones necesarias de optimalidad suponiendo que x^* es un mínimo local derivaremos condiciones en $\nabla f(x^*)$ y $\nabla^2 f(x^*)$.

Teorema. (Condiciones necesarias de primer orden). *Si x^* es un mínimo local y f es continuamente diferenciable en una vecindad abierta de x^*, entonces*

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

Demostración.
Procederemos por contradicción, supongamos que $\nabla f(x^*) \neq 0$.

Sea $p = -\nabla f(x^*)$, se tiene que

$$p^t \nabla f(x^*) = -\|\nabla f(x^*)\|^2 < 0$$

Como ∇f es continua cerca de x^*, entonces existe un escalar $T > 0$ tal que

$$p^t \nabla f(x^* + tp) < 0 \qquad \forall t \in [0, T]$$

Entonces para cualquier $\bar{t} \in (0, T]$ por el teorema de Taylor tenemos que

$$f(x^* + \bar{t}p) = f(x^*) + \bar{t}p^t \nabla f(x^* + tp) \qquad \text{para alguna } t \in (0, \bar{t})$$

Entonces se tiene que

$$f(x^* + \bar{t}p) < f(x^*) \qquad \forall\, \bar{t} \in (0, T]$$

pero esto es una contradicción ya que supusimos que x^* es un mínimo local, por lo tanto la hipótesis es falsa y se tiene que

$$\nabla f(x^*) = 0 \qquad \blacksquare$$

Si $\nabla f(x^*) = 0$ decimos que x^* es un punto estacionario o un punto crítico de f. Entonces todo mínimo local debe ser un punto crítico.

Veremos ahora las condiciones necesarias de segundo orden para un mínimo local, antes introducimos la definción de una matriz positiva definida.

Definición. Una matriz $B \in \mathbb{R}^{n \times n}$ se dice que es positiva definida si

$$p^t B p > 0 \qquad \forall\, p \neq 0$$

y se dice que es positiva semidefinida si

$$p^t B p \geq 0 \qquad \forall\, p \neq 0$$

Teorema. (Condiciones necesarias de segundo orden). *Si x^* es un mínimo local de f y $\nabla^2 f$ es continua en una vecindad abierta de x^*, entonces $\nabla f(x^*) = 0$ y $\nabla^2 f(x^*)$ es positiva semidefinida.*

Demostración.

Del teorema anterior, tenemos que $\nabla f(x^*) = 0$. Demostraremos que $\nabla^2 f(x^*)$ es positiva semidefinida.

Procederemos por contradicción, supongamos que existe un vector p tal que $p^t \nabla^2 f(x^*) p < 0$, como $\nabla^2 f$ es continua cerca de x^*, entonces existe un escalar $T > 0$ tal que

$$p^t \nabla^2 f(x + tp) p < 0 \qquad \forall\, t \in [0, T]$$

Realizando la expasión en serie de Taylor alrededor de x^* tenemos que para toda $\bar{t} \in (0, T]$ y alguna $t \in (0, \bar{t})$ se tiene que

$$\begin{aligned} f(x^* + \bar{t}p) &= f(x^*) + \bar{t}p^t \nabla f(x^*) + \frac{1}{2}\bar{t}^2 p^t \nabla^2 f(x^* + tp)p \\ &< f(x^*) \end{aligned}$$

pero esto contradice el hecho de que x^* es un mínimo local. Por lo tanto se debe tener que

$$\nabla^2 f(x^*) \text{ positiva semidefinida} \quad \blacksquare$$

Tenemos entonces ya establecidas las condiciones necesarias de primer y segundo orden, veremos ahora el resultado que establece las condiciones suficientes de segundo orden para un mínimo local.

Teorema. (Condiciones suficientes de segundo orden). *Supongamos que $\nabla^2 f$ es continua en una vecindad abierta de x^* y que $\nabla f(x^*) = 0$ y $\nabla^2 f(x^*)$ es positiva definida. Entonces x^* es un mínimo local estricto de f.*

Demostración.
Como la matriz hessiana es continua y positiva definida en x^*, entonces podemos escoger un $r > 0$ tal que $\nabla^2 f$ sea positiva definida para toda x en la bola abierta de radio r, esto es, en $\mathcal{B} = \{z : \|z - x^*\| < r\}$.

Tomando cualquier vector $p \neq 0$ tal que $\|p\| < r$, tenemos que $x^* + p \in \mathcal{B}$ de tal forma que

$$\begin{aligned} f(x^* + p) &= f(x^*) + p^t \nabla f(x^*) + \frac{1}{2} p^t \nabla^2 f(z) p \\ &= f(x^*) + \frac{1}{2} p^t \nabla^2 f(z) p \end{aligned}$$

donde $z = x^* + tp$ para alguna $t \in (0, 1)$. Como $z \in \mathcal{B}$, tenemos que

$$p^t \nabla^2 f(x^*) p > 0$$

lo cual implica que

$$f(x^* + p) > f(x^*)$$

con lo cual se tiene que x^* es un mínimo local estricto. $\blacksquare$

Las condiciones suficientes de segundo orden son mas fuertes que las condiciones necesarias, en el sentido que garantizan que el mínimo local es un minimizador estricto. Sin embargo, las condiciones suficientes de segundo orden no son necesarias: un punto x^* puede ser un mínimo local estricto y no satisfacer las condiciones suficientes. Como ejemplo de esto, considere la función $f(x) = x^4$, donde el punto $x^* = 0$ es un mínimo local estricto en donde la hessiana no es positiva definida.

Entonces en un algoritmo de minimización esperamos encontrar un punto que satisfaga las condiciones de segundo orden para aproximar la solución buscada. Para algoritmos que solo utilizan información del gradiente, lo que podemos esperar es producir una sucesión tal que converja a un punto cítico de f, esto es,

$$\lim_{k\to\infty} \nabla f(x_k) = 0$$

Para algoritmos que utilicen información de la matriz Hessiana, se espera que se satisfagan las condiciones de segundo orden, lo cual puede obtenerse si

$$\lim_{k\to\infty} \inf \lambda_1(\nabla^2 f(x_k)) \geq 0$$

donde $\lambda_1(A)$ es el eigenvalor mas pequeño de la matriz A.

Tenemos entonces la caracterización de la solución, veremos ahora en qué consisten los algoritmos para encontrar una solución.

1.3 Métodos de Descenso

En los últimos años se ha visto un gran desarrollo en algoritmos para optimización sin restricciones para funciones suaves.

Todos los algoritmos de minimización requieren que el usuario proporcione un punto inicial, el cual usualmente se denota por x_0, si se tiene una idea de la solución del problema se escoge x_0 como una estimación de la solución, de manera que con un buen punto inicial esperamos que el algoritmo converja rápidamente. Si no se tiene información adicional del problema, se puede escoger x_0 de forma arbitraria.

Iniciando en x_0 los algoritmos de optimización generan una sucesión $\{x_k\}_{k=0}^{\infty}$ que terminará cuando ya no se avance en la sucesión o cuando algún iterando satisfaga las condiciones de mínimo local con alguna precisión. Para decidir cómo moverse de una iteración a la siguiente, los algoritmos utilizan información de la función en x_k y posiblemente de iteraciones anteriores $x_0, x_1, \ldots, x_{k-1}$, con esta información se encuentra un nuevo iterando x_{k+1} donde el valor de la función sea menor que en x_k.

Los métodos de descenso generan una sucesión $\{x_k\}$ de aproximaciones a la solución de manera que

$$f(x_{k+1}) < f(x_k)$$

para toda k, esto es, se acepta la nueva iteración x_{k+1} si produce una reducción en el valor de la función objetivo, de tal forma que el valor de la función en la iteración actual es menor que el valor de la función en iteraciones anteriores. Una de las características de los métodos de descenso es que en la iteración k se debe avanzar sobre una dirección p la cual tiene que ser una dirección de descenso, esto es, una dirección sobre la que la función, localmente, decrezca.

Defincion. Una dirección p se dice de descenso en el punto x_k si la derivada direccional de f en x_k sobre la dirección p es negativa, esto es, si satisface

$$p^t \nabla f(x_k) < 0$$

La condición de descenso por sí sola no es suficiente para garantizar que la iteración $\{x_k\}$ se aproxime a un mínimo local. Se requieren condiciones mas fuertes para forzar a la sucesión a entrar en una vecindad del mínimo local. Una vez que la iteración entra en tal vecindad, los métodos de descenso usualmente convergen rápidamente al mínimo local.

Existen dos estrategias fundamentales para llevar la iteración a una vecindad del mínimo local: búsqueda lineal y región de confianza. La mayoría de los algoritmos utilizan una de ellas.

1.3.1 Búsqueda lineal y región de confianza

En la estrategia de búsqueda lineal el algoritmo escoge una dirección de descenso, p_k, llamada dirección de búsqueda, y busca a lo largo de esa dirección, partiendo de x_k, un nuevo elemento de la iteración donde el valor de la función decrezca.

La distancia a recorrer sobre la dirección p_k puede encontrarse resolviendo aproximadamente el problema unidimensional de encontrar un $\alpha^* \in \mathbb{R}$ tal que

$$\alpha^* = \arg\min_{\alpha>0} f(x_k + \alpha p_k)$$

Si este problema se resuelve exactamente, se tendría el máximo beneficio sobre la dirección p_k, sin embargo resolver el problema exactamente es muy costoso en la

práctica, es por esto que se utilizan soluciones aproximadas al mínimo de esta función por medio de un algoritmo que genera un número limitado de puntos de prueba, hasta que se encuentra un α cercano al mínimo exacto. Una vez que se tiene la nueva iteración x_{k+1} se genera una nueva dirección de descenso y se repite el proceso.

En la segunda estrategia, conocida como región de confianza, se utiliza información local de f para construir un modelo de la función, Ψ_k, de manera que el comportamiento de Ψ_k cerca de x_k sea similar al comportamiento de f. Entonces minimizando Ψ_k nos acercaremos al mínimo de f. Como el modelo Ψ_k puede no ser una muy buena aproximación de f cuando no se está cerca de x_k, se restringe la búsqueda del mínimo de Ψ_k a alguna región alrededor de x_k, usualmente a una bola con centro en x_k y radio Δ_k, a Δ_k se le conoce como el radio de la región de confianza. En otras palabras, generamos el paso p_k como

$$p_k = \arg \min_p \Psi_k(x_k + p) \tag{1.1}$$

donde $x_k + p$ está dentro de la región de confianza. Si con p_k no se produce una reducción suficiente en f, quiere decir que la región de confianza es muy grande, se disminuye Δ_k y se vuelve a resolver (1.1).

El modelo Ψ_k usualmente es una función cuadrática de la forma

$$\Psi_k(x_k + p) = f(x_k) + p^t \nabla f_(x_k) + \frac{1}{2} p^t B_k p$$

donde $\nabla f(x_k)$ es el gradiente de la función y B_k es una matriz que aproxima la matriz hessiana en x_k.

Entonces la estrategia de región de confianza, determina primero cuánto moverse y después la dirección en la que hay que hacerlo, mientras que la búsqueda lineal primero se determina la dirección sobre la que hay que avanzar y despúes cuánto moverse sobre esa dirección. La estrategia de búsqueda lineal se trata con detalle en el capítulo 3, mientras que la región de confianza en el capítulo 5.

Veremos ahora, los métodos mas conocidos de minimización sin restricciones.

1.4 Método de descenso más rápido

El método mas conocido de minimización sin restricciones es el método de descenso más rápido. Se sabe que localmente una función decrece más rápidamente en la dirección de $-\nabla f(x)$, entonces parece razonable generar un método que utilice búsqueda lineal, de manera que en cada paso se genere la dirección de descenso como

$$p_k = -\nabla f(x_k)$$

La iteración en este caso tiene la forma

$$x_{k+1} = x_k + \alpha_k p_k$$

Una de las ventajas de este método es que solo requiere el cálculo de la primera derivada de la función, sin embargo, se ha visto que su desarrollo en la práctica no es satisfactorio, la convergencia es demasiado lenta y en ocasiones presenta un comportamiento oscilatorio.

Otro de los métodos mas conocidos, y quizá el mas importante, de minimización sin restricciones es el método de Newton, enseguida presentamos en qué consiste este método.

1.5 Método de Newton

Supongamos que tenemos $f : \mathbb{R}^n \to \mathbb{R}$ una función dos veces continuamente diferenciable, el método de Newton para minimizar una función puede derivarse suponiendo que tenemos una aproximación x_k al mínimo local de f y que en una vecindad de x_k se satisface

$$f(x_k + w) \approx f(x_k) + \Psi(w)$$

donde

$$\Psi(w) = \nabla f(x_k)^t w + \frac{1}{2} w^t \nabla^2 f(x_k) w$$

es el modelo local cuadrático de f en x_k. Si esta aproximación es buena, entonces una mejor aproximación se obtiene con

$$x_{k+1} = x_k + s_k$$

donde el paso s_k es el mínimo de Ψ_k. Por las condiciones de un mínimo local, s_k debe ser tal que

$$\nabla \Psi_k(s_k) = \nabla f(x_k) + \nabla^2 f(x_k) s_k = 0$$

Entonces el método de Newton parte de una aproximación inicial x_0 y genera una sucesión de la forma

$$x_{k+1} = x_k - \nabla^2 f(x_k)^{-1} \nabla f(x_k) \tag{1.2}$$

Esta iteración es de hecho el método de Newton para sistemas de ecuaciones no lineales, donde se resuelve el sistema

$$\nabla f(x) = 0$$

Entonces si consideramos $F(x) = \nabla f(x)$ podemos enunciar el método de Newton en una forma mas general.

Consideremos el problema de resolver el sistema de n ecuaciones con n incógnitas

$$F(x) = 0$$

donde $F : \mathbb{R}^n \to \mathbb{R}^n$. El método de Newton para este problema puede obtenerse suponiendo que se tiene una aproximación x_k a la solución del sistema de ecuaciones no lineales y que en una vecindad de x_k se tiene la aproximación

$$F(x_k + w) \approx L_k(w) = F(x_k) + F'(x_k)w$$

donde $F'(x_k)$ es la matriz Jacobiana de F en x_k. Entonces se construye la siguiente aproximación como

$$x_{k+1} = x_k + s_k$$

donde se pide que el paso s_k satisfaga el sistema de ecuaciones lineales

$$L_k(w) = 0$$

Entonces el método de Newton parte de una aproximación inicial de x_0 y genera la sucesión

$$x_{k+1} = x_k - F'(x_k)^{-1}F(x_k) \tag{1.3}$$

Note que (1.2) es un caso especial de (1.3) cuando se considera $F(x) = \nabla f(x)$. Debido a estas relaciones el comportamiento local del método de Newton usualmente se estudia considerando la iteración (1.3).

Uno de los principales resultados del método de Newton es su convergencia local para cualquier punto inicial que se encuentre en una vecindad del mínimo local.

Teorema. *Sea $F : \mathbb{R}^n \to \mathbb{R}^n$ un mapeo continuo y diferenciable definido en un conjunto abierto D, suponga que $F(x^*) = 0$ para algún $x^* \in D$ y que $F'(x^*)$ es no singular. Entonces existe un conjunto abierto S tal que para cualquier $x_0 \in S$ la iteración de Newton (1.3) está bien definida, permanece en S y converge a x^*.*

Demostración.
Sea α una constante fija en $(0, 1)$. Como F' es continua en x^* y $F'(x^*)$ es no singular, existe una bola abierta $S = \{x : \|x - x^*\| < \epsilon\}$ y una constante positiva μ tal que

$$\|F'(x)^{-1}\| \leq \mu, \qquad \|F'(y) - F'(x)\| \leq \frac{\alpha}{\mu}$$

para toda $x, y \in S$. Supongamos que $x_k \in S$, como

$$x_{k+1} = x_k - F'(x_k)^{-1}F(x_k)$$

y $F(x^*) = 0$, entonces tenemos que

$$\begin{aligned} x_{k+1} - x^* &= x_k - x^* - F'(x_k)^{-1}F(x_k) \\ &= x_k - x^* - F'(x_k)^{-1}[F(x_k) - F(x^*)] \\ &= -F'(x_k)^{-1}[F(x_k) - F(x^*) - F'(x_k)(x_k - x^*)] \end{aligned}$$

y entonces, se tiene que

$$\begin{aligned} \|x_{k+1} - x^*\| &\leq \|F'(x_k)^{-1}\|\|F(x_k) - F(x^*) - F'(x_k)(x_k - x^*)\| \\ &\leq \mu\|F(x_k) - F(x^*) - F'(x_k)(x_k - x^*)\| \end{aligned}$$

Ahora, por el teorema fundamental del cálculo, tenemos que

$$F(x_k) - F(x^*) - F'(x_k)(x_k - x^*) = \int_0^1 [F'(x^* + t(x_k - x^*) - F'(x_k)](x_k - x^*)dt$$

y entonces se cumple que

$$\|x_{k+1} - x^*\| \leq \mu \left\{ \max_{0 \leq t \leq 1} \|F'(x^* + t(x_k - x^*) - F'(x_k)\| \right\} \|(x_k - x^*)\| \quad (1.4)$$

Entonces, se tiene que

$$\begin{aligned} \|x_{k+1} - x^*\| &\leq \mu\frac{\alpha}{\mu}\|x_k - x^*\| \\ &\leq \alpha\|x_k - x^*\| \end{aligned}$$

Como $\alpha < 1$, entonces tenemos que si $x_0 \in S$, entonces $x_k \in S$ para $k = 1, 2, \ldots$, y la sucesión $\{x_k\}$ converge a x^*. ■

Entonces tenemos que S es el *dominio de atracción*, esto es, la región donde se garantiza que la iteración de Newton converge. Desafortunadamente para muchos problemas este dominio de atracción es muy pequeño y es necesario utilizar estrategias de globalización de manera que hagan que la iteración en algún momento entre en S, de forma que una vez ahi, la convergencia está garantizada.

Uno de los principales atractivos del método de Newton es su velocidad de convergencia. Si la función es Lipschitz continua entonces en una vecindad del mínimo local, el método converge cuadráticamente, esto es, existe $\beta > 0$ tal que

$$\|x_{k+1} - x^*\| \leq \beta \|x_k - x^*\|^2$$

La convergencia cuadrática implica que el número de cifras significativas de x_k al aproximar a x^* se duplica en cada iteración. Establecemos la convergencia cuadrática del método de Newton en el siguiente resultado.

Teorema. *Sea $F : \mathbb{R}^n \to \mathbb{R}^n$ con las hipótesis del teorema anterior. Entonces la sucesión $\{x_k\}$ producida por la iteración (1.3) converge superlinealmente a x^*. Además, si $\kappa > 0$ y*

$$\|F'(x) - F'(x^*)\| \leq \kappa \|x - x^*\|, \qquad x \in D \tag{1.5}$$

entones la sucesión converge cuadráticamente a x^*.

Demostración.

La convergencia de la sucesión se demostró en el teorema anterior, solo falta demostrar la velocidad de convergencia. Definamos

$$\beta_k = \mu \left\{ \max_{0 \leq t \leq 1} \|F'(x^* + t(x_k - x^*) - F'(x_k)\| \right\}$$

y supongamos que $x_0 \in S$ con μ y S definidos como en el teorema anterior. Las hipótesis de F' en x^* y la convergencia de la sucesión a x^* implican que $\{\beta_k\}$ converge a cero. Entonces, tenemos que se satisface

$$\|x_{k+1} - x^*\| \leq \mu \left\{ \max_{0 \leq t \leq 1} \|F'(x^* + t(x_k - x^*) - F'(x_k)\| \right\} \|(x_k - x^*)\|$$

de donde resulta

$$\|x_{k+1} - x^*\| \leq \beta_k \|x_k - x^*\|$$

lo cual demuestra la convergencia superlineal a x^*.

Además, si se satisface (1.5) entonces

$$\begin{aligned} \beta_k &\leq \mu\kappa(\|x^* + t(x_k - x^*) - x_k\|) \\ &\leq 2\mu\kappa\|x_k - x^*\| \end{aligned}$$

de donde tendremos que

$$\|x_{k+1} - x^*\| \leq 2\mu\kappa \|x_k - x^*\|^2$$

lo cual implica la convergencia cuadrática. ■

El método de Newton con alguna estrategia de globalización como búsqueda lineal o región de confianza opera muy bien en la práctica, pero tiene la desventaja de ser muy costoso. Como una alternativa recomendable, se proponen los métodos Quasi-Newton, los cuales tienen una velocidad de convergencia menor que la de Newton, superlineal, pero con la ventaja de ser económicos y confiables. Se recomiendan sobre todo en problemas densos.

1.6 Problemas de gran escala

En la actualidad se han desarrollado algoritmos muy eficientes para tratar problemas de optimización con un gran número de variables, llamados problemas de gran escala. La adaptación del método de Newton a este tipo de problemas, dio como resultado un método muy efectivo al trabajar con problemas de grandes dimensiones: el método de Newton truncado, éste consiste en obtener una aproximación a la dirección de Newton, resolviendo las ecuaciones de Newton, con algún método iterativo.

En muchas aplicaciones de la práctica en que se tienen que resolver problemas con un gran número de variables, la función objetivo puede presentar cierta estructura especial importante al momento de elegir el método de optimización a utilizar. Se ha observado que al trabajar con un gran número de variables muchas funciones presentan la estructura de separabilidad parcial, esto es, se pueden descomponer en suma de funciones, las cuales sólo dependen de algunas cuantas variables. Algunos métodos de optimización aprovechan la estructura de la función para obtener algoritmos mas económicos y eficientes. En el presente trabajo estudiaremos algunos métodos de optimización y veremos algunas aplicaciones a este tipo de funciones parcialmente separables.

Referencias

J. Dennis, R. Schnabel, (1983) *Numerical Methods for Unconstrained Optimization and nonlinear equations* Prentice–Hall.

R. Fletcher (1987). *Practical Methods of Optimization* John Wiley & Sons, 2nd.Edition.

C. Kelley (1999) *Iterative Methods for optimization* SIAM Publications.

J. Moré, D. Sorensen (1984). Newton's Method, *Studies in Numerical Analysis*, Ed. G.H. Golub, pp. 29–82.

J. Nocedal and J. Wright (1999). *Numerical Optimization*, Springer Series in Operations Research.

Capítulo 2

Métodos de Búsqueda Lineal

Queremos resolver el problema de optimización con un método de descenso, esto es, partiendo de un x_0 inicial generamos una sucesión $\{x_k\}$ con la propiedad de que $f(x_{k+1}) < f(x_k)$ y donde cada iteración sea una corrección de la iteración anterior, es decir,

$$x_{k+1} = x_k + s_k$$

con $s_k = \alpha_k p_k$, p_k una dirección de descenso generada con alguna estrategia general tal que $\nabla f_k^t p_k < 0$ y $\alpha_k \in \mathbb{R}$.

El problema que abordaremos en este capítulo es el de determinar un α_k adecuado en cada iteración, esto es, una vez determinada p_k, cuánto avanzar en esa dirección. α_k debe encontrarse de manera que se garantice convergencia global del método de descenso y de preferencia de manera económica. Usualmente se hace referencia a α_k como el tamaño de paso en la iteración k.

Al procedimiento para escoger α_k es lo que se le conoce como Búsqueda lineal y es usualmente un proceso iterativo finito.

Teóricamente α_k debe ser tal que

$$\alpha_k \approx \alpha^* = \arg\min_{\alpha} f(x_k + \alpha p_k)$$

de manera que la reducción en f esté garantizada en cada paso. A los métodos de búsqueda lineal que tratan de obtener una aproximación muy buena de α^* se les conoce como búsquedas lineales exactas. Sin embargo, estos métodos son muy costosos, en la práctica se trabaja lo que se conoce como búsquedas lineales inexactas, se busca que α_k satisfaga ciertas condiciones que garanticen la convergencia del método de descenso a un bajo costo, aunque no aproximen con gran exactitud a α^*. Enseguida veremos que condiciones deben satisfacer el tamaño de paso de manera que garantice convergencia global.

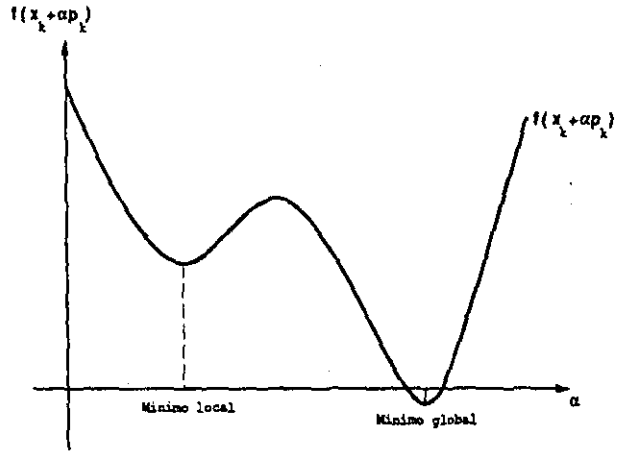


Figura 2.1: La longitud de paso ideal es el mínimo global

2.1 Condiciones de Wolfe–Powell

Los primeros algoritmos de búsqueda lineal que se desarrollaron trataban de realizar búsquedas lineales exactas, pero se dieron cuenta de que esto resultaba muy costoso sobre todo si se estaba lejos de la solución x^*, entonces surgieron una variedad de algoritmos donde se debilitaba mucho la búsqueda lineal, lo único que se pedía era que $f(x_{k+1}) < f(x_k)$, pero a veces se lograba un decrecimiento insignificante y se tenía como resultado un proceso demasiado lento. Fue entonces que se planteó el problema de establecer qué condiciones debe de satisfacer α_k de manera que se logre convergencia global y además que se encontrara mediante un proceso económico.

Goldstein en 1965 propuso dos condiciones sobre α_k, la primera para garantizar una reducción significativa de la función objetivo f, esto se mide mediante la desigualdad

$$f(x_k + \alpha p_k) \leq f(x_k) + \rho\alpha_k \nabla f(x_k)^t p_k \tag{2.1}$$

o equivalentemente,

$$f(x_k) - f(x_k + \alpha_k p_k) \geq -\rho\alpha_k \nabla f(x_k)^t p_k$$

con $\rho \in (0, 1/2)$ un parámetro fijo, esto es, que la reducción en f debe ser proporcional tanto el tamaño de paso α_k y a la derivada direccional $\nabla f(x_k)^t p_k$. La segunda condición de Goldstein evita que α_k esté muy cerca del origen, se pide que

$$f(x_k + \alpha_k p_k) \geq f(x_k) + (1-\rho)\alpha_k \nabla f(x_k)^t p_k \tag{2.2}$$

Las condiciones de Goldstein se ilustran en la figura 2.2.

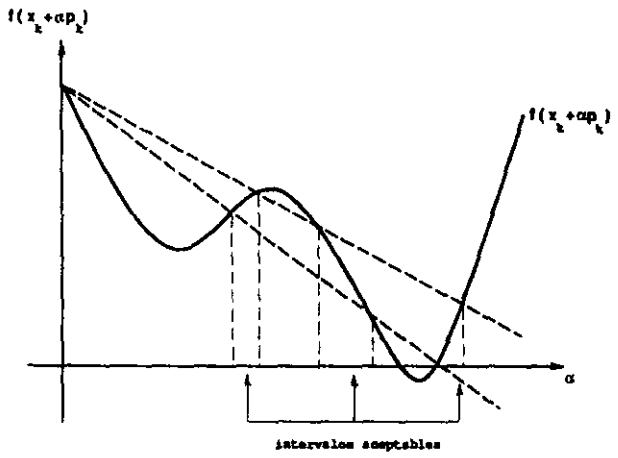


Figura 2.2: Condiciones de Goldstein

Una observación importante de las condiciones de Goldstein es que la segunda condición puede excluir el mínimo de f a lo largo de la línea $f(x_k + \alpha p_k)$ como se ilustra en la figura 2.2; es por esto que (2.2) usualmente se reemplaza por una condición sobre las pendientes de la curva unidimensional, esta condición se conoce como condición de curvatura y establece que

$$\nabla f(x_k + \alpha_k p_k)^t p_k \geq \sigma \nabla f(x_k)^t p_k \tag{2.3}$$

con $0 < \rho < \sigma < 1$, esto es, la pendiente en α_k es menor que la pendiente en el origen, esto tiene sentido ya que una pendiente menor y negativa indica que estamos reduciendo f significativamente a lo largo de la dirección p_k.

Las condiciones (2.1) y (2.3) se conocen como las condiciones de Wolfe–Powell. Las reescribimos para futuras referencias como

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \rho \alpha_k \nabla f_k^t p_k \tag{2.4}$$

$$\nabla f(x_k + \alpha_k p_k)^t p_k \geq \sigma \nabla f_k^t p_k \tag{2.5}$$

con $0 < \rho < \sigma < 1$.

Para trabajar con la función unidimensional $f(x_k + \alpha p_k)$ frecuentemente se utiliza la notación

$$\varphi(\alpha) = f(x_k + \alpha p_k)$$

en donde las condiciones de Wolfe–Powell se escriben como

$$\varphi(\alpha_k) \leq \varphi(0) + \rho\alpha_k\varphi\prime(0) \tag{2.6}$$
$$\varphi\prime(\alpha_k) \geq \sigma\varphi\prime(0) \tag{2.7}$$

con $0 < \rho < \sigma < 1$. Geométricamente (2.6) indica que $\varphi(\alpha_k)$ debe estar abajo de la llamada ρ–línea, la línea con ordenada en el origen $f(x_k)$ y pendiente negativa $\rho\varphi\prime(0)$, los intervalos donde esto se satisface se muestran en la figura 2.3, para ilustrar (2.7) se presenta la figura 2.4 y para los puntos que satisfacen ambas condiciones se muestran en la figura 2.5.

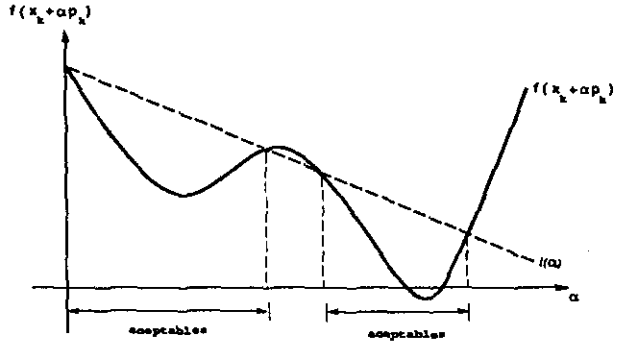


Figura 2.3: Condición de decrecimiento suficiente

Un punto que satisface (2.6) y (2.7) puede no encontrarse cerca del mínimo de $\varphi(\alpha)$, es por esto que se modificó un poco (2.7) para forzar a α_k a estar al menos en la frontera de una vecindad de un mínimo local o de un punto estacionario de φ, para lograr esto se tienen que satisfacer las condiciones fuertes de Wolfe–Powell

$$\varphi(\alpha_k) \leq \varphi(0) + \rho\alpha_k\varphi\prime(0) \tag{2.8}$$
$$|\varphi\prime(\alpha_k)| \geq \sigma|\varphi\prime(0)| \tag{2.9}$$

con $0 < \rho < \sigma < 1$. Aquí no se permiten pendientes muy positivas.

Para garantizar que bajo las condiciones de Wolfe–Powell se alcanza convergencia global, lo primero que tenemos que establecer es que existen puntos aceptables para estas condiciones. La restricción $\sigma > \rho$ es fundamental para asegurar que existen puntos aceptables y que pueden localizarse en un número finito de pasos, como se enuncia en el siguiente resultado.

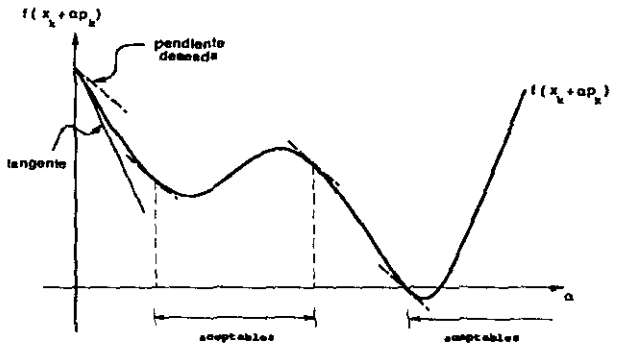


Figura 2.4: Condición de curvatura

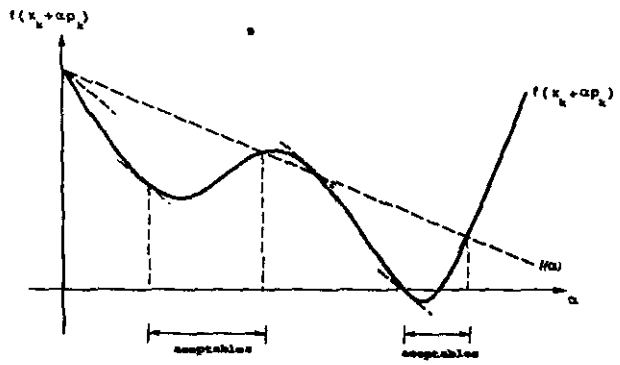


Figura 2.5: Longitudes de paso que satisfacen las condiciones de Wolfe–Powell

Lema. *Suponga $f : \mathbb{R}^n \to \mathbb{R}$ continua y diferenciable. Sea p_k una dirección de descenso en x_k y suponga que f está acotada por abajo en el rayo $\{x_k + \alpha p_k : \alpha > 0\}$. Entonces si $0 < \rho < \sigma < 1$ entonces existe un intervalo de longitudes de paso que satisfacen las condiciones de Wolfe–Powell y las condiciones fuertes de Wolfe–Powell.*

Demostración.
Como $\varphi(\alpha) = f(x_k + \alpha p_k)$ está acotada por abajo para toda $\alpha > 0$, y como $0 < \rho < 1$, la línea $l(\alpha) = f(x_k) + \alpha(\rho \nabla f_k^t p_k)$ debe intersectar la gráfica de φ al menos una vez. Sea $\alpha_f > 0$ el valor mas pequeño de intersección de α, esto es,

$$f(x_k + \alpha_f p_k) = f(x_k) + \alpha_f \rho \nabla f_k^t p_k \tag{2.10}$$

La condición de decrecimiento suficiente (2.4) se satisface para todos las longitudes de paso menores que α_f.

Por el teorema del valor medio existe $\alpha_\varepsilon \in (0, \alpha_f)$ tal que

$$f(x_k + \alpha_f p_k) - f(x_k) = \alpha_f \nabla f(x_k + \alpha_\varepsilon p_k)^t p_k \tag{2.11}$$

Combinando (2.10) y (2.11) tenemos que

$$\nabla f(x_k + \alpha_\varepsilon p_k)^t p_k = \rho \nabla f_k^t p_k > \sigma \nabla f_k^t p_k \tag{2.12}$$

ya que $\rho < \sigma$ y $\nabla f_k^t p_k < 0$. Entonces α_ε satisface las condiciones de Wolfe–Powell (2.4) y (2.5) y las desigualdades son estrictas en ambos casos. Entonces por la hipótesis de suavidad en f, existe un intervalo alrededor de α_ε en el cual se satisfacen las condiciones de Wolfe–Powell. Como el término en la izquierda de (2.12) es negativo, entonces las condiciones fuertes también se satisfacen, lo cual termina la demostración. ■

2.2 Convergencia de métodos con búsqueda lineal

Dentro de las propiedades mas importantes de las condiciones de Wolfe–Powell está que garantizan convergencia global para el método de descenso bajo ciertas condiciones de la dirección de búsqueda, esto ya que si la dirección p_k es cercanana ser ortogonal a la dirección de máximo descenso, entonces puede alcanzarse una reducción muy pequeña en la función. Esta posibilidad puede evitarse si la dirección p_k satisface

un criterio del ángulo, necesitamos que p_k y $-\nabla f(x_k)$ no estén cerca de ser ortogonales, esto es, si θ_k es el ángulo entre p_k y $-\nabla f(x_k)$, entonces existe una constante $\delta > 0$ tal que

$$\cos\theta_k = \frac{-\nabla f_k^t p_k}{\|\nabla f_k\|\|p_k\|} \geq \delta > 0 \qquad \text{para toda } k \tag{2.13}$$

Si esto se satisface, entonces tendremos convergencia global del método de descenso con búsqueda lineal. Establecemos esto en el siguiente resultado debido a Zoutendij.

Teorema. *Considere un método de descenso donde*

$$x_{k+1} = x_k + \alpha_k p_k$$

donde p_k es una dirección de descenso y α_k satisface las condiciones de Wolfe–Powell. Suponga que f está acotada por abajo en $\mathbb{R}^n$ y que f es continuamente diferenciable en un conjunto abierto D que contiene el conjunto de nivel $\mathcal{L} = \{x : f(x) \leq f(x_0)\}$, donde x_0 es el punto inicial de la iteración. Suponga también que el gradiente $\nabla f(x)$ es Lispchitz continuo en D, esto es, existe una constante L tal que

$$\|\nabla f(x) - \nabla f(\hat{x})\| \leq L\|x - \hat{x}\| \qquad \textit{para toda } x, \hat{x} \in D$$

y si se satisface (2.13), entonces se tiene que

$$\|\nabla f_k\| \longrightarrow 0$$

Demostración.
De la primera condición de Wolfe–Powell tenemos que

$$f(x_{k+1}) - f(x_k) \leq \rho\alpha_k \nabla f_k^t p_k \tag{2.14}$$

veremos que α_k está acotada inferiormente, en primer lugar, de la segunda condición

$$\nabla f_{k+1}^t p_k - \nabla f_k^t p_k \geq \sigma \nabla f_k^t p_k - \nabla f_k^t p_k$$

lo cual implica que

$$(\nabla f_{k+1} - \nabla f_k)^t p_k \geq (\sigma - 1)\nabla f_k^t p_k$$

esta ecuación junto con la condición de Lipschitz, la cual puede escribirse como

$$(\nabla f_{k+1} - \nabla f_k)^t p_k \leq \alpha_k L\|p_k\|^2$$

conducen a que

$$\alpha_k \geq \frac{\sigma - 1}{L\|p_k\|^2} \nabla f_k^t p_k$$

sustituyendo esto en la ecuación (2.14), tenemos que

$$f_{k+1} \leq f_k + \rho \frac{\sigma - 1}{L} \frac{(\nabla f_k^t p_k)^2}{\|p_k\|^2}$$

de la definición de $\cos\theta_k$, tenemos que

$$f_{k+1} \leq f_k + C \cos^2 \theta_k \|\nabla f_k\|^2 \tag{2.15}$$

donde $C = \rho \frac{\sigma - 1}{L}$, ahora sumando sobre todos los índices en (2.15), tenemos que

$$f_{k+1} \leq f_0 + C \sum_{j=1}^{k} \cos^2 \theta_j \|\nabla f_j\|^2 \tag{2.16}$$

Como f está acotada inferiormente, entonces

$$f_0 - f_{k+1} < M \qquad \forall k$$

donde M es una constante, entonces tomando límites en ambos lado de (2.16), tenemos que

$$\sum_{k=1}^{\infty} \cos^2 \theta_k \|\nabla f_k\|^2 < \infty$$

de donde se sigue que

$$\cos^2 \theta_k \|\nabla f_k\|^2 \longrightarrow 0$$

pero como se satisface (2.13), entonces se tiene que

$$\|\nabla f_k\| \longrightarrow 0$$

que es lo que queríamos demostrar. ■

Entonces en un método de descenso con búsqueda lineal que satisface las condiciones de Wolfe–Powell los gradientes convergen a cero siempre que las direcciones de búsqueda no estén cerca a ser ortogonales al gradiente. En particular en el método de descenso mas rápido la dirección de búsqueda hace un ángulo cero con la dirección

del gradiente negativo, entonces tendremos convergencia global si se utiliza búsqueda lineal con Wolfe–Powell.

Tenemos entonces ya establecidas las condiciones que garantizan convergencia global, nos enfocaremos ahora al problema práctico de encontrar un tamaño de paso adecuado en cada iteración, es decir, un tamaño de paso que satisfaga las condiciones de Wolfe–Powell.

2.3 Algoritmos para la búsqueda lineal

Los algoritmos de búsqueda lineal usualmente son procesos iterativos que constan de dos fases: la fase de bracketing o localización en la cual se busca encontrar un intervalo que contenga puntos aceptables para las condiciones de Wolfe–Powell y la fase de seccionamiento, en donde el intervalo es dividido para escoger un punto aceptable dentro de dicho intervalo.

Existen varios algoritmos para la búsqueda lineal, de los mas conocidos podemos mencionar el debido a Moré y Thuente [34] y el de Fletcher [21]. En esta sección mencionaremos las ideas del algoritmo de Fletcher.

Usaremos solo en esta sección la notación

$$f(\alpha) = f(x_k + \alpha p_k)$$

Entonces queremos encontrar un α_k que satisfaga las condiciones de Wolfe–Powell.

$$f(\alpha) \leq f(0) + \rho\alpha f'(0) \tag{2.17}$$

$$|f'(\alpha)| \leq -\sigma f'(0) \tag{2.18}$$

$0 < \rho < \sigma < 1$.

Describiremos un algoritmo particular de búsqueda lineal que satisface estas condiciones haciendo un número finito (usualmente pequeño) de evaluaciones de $f(\alpha)$ y $f'(\alpha)$.

Si la función f está acotada inferiormente, entonces existe un intervalo con puntos aceptables como ya se ha demostrado, para evitar que el algoritmo fracase en caso de que la función no esté acotada, supondremos que el usuario puede proporcionar una cota inferior $\bar{f}$ de $f(\alpha)$, mas precisamente se supone que el usuario está preparado para aceptar cualquier valor de $f(\alpha)$ para el cual $f(\alpha) \leq \bar{f}$. Una consecuencia de esta suposición es que la búsqueda lineal se puede restringir al intervalo $(0, \mu]$, donde

$$\mu = \frac{\bar{f} - f(0)}{\rho f'(0)}$$

es el punto en el cual la ρ–línea intersecta la línea $f = \bar{f}$.

Describiremos ahora con detalle las dos fases del algoritmo de búsqueda lineal.

2.3.1 Localización

En la primera fase del algoritmo, la localización, queremos encontrar un intervalo o bracket con puntos aceptables. En esta fase la iteración α_i se mueve hacia la derecha con grandes saltos hasta que detecta o $f \leq \bar{f}$ o se localiza un corchete con un intervalo de puntos aceptables. Inicialmente $\alpha_0 = 0$, α_1 es dado $(0 < \alpha_1 \leq \mu)$ y entonces el algoritmo de localización se puede describir de la siguiente manera.

Si $f(\alpha_1) > f(0) + \alpha_1 \rho f'(0)$, entonces estamos en la siguiente situación, ilustrada en la figura 2.6

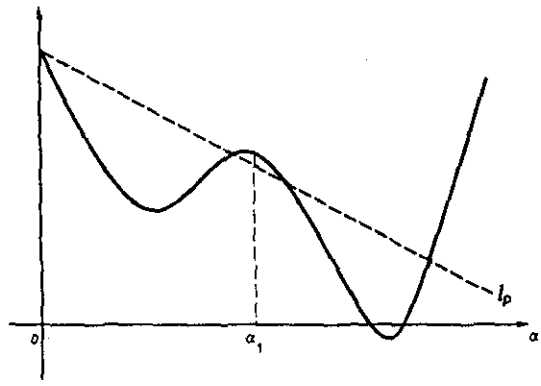


Figura 2.6: El punto prueba está arriba de la ρ–línea

entonces $[0, \alpha_1]$ es un bracket con puntos aceptables.

Si $f(\alpha_1) > f(0)$, tenemos una gráfica similar a la figura (2.6) y entonces $[0, \alpha_1]$ es un bracket con puntos aceptables.

Si no pasa esto, es decir, $f(\alpha_1)$ no está arriba de la ρ–línea y $f(\alpha_1) < f(0)$, entonces tenemos una de las situaciones como las que se ilustran en la figura 2.7

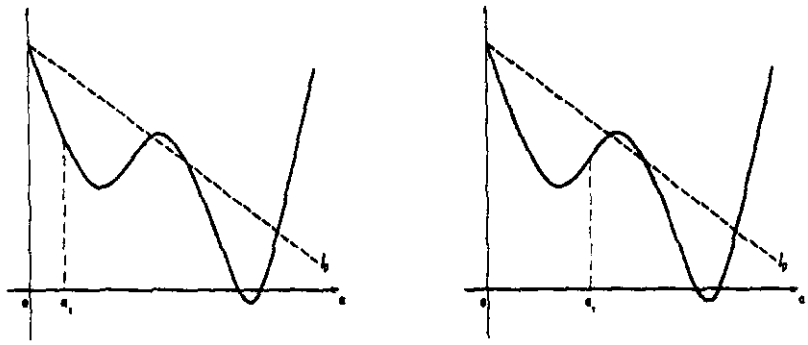


Figura 2.7: a)$f'(\alpha_1) < 0$ b)$f'(\alpha_1) > 0$

En este caso calculamos $f'(\alpha_1)$, si $|f'(\alpha_1)| \leq -\sigma f'(0)$, entonces α_1 es un punto aceptable para las condiciones de Wolfe–Powell y salimos del algoritmo de búsqueda lineal sin hacer seccionamiento.

Si no se satisface la segunda condición de Wolfe–Powell, pero si $f'(\alpha_1) > 0$, como en la figura (2.7) b), entonces $[0, \alpha_1]$ es un bracket con puntos aceptables ya que $f'(0) < 0$ y como la función es continua, en algún momento llego a un mínimo en el intervalo $[0, \alpha_1]$.

Si $f'(\alpha_1) < 0$, entonces lo que hay que hacer es movernos hacia la derecha, quiere decir que no hemos llegado al mínimo de la línea, que la función sigue bajando, como se muestra en la figura (2.7) a). El movimiento hacia la derecha puede hacerse de varias maneras, una forma de hacerlo es calcular el polinomio de interpolación con los datos con los que contamos hasta el momento, esto es, con 0, $f(0)$, $f'(0)$, α_1, $f(\alpha_1)$, $f'(\alpha_1)$ podemos determinar el polinomio cuadrático o cúbico que interpola estos datos, encontrar el mínimo α_2, de este polinomio y ese será nuestro nuevo punto en la sucesión, el cual habremos de probar si es un punto aceptable. Pero para garantizar convergencia rápida, el nuevo punto no debe estar muy cercano a 0 y debemos asegurarnos que avance lo suficiente, lo que se hace en la práctica es pedir que avance a lo más τ_1 veces α_1 a partir de α_1, y a la vez se pide que el nuevo punto no exceda μ. Esto es,

$$\alpha_2 \in [2\alpha_1, \min(\mu, \alpha_1 + \tau_1\alpha_1)]$$

donde $\tau_1 > 1$ es un factor fijo por el cual se mide cuánto se permite avanzar, típicamente $\tau_1 = 9$.

Entonces en el caso que estamos es

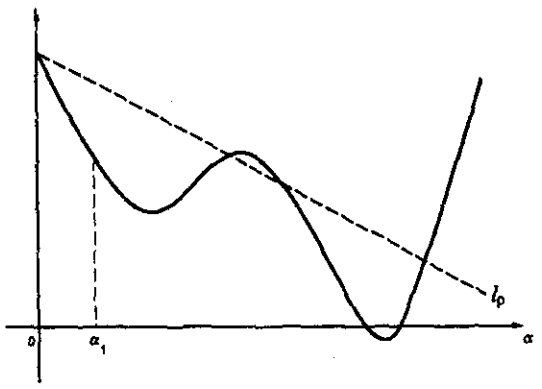


Figura 2.8: El punto α_1 tiene que moverse hacia la derecha

esto es, $f(\alpha_1)$ está debajo de la línea ρ y con $f'(\alpha_1) < 0$. Calculando α_2 por el procedimiento anterior, entonces volvemos a caer en alguno de los dos casos anteriores. Primer caso. Si $f(\alpha_2) > f(0) + \rho\alpha_2 f'(0)$, como se muestra en la figura 2.9,

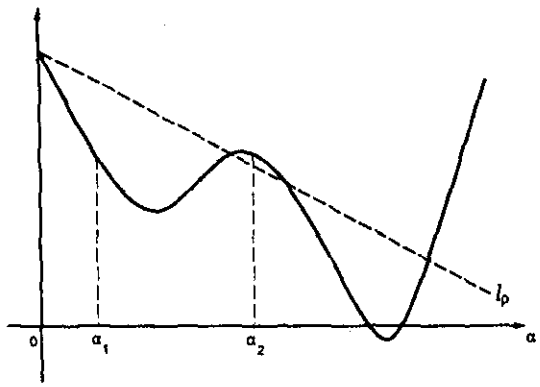


Figura 2.9: $f(\alpha_2)$ arriba de la ρ–línea

entonces $[\alpha_1, \alpha_2]$ es un bracket con puntos aceptables.

Segundo caso. Si $f(\alpha_2) > f(\alpha_1)$, gráficamente,

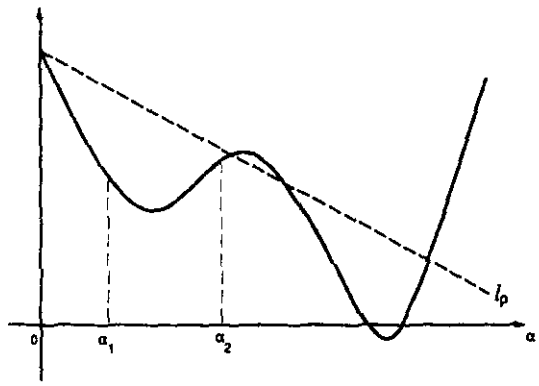


Figura 2.10: $f(\alpha_2) > f(\alpha_1)$ y por debajo de la ρ–línea

entonces $[\alpha_1, \alpha_2]$ es un bracket con puntos aceptables.

Si no ocurre esto, entonces calculamos $f'(\alpha_2)$. Si $|f'(\alpha_2)| < -\sigma f'(0)$, entonces salimos, con α_2 ya que es un punto que satisface las dos condiciones de Wolfe–Powell.

Si no pasa esto, entonces estamos en una de las dos condiciones siguientes, que están ilustradas en la figura 2.11

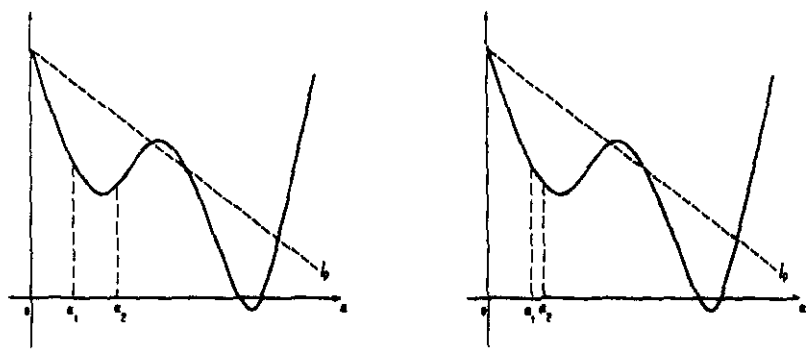


Figura 2.11: a)$f(\alpha_1) > f(\alpha_2)$, $f'(\alpha_2) > 0$ b) $f(\alpha_1) > f(\alpha_2)$, $f'(\alpha_2) < 0$

Si $f(\alpha_1) > f(\alpha_2)$ y $f'(\alpha_2) > 0$ como en **a)**, entonces $[\alpha_2, \alpha_1]$ es un bracket con puntos aceptables. Si $f(\alpha_1) > f(\alpha_2)$ y $f'(\alpha_2) < 0$ como en **b)**, entonces tenemos que avanzar mas hacia la derecha, esto lo podemos hacer calculando el mínimo del polinomio de interpolación ahora con los datos del valor de la función y la derivada

en α_1 y α_2. El nuevo punto no debe estar muy cerca de α_1 y debe avazar lo suficiente, se tiene que

$$\alpha_3 \in [2\alpha_2 - \alpha_1, \min(\mu, \alpha_2 + \tau_1(\alpha_2 - \alpha_1))]$$

y volvemos a iniciar el proceso.

Entonces el algoritmo para la fase de localización es el siguiente:

Algoritmo: Localización
Entrada: $\alpha_1, f(0), f'(0), \mu, \rho, \sigma, \bar{f}$

Salida: α_i, a_i, b_i
for $i = 1, 2, \ldots,$ do

begin evaluar $f(\alpha_i)$
if $f(\alpha_i) \leq \bar{f}$ then terminar
if $f(\alpha_i) > f(0) + \alpha_i \rho f'(0)$ o $f(\alpha_i) \geq f(\alpha_{i-1})$
then begin $a_i = \alpha_{i-1}$; $b_i = \alpha_i$; terminar B, end;
evaluar $f'(\alpha_i)$
if $|f'(\alpha_i)| \leq -\sigma f'(0)$ then terminar
if $f'(\alpha_i) \geq 0$
then begin $a_i = \alpha_i$, $b_i = \alpha_{i-1}$; terminar B, end;
if $\mu \leq 2\alpha_i - \alpha_{i-1}$
then $\alpha_{i+1} = \mu$
else escoger $\alpha_{i+1} \in [2\alpha_i - \alpha_{i-1}, \min(\mu, \alpha_i + \tau_1(\alpha_i - \alpha_{i-1}))]$, end;
end

En el algoritmo terminar B se utiliza para indicar terminación solamente la fase de localización, mientras que terminar indica terminación de la búsqueda lineal con un punto aceptable α_i o con un punto en el cual $f(\alpha_i) \leq \bar{f}$. Si se termina por un terminar B entonces tenemos un corchete $[a_i, b_i]$ (es conveniente permitir $a_i < b_i$ o $b_i < a_i$ en esta notación de corchete) que está en alguno de los siguientes casos que se ilustran en la figura 2.12.

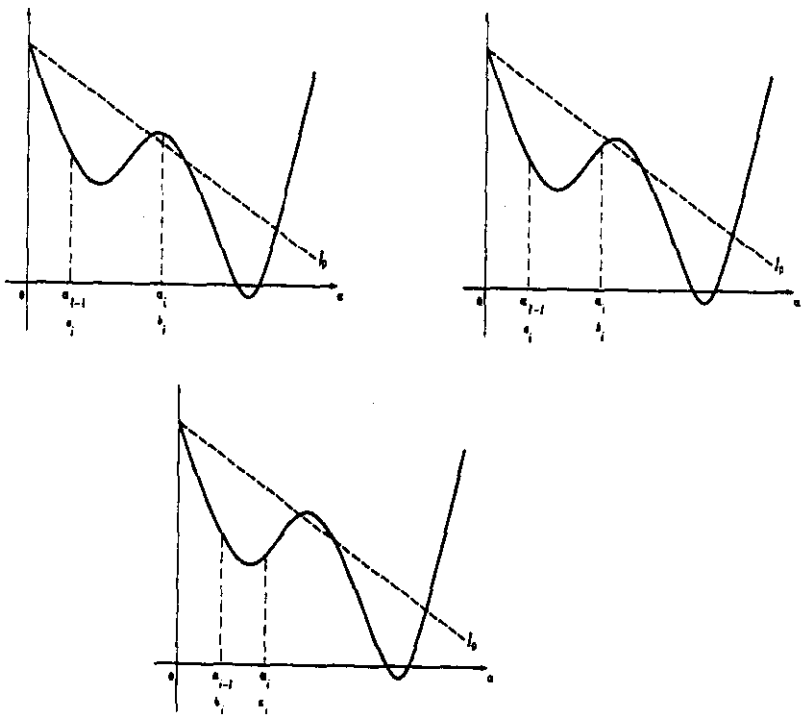


Figura 2.12: Corchetes al salir de la fase de Localización

En cualquiera de estos casos se tiene que el corchete tiene las siguientes propiedades:

i) a_i es el punto donde se tiene el menor valor de f que satisface la primera condición de Wolfe–Powell (2.17).

ii) $f'(a_i)$ ha sido calculada y satisface

$$(b_i - a_i)f'(a_i) < 0$$

pero no satisface (2.18).

iii) b_i satisface $f(b_i) > f(0) + b_i\rho f'(0)$ o $f(b_i) \geq f(a_i)$ o ambas.

Para tales corchetes se satisface el siguiente resultado.

Lema *Si $\sigma \geq \rho$, un corchete que satisface las propiedades anteriores contiene un intervalo de puntos aceptables para las condiciones de Wolfe–Powell.*

Demostración.

Si $b_i > a_i$, considere la línea L que pasa por $(a_i, f(a_i))$ y que tenga pendiente $\rho f'(0)$, esto es, paralela a la ρ–línea, como se muestra en la figura 2.13

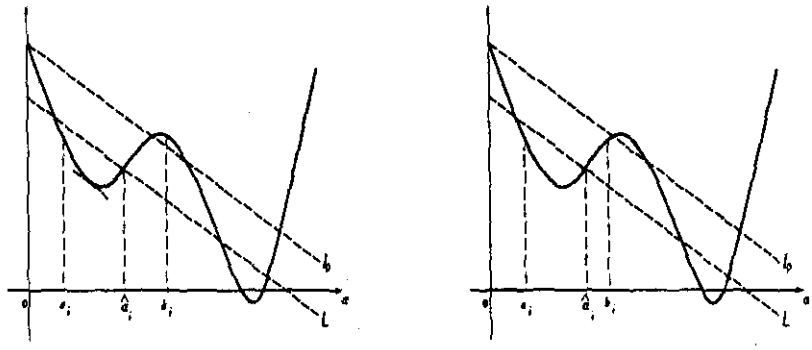


Figura 2.13: La línea L en el caso que $a_i < b_i$

Sea $\hat{\alpha}_i$ el punto en (a_i, b_i) más cercano a a_i en el cual la gráfica de $f(\alpha)$ intersecta L. La existencia de $\hat{\alpha}_i$ se tiene ya que f es continua y dado que $f(b_i) > f(a_i)$ o $f(b_i) > f(0) + \rho b_i f'(0)$. Como $b_i - a_i > 0$, y tenemos que $(b_i - a_i)f'(a_i) < 0$, entonces esto implica que

$$f'(a_i) < 0$$

Utilizando el teorema del valor medio, tenemos que existe $\tilde{\alpha}_i$ tal que $\tilde{\alpha}_i \in (a_i, \hat{\alpha}_i)$ tal que

$$f'(\tilde{\alpha}_i) = \frac{f(\hat{\alpha}_i) - f(a_i)}{\hat{\alpha}_i - a_i} = \rho f'(0) > \sigma f'(0)$$

ya que estamos suponiendo que $\sigma \geq \rho$, y entonces $\tilde{\alpha}_i$ satisface la segunda condición de Wolfe–Powell y ya teníamos que satisface también la primera condición, entonces por continuidad de f tenemos que existe un intervalo de puntos aceptables.

Si $b_i < a_i$ entonces estamos en el caso que se ilustra en la figura 2.14.

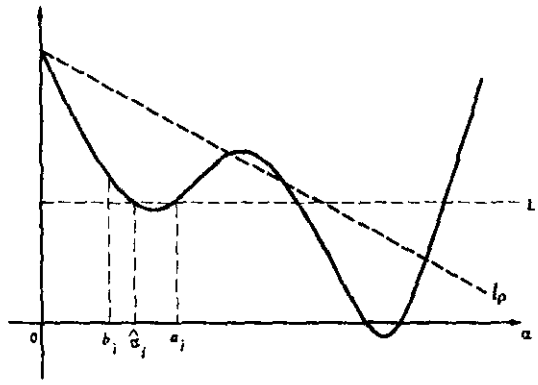


Figura 2.14: La línea L en el caso que $b_i < a_i$

Consideremos la línea que pasa por $(a_i, f(a_i))$ con pendiente cero y de nuevo sea $\hat{\alpha}_i$ el punto más cercano a a_i donde se intersecta la línea L con $f(\alpha)$, este punto existe por las propiedades ii) y ii), entonces tenemos que

$$f(b_i) > f(a_i), \quad (b_i - a_i) < 0, \quad (b_i - a_i)f'(a_i) < 0$$

$$\Rightarrow f'(a_i) > 0$$

entonces por el Teorema del Valor Medio existe $\tilde{\alpha}_i \in (\hat{\alpha}_i, a_i)$ tal que

$$f'(\tilde{\alpha}_i) = \frac{f(\hat{\alpha}_i) - f(a_i)}{\hat{\alpha}_i - a_i} = 0 > \rho f'(0) > \sigma f'(0)$$

y entonces $\tilde{\alpha}_i$ satisface la segunda condición, como ya teníamos que satisfacía la primera condición, entonces es un punto aceptable. De nuevo, por la continuidad de f tenemos un intervalo con puntos aceptables.

Esto termina la demostración. ■

Este Lema muestra que la fase de localización ha alcanzado un corchete con puntos aceptables. Ahora viene la fase del seccionamiento.

2.3.2 Seccionamiento

En la fase de seccionamiento escogeremos un punto dentro del intervalo de puntos aceptables obtenido en la localización. Para esto se genera un sucesión de corchetes

$[a_j, b_j]$ para $j = i, i+1, \ldots$ cuyas longitudes tienden a cero. En cada iteración se escoge un nuevo punto $\alpha_j \in [a_j, b_j]$ y el siguiente corchete es $[a_j, \alpha_j]$, $[\alpha_j, a_l]$, o $[\alpha_j, b_j]$, la elección se hace de manera que que las propiedades i), ii), iii) se sigan satisfaciendo. La fase del seccionamiento termina cuando el punto actual α_j satisface las condiciones de Wolfe–Powell.

Entonces al salir de la localización tenemos un corchete de alguna de las siguientes formas que se ilustran en la figura 2.15

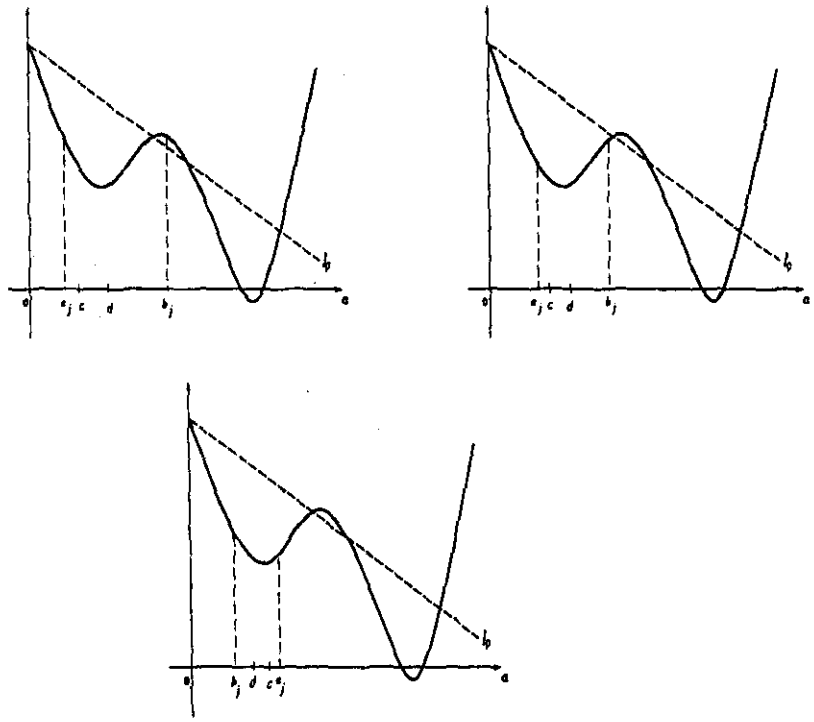


Figura 2.15: Corchetes al salir de la fase de Localización

Escogemos $\alpha_j \in [a_j + \tau_2(b_j - a_j), b_j - \tau_3(b_j - a_j)] = [c, d]$ con $0 < \tau_2 < \tau_3 \leq 1/2$, esto garantiza que la longitud del intervalo disminuya en cada iteración. Enseguida calculamos $f(\alpha_j)$ y generamos el nuevo corchete:

Si $f(\alpha_j) > f(0) + \rho\alpha_j f'(0)$ entonces tenemos que en el nuevo corchete

$$a_{j+1} = a_j, \quad b_{j+1} = \alpha_j$$

Si $f(\alpha_j) > f(a_j)$, entonces en el nuevo corchete

$$a_{j+1} = a_j, \quad b_{j+1} = \alpha_j$$

Si no pasa esto, es decir, si $f(a_j) > f(\alpha_j)$ entonces calculamos $f'(\alpha_j)$.

Si $f'(\alpha_j) < \sigma f'(0)$ salimos de la búsqueda lineal y α_j es un punto que satisface las dos condiciones de Wolfe–Powell.

Si no se satisface la segunda condición, entonces

$$a_{j+1} = a_j$$

y veamos cómo escoger b_{j+1}.

Lo vamos a hacer por casos, de acuerdo a los corchetes que aparecen en la figura 2.15. Los casos de las dos primeras figuras son similares, entonces se tratarán como uno solo, en este caso puede presentarse una de las siguientes situaciones de la figura 2.16

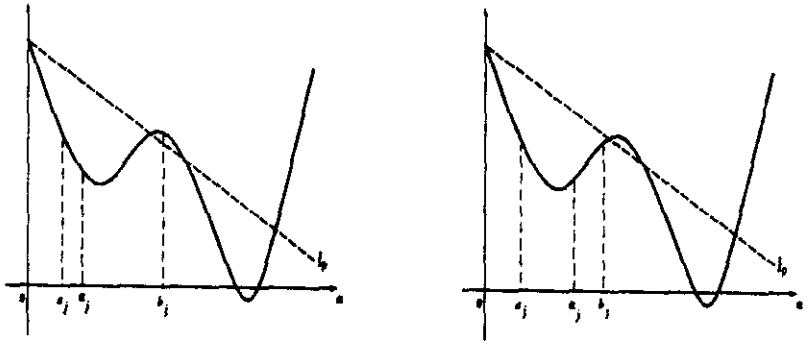


Figura 2.16: a)$(b_j - a_j)f'(\alpha_j) < 0$ b) $(b_j - a_j)f'(\alpha_j) > 0$

En el caso a) se tiene que en el nuevo corchete

$$a_{j+1} = \alpha_j, \quad b_{j+1} = b_j$$

En el caso b) se tiene que en el nuevo corchete

$$a_{j+1} = \alpha_j, \quad b_{j+1} = a_j$$

Para el caso del tercer corchete, se puede presentar una de las siguientes situaciones de la figura 2.17.

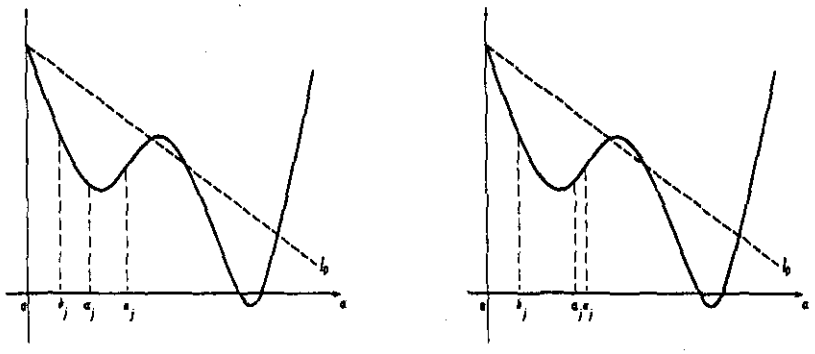


Figura 2.17: a)$(b_j - a_j)f'(\alpha_j) > 0$ b) $(b_j - a_j)f'(\alpha_j) < 0$

En el caso a) se tiene que en el nuevo corchete

$$a_{j+1} = \alpha_j, \quad b_{j+1} = a_j$$

En el caso b) se tiene que en el nuevo corchete

$$a_{j+1} = \alpha_j, \quad b_{j+1} = b_j$$

Entonces si encontramos $a_{j+1} = \alpha_j$ nos fijamos en el signo de $(b_j - a_j)f'(\alpha_j)$

Si $(b_j - a_j)f'(\alpha_j) > 0$, entonces $b_{j+1} = a_j$
Si $(b_j - a_j)f'(\alpha_j) < 0$, entonces $b_{j+1} = b_j$

Entonces el algoritmo puede ser descrito de la siguiente manera:

Algoritmo: Seccionamiento
Entrada: $[a_j, b_j]$, $f(0)$, $f'(0)$, ρ, σ, τ_2, τ_3, i.

Salida: α_j

for $j = i, i+1, \ldots,$ do
begin escoger $\alpha_j \in [a_j + \tau_2(b_j - a_j), b_j - \tau_3(b_j - a_j)]$

evaluar $f(\alpha_j)$
if $f(\alpha_j) > f(0) + \alpha_j \rho f'(0)$ o $f(\alpha_j) > f(a_j)$
  $a_{j+1} = a_j$
  $b_{j+1} = \alpha_j$
else
  evaluar $f'(\alpha_j)$
  if $|f'(\alpha_j)| \leq -\sigma f'(0)$, salir, end
  $a_{j+1} = \alpha_j$
  if $(b_j - a_j) f'(\alpha_j) \geq 0$
    $b_{j+1} = a_j$
  else
    $b_{j+1} = b_j$
  end
end

con $0 < \tau_2 < \tau_3 \leq 1/2$.

Entonces tenemos que si

$$\begin{aligned} \text{a)}: \quad |b_{j+1} - a_{j+1}| &= |b_j - a_j - \tau_2(b_j - a_j)| \\ &= |(1 - \tau_2)(b_j - a_j)| \end{aligned}$$

$$\begin{aligned} \text{b)}: \quad |b_{j+1} - a_{j+1}| &= |b_j - \tau_3(b_j - a_j) - aj| \\ &= |(1 - \tau_3)(b_j - a_j)| \end{aligned}$$

Como $0 < \tau_2 < \tau_3 \leq 1/2$, entonces

$$\begin{aligned} |b_{j+1} - a_{j+1}| &\leq (1 - \tau_2)|b_j - a_j| \\ &\leq 1/2|b_j - a_j| \end{aligned} \tag{2.19}$$

y esto garantiza la convergencia de la longitud del corchete a cero. Los valores típicos son $\tau_2 = 1/10$, ($\tau_2 \leq \sigma$ es recomendable) y $\tau_3 = 1/2$ aunque el algoritmo no es muy sensible a los valores precisos que se utilicen.

La elección de α_j en la segunda línea del algoritmo se puede hacer interpolando un polinomio cuadrático o cúbico según la información con que se cuente y obtener el mínimo del polinomio en el intervalo requerido.

Al salir del algoritmo de búsqueda lineal, el punto α_j satisface las dos condiciones de Wolfe–Powell y es el tamaño de paso que se utiliza en el algoritmo de minimización.

Enseguida tenemos el resultado que garantiza que el algoritmo descrito converge.

Teorema. *Si $\sigma \geq \rho$ y el bracket inicial satisface las propiedades* **i)**, **ii)** *y* **iii)** *entonces el esquema de seccionamietno anterior cumple una de las siguientes propiedades:*

a) El esquema termina con un α_j que satisface las condiciones de Wolfe–Powell

b) El esquema fracasa para terminar y existe un punto c tal que para j suficientemente grande $a_j \to c$ y $b_j \to c$ monótonamente (no estricto) y c es un punto aceptable para las condiciones de Wolfe–Powell.

Además, si $\sigma > \rho$ entonces **b)** *no puede ocurrir y el esquema debe terminar.*

Demostración.

De (2.20) tenemos que $|a_j - b_j| \to 0$ y como se describe en el algoritmo $[a_{j+1}, b_{j+1}] \subset [a_j, b_j]$, entonces existe un punto límite c tal que $a_j \to c$ y $b_j \to c$ y como $\alpha_j \in [a_j, b_j]$ entonces, $\alpha_j \to c$.

Como a_j satisface que $f(a_j) < f(0) + a_j \rho f'(0)$ pero no satisface $|f'(a_j) \leq -\sigma f'(0)$, entonces se sigue que c satisface $f(c) < f(0) + c\rho f'(0)$ y

$$|f'(c)| \geq -\sigma f'(0) \tag{2.20}$$

Suponga que existe una subsucesión infinita de brackets para los cuales $b_j < a_j$, entonces por la propiedad **iii)**

$$f(b_j) > f(a_j) \Rightarrow f(b_j) - f(a_j) \geq 0$$

pero por **ii)** $(b_j - a_j) f'(a_j) \leq 0 \Rightarrow f'(a_j) \geq 0$ entonces por el Teorema del Valor Medio y la existencia de c se tiene que $f'(c) \leq 0$.

Pero por **ii)** se tiene que en el límite $f'(c) \geq 0$, entonces esto implica que $f'(c) = 0$, pero esto contradice la suposición (2.20).

Entonces no puede existir una subsucesión infinita de brackets para los cuales $b_j > a_j$, entonces $a_j \to c$ y $b_j \to c$ para j suficientemente grande.

Consideremos ahora el caso $b_j > a_j$ como $f(a_j) < f(0) + a_j\rho f'(0)$ y $f'(a_j) < 0$ ya que $(b_j - a_j)f'(a_j) < 0$ por **iii)** se tiene que

$$f(b_j) - f(a_j) > (b_j - a_j)\rho f'(0)$$

entonces

$$\frac{f(b_j) - f(a_j)}{b_j - a_j} > \rho f'(0)$$

y por el Teorema del Valor Medio existe c tal que

$$f'(c) \geq \rho f'(0)$$

Pero $(b_j - a_j)f'(a_j) < 0$ lo que implica que $f'(a_j) < 0$ y entonces $f'(c) \leq 0$. Si $\sigma > \rho$ de nuevo estas desigualdades contradicen el hecho de que $|f'(c)| \geq -\sigma f'(0)$ ya que si $\sigma > \rho$ entonces $\sigma f'(0) \leq \rho f'(0)$ esto implica que $|f'(c)| \geq \rho f'(0)$.

Esto muestra que **b)** no puede ocurrir si $\sigma \geq \rho$, en cuyo caso el algoritmo debe terminar.

Si $\sigma = \rho$ la posibilidad de no terminación existe con un punto límite c para el cual se satisface la condición de Wolfe–Powell y $f'(c) = \rho f'(0)$. ∎

Para propósitos prácticos el teorema anterior indica que se debe seleccionar $\sigma > \rho$, en cuyo caso el algoritmo de seccionamiento tiene garantía de terminar con un punto aceptable.

Con esto queda establecido que el algoritmo encuentra un punto aceptable, α_k.

2.3.3 Parámetros del algoritmo de búsqueda lineal

El algoritmo descrito contiene algunos parámetros que no se han especificado totalmente, mencionaremos ahora algunos de los valores utilizados en la práctica.

El algoritmo está descrito bajo el supuesto de que no se cometen errores de redondeo, y éstos pueden causar problemas sobre todo cuando x_k es cercana a x^*. En este caso, aún cuando $f'(0) < 0$, puede ocurrir que $f(\alpha) \geq f(0)$ para toda $\alpha > 0$, debido a errores de redondeo. Esta situación también puede presentarse si se tiene algún error en la formulación de la derivada. Entonces Fletcher propone otro criterio de paro en la fase del seccionamiento: terminar después de la línea 3 si $(a_j - \alpha_j)f'(a_j) \leq \epsilon$, donde ϵ es una tolerancia propuesta por el usuario relacionada con un parámetro para

abandonar el método de minimización si se satisface que $f_k - f_{k+1} \leq \epsilon$, en este caso se sale del algoritmo con la indicación de que no se pudo avanzar mas.

Dentro del algoritmo se menciona el uso de un polinomio de interpolación cuadrático o cúbico para generar nuevos puntos en la sucesión, tanto en la fase de localización como en el seccionamiento, estos polinomios y sobre todo los mínimos de estos polinomios deben calcularse de manera estable. Dado el intervalo $[0, 1]$ y los valores de la función y su derivada $f_0 = f(0)$, $f_1 = f(1)$, $f_0' = f'(0)$ entonces existe un único polinomio cuadrático que interpola estos valores y está definido por

$$q(z) = f_0 + f_0' z + (f_1 - f_0 - f_0')z^2$$

Si además contamos con f_1' entonces el correspondiente interpolante cúbico es

$$c(z) = f_0 + f_0' z + \eta z^2 + \psi z^3$$

donde

$$\eta = 3(f_1 - f_0)2f_0' - f_1', \qquad \psi = f_0' + f_1' - 2(f_1 - f_0)$$

Como en los algoritmos de localización y seccionamiento se trabaja con intervalos $[a, b]$ en lugar de $[0, 1]$ y $a > b$ está permitido, entonces se hace la transformación

$$\alpha = a + z(b - a)$$

el cual mapea $[0, 1]$ en $[a, b]$. Este cambio a $[0, 1]$ se realiza para que el algoritmo de interpolación sea mas estable.

Otro parámetro importante es la elección de α_1 inicial, un buen punto inicial evitará evaluaciones innecesarias de f y de su derivada. En caso de que la forma de escoger p_k esté basada en el método de Newton, como éste tiene convergencia local cuadrática, está bastante aceptado que $\alpha_1 = 1$ es una buena elección, ya que en una vecindad de x^* este paso será suficiente para avanzar en el algoritmo.

De manera similar, si p_k se obtiene por algún método Quasi–Newton, como localmente se tiene convergencia superlineal, entonces $\alpha_1 = 1$ también trabaja bien en este caso.

Si p_k no se escoge por Newton o Quasi–Newton, Fletcher propone un estimador inicial que ha funcionado bien en la práctica. Si se cuenta con un estimador $\Delta f > 0$ de la reducción que debe alcanzar f en la búsqueda lineal, entonces es posible encontrar un punto inicial minimizando la cuadrática formada con los datos $f(0), f'(0)$ y Δf. Encontrando el polinomio de interpolación tenemos que

$$f(\alpha) = f(0) + f'(0)\alpha + A\alpha^2$$

si α_1 es el mínimo, obtenemos que

$$A = -\frac{f'(0)}{2\alpha_1}$$

de aquí

$$f(\alpha_1) = f(0) + f'(0)\alpha_1 - \frac{f'(0)\alpha_1}{2}$$

entonces

$$f(\alpha_1) - f(0) = \frac{f'(0)\alpha_1}{2}$$

$$\Rightarrow -\Delta f = \frac{f'(0)\alpha_1}{2}$$

de donde resulta el estimador

$$\alpha_1 = \frac{-2\Delta f}{f'(0)}$$

En la primera iteración del método de minimización Δf debe suprimirse, pero en las siguientes iteraciones se ha encontrado que

$$\Delta f = \max(f_{(k-1)} - f_{(k)}, 10\epsilon)$$

funciona bien, esta es la reducción de la iteración anterior convenientemente salvaguardada.

2.3.4 Experimentos numéricos

Se realizó una implementación en fortran y otra en MATLAB del algoritmo aquí descrito y se realizaron pruebas computacionales para comparar su efectividad con las funciones unidimensionales propuestas por Moré y Thuente en [34]. En todos los casos las funciones tienen un único mínimo global. Los resultados presentados se obtuvieron en una máquina PENTIUM con 133 Mhz y 16 Mb en memoria RAM.

Se presentan los resultados de los experimentos partiendo de varios puntos iniciales, $\alpha_1 = 10^i$, para $i \pm 1, \pm 3$ para ilustrar el comportamiento del algoritmo con diferentes puntos iniciales. Primero se presenta en la tabla los resultados reportados por Moré y Thuente [34] y después los resultados obtenidos con nuestra implementación. Se especifican los valores de ρ y σ utilizados en cada caso.

La primera función tiene una región de concavidad a la derecha del mínimo y está definida por

$$f(\alpha) = -\frac{\alpha}{(\alpha^2 + \beta)} \tag{2.21}$$

con $\beta = 2$, la gráfica de esta función se ilustra en la figura 2.18 y los resultados se presentan en la Tabla 1.

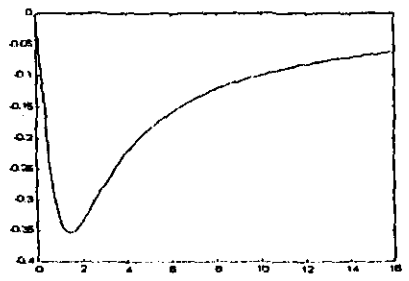


Figura 2.18: Gráfica de la función (2.21) con $\beta = 2$.

Tabla 1. Resultados obtenidos para la función de la figura 2.18 con $\rho = 0.001$ y $\sigma = 0.1$

	Resultados en [34]			Resultados del Algoritmo		
α_1	no.iter.	α_n	$f\prime(\alpha_n)$	no.iter.	α_n	$f\prime(\alpha_n)$
10^{-3}	6	1.4	$-9.2\ 10^{-3}$	5	1.5	$2.0\ 10^{-2}$
10^{-1}	3	1.4	$4.7\ 10^{-3}$	4	1.4	$6.0\ 10^{-3}$
10^{+1}	1	10	$9.4\ 10^{-3}$	1	10	$-9.8\ 10^{-2}$
10^{+3}	4	37	$7.3\ 10^{-4}$	6	31.2	$1.0\ 10^{-3}$

La segunda función de prueba es cóncava a la izquierda del mínimo y se define como

$$f(\alpha) = (\alpha + \beta)^5 - 2(\alpha + \beta)^4 \tag{2.22}$$

con $\beta = 0.004$. La gráfica de esta función se ilustra en la figura 2.19 y los resultados computacionales se presentan en la Tabla 2.

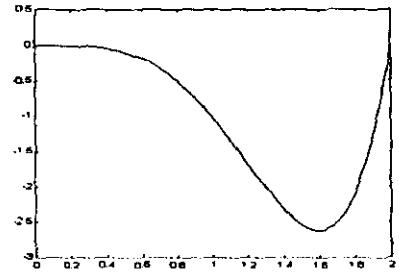


Figura 2.19: Gráfica de la función (2.22) con $\beta = 0.004$.

Tabla 2. Resultados obtenidos para la función de la figura 2.19 con $\rho = 0.1$ y $\sigma = 0.1$

	Resultados en [34]			Resultados del Algoritmo		
α_1	no.iter.	α_n	$f'(\alpha_n)$	no.iter.	α_n	$f'(\alpha_n)$
10^{-3}	12	1.6	$7.1\ 10^{-9}$	11	1.5	$3.7\ 10^{-3}$
10^{-1}	8	1.6	$1.0\ 10^{-10}$	9	1.5	$1.0\ 10^{-3}$
10^{+1}	8	1.6	$-5.0\ 10^{-9}$	7	1.5	$1.8\ 10^{-2}$
10^{+3}	11	1.6	$-2.3\ 10^{-8}$	9	1.5	$1.8\ 10^{-2}$

La tercera función de prueba está definida en términos de los parámetros l y β, por

$$f(\alpha) = f_0(\alpha) + \frac{2(1-\beta)}{l\pi} \text{ sen } \left(\frac{l\pi}{2}\alpha\right) \tag{2.23}$$

donde

$$f_0(\alpha) = \begin{cases} 1-\alpha & \text{si } \alpha \leq 1-\beta \\ \alpha - 1 & \text{si } \alpha \geq 1+\beta \\ \frac{1}{2\beta}(\alpha-1)^2 + \frac{1}{2}\beta & \text{si } \alpha \in [1-\beta, 1+\beta] \end{cases}$$

El parámetro β controla el tamaño de $f'(0) = -\beta$. Este parámetro también controla el tamaño del intervalo donde se satisface la segunda condición de Wolfe–Powell, ya que $|f'(\alpha)| \geq \beta$ para $|\alpha - 1| \geq \beta$ y entonces la segunda condición de Wolfe–Powell solo puede satisfacerse para $|\alpha - 1| < \beta$. El parámetro l controla el número de oscilaciones en la función. Note que si l es impar entonces $f'(1) = 0$ y que si $l = 4k - 1$ para algún entero $k \geq 1$, entonces $f''(1) > 0$.

Escogemos $\beta = 0.01$ y $l = 39$. La gráfica de la función con estos parámetros aparece en la figura 2.20 y los resultados computacionales se presentan en la Tabla 3.

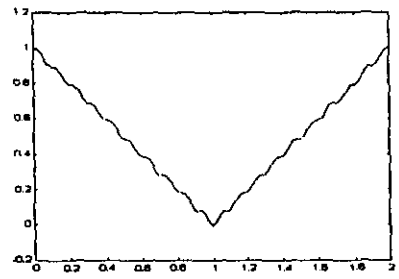


Figura 2.20: Gráfica de la función (2.23) con $\beta = 0.01$ y $l = 39$.

Tabla 3. Resultados obtenidos para la función de la figura 2.20 con $\rho = 0.1$ y $\sigma = 0.1$

	Resultados en [34]			Resultados del Algoritmo		
α_1	no.iter.	α_n	$f\prime(\alpha_n)$	no.iter.	α_n	$f\prime(\alpha_n)$
10^{-3}	12	1.0	$-5.1\ 10^{-5}$	10	1.0	$6.1\ 10^{-6}$
10^{-1}	12	1.0	$-1.9\ 10^{-4}$	6	1.0	$-3.2\ 10^{-7}$
10^{+1}	10	1.0	$-2.0\ 10^{-6}$	2	1.0	$-4.1\ 10^{-15}$
10^{+3}	13	1.0	$-1.6\ 10^{-5}$	4	1.0	$-4.1\ 10^{-15}$

Se observa que los resultados, comparados con los reportados por Moré y Thuente, son satisfactorios. El punto encontrado está muy cerca del obtenido en [34] y en cuanto al número de iteraciones en las primeras dos funciones está arriba o abajo una iteración con respecto a los resultados de Moré, pero en general son resultados similares. En cuanto a la tercera función, se observa que nuestro algoritmo necesita menos iteraciones para alcanzar convergencia, obteniendo un punto aceptable con menos esfuerzo computacional.

Referencias

R. Fletcher (1987). *Practical Methods of Optimization* John Wiley & Sons, 2nd.Edition.

J.Moré, D.J. Tuente, (1994). Line Search Algorithms with guaranteed sufficient decrease, *ACM Transactions on Mathematical Software* vol. 20, pp. 286–307.

J. Nocedal and J. Wright (1999). *Numerical Optimization*, Springer Series in Operations Research.

Capítulo 3

Gradiente Conjugado Lineal

Las funciones cuadráticas juegan un papel muy importante dentro de los métodos de solución de optimización no lineal. Recordemos que localmente toda función en el mínimo puede ser vista como una cuadrática y que de la familia de funciones no lineales, las cuadráticas son las mas fáciles de tratar. Las técnicas para la minimización de una función cuadrática son el punto de partida en el que están basados muchos de los métodos de optimización, por lo que nos es necesario entender el comportamiento de una función cuadrática y contar con algoritmos que calculen sus mínimos de manera eficiente y económica.

Describiremos en este capítulo un algoritmo para minimizar cuadráticas desarrollado en los años 50's por Hestenes y Stifiel conocido como Gradiente Conjugado, ésta es una técnica principalmente útil cuando se cuenta con un gran número de variables ya que resulta un proceso bastante económico que solo requiere para operar unos cuantos vectores. Este procedimiento cuenta con una característica importante: bajo aritmética exacta la convergencia está garantizada en a lo más n pasos, con n la dimensión del problema, sin embargo, como veremos en algunos casos, pueden converger en un número pequeño de iteraciones relativo a la dimensión del problema.

3.1 Problema

Consideremos el problema de minimizar la función cuadrática descrita por

$$\Psi(x) = \frac{1}{2}x^t Ax + b^t x$$

donde $A \in \mathbb{R}^{n\times n}$ simétrica, $b \in \mathbb{R}^n$. Veremos primero bajo qué condiciones esta cuadrática tiene mínimo con el siguiente resultado.

Lema. *Sea* $\Psi : \mathbb{R}^n \to \mathbb{R}$ *la función cuadrática*

$$\Psi(x) = \frac{1}{2}x^t Ax + b^t x$$

donde $b \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$ *matriz simétrica.*

a) *La cuadrática* Ψ *tiene un mínimo si y sólo si* A *es positiva semidefinida y* $-b$ *está en la Imagen de* A.

b) *La cuadrática* Ψ *tiene un único mínimo si y sólo si* A *es positiva definida.*

c) *Si* A *es positiva semidefinida, entonces cada solución de la ecuación* $Ax = -b$ *es un minimizador global.*

Demostración.
(a) $\Longleftarrow$) Supongamos que $A \geq 0$ y $b \in Im(A)$. Entonces $Ax = -b$ tiene una solución p, y se tiene

$$\begin{aligned}
\Psi(p+x) &= \frac{1}{2}(p+x)^t A(p+x) + b^t(p+x) \\
&= \frac{1}{2}p^t Ap + b^t x + (Ap)^t x + \frac{1}{2}x^t Ax + b^t p \\
&= \Psi(p) + (b+Ap)^t x + \frac{1}{2}x^t Ax \\
&= \Psi(p) + \frac{1}{2}x^t Ax \\
&\geq \Psi(p)
\end{aligned}$$

La última desigualdad se da ya que $A \geq 0$, ahora, esto se cumple para toda $x \in \mathbb{R}^n$, entonces p es un mínimo de Ψ.

$\Longrightarrow$) Supongamos que p es el mínimo de Ψ, entonces por las condiciones necesarias de primer orden de un mínimo local, se tiene que

$$\nabla\Psi(p) = 0$$

$$\Rightarrow Ap + b = 0, \Rightarrow -b \in Im(A)$$

y se tiene que $\nabla^2\Psi(p) \geq 0, \Rightarrow A = \nabla^2\Psi(p) \geq 0$. Con esto queda demostrado (a)

(b) $\Longleftarrow$) Se utiliza el mismo argumento que en (a) sólo que ahora se satisface $x^t Ax > 0$ lo que implica que

$$\Psi(p+x) > \Psi(p), \quad \forall x$$

lo que implica que p es el único mínimo de Ψ.

$\Longrightarrow$) Supongamos que $A \geq 0$ no es positiva definida, entonces existe un vector $x \neq 0$ tal que $Ax = 0$, entonces por lo anterior, tenemos que

$$\Psi(p+x) = \Psi(p)$$

y el mínimo no es único lo cual indica una contradicción, entonces $A > 0$.

(c) se sigue de (a). ■

Dada la cuadrática Ψ se tiene un buen procedimiento numérico para encontrar su mínimo. Primero se intenta calcular la factorización de Cholesky de A, esta factorización existe si y sólo si $A \geq 0$ y en este caso se obtiene una matriz triangular superior R tal que

$$B = R^t R$$

Si durante el proceso de factorización se encuentra un elemento negativo en la diagonal, entonces A no es positiva semidefinida y el lema anterior muesta que Ψ no tiene mínimo. Si la factorización existe y R es no singular, entonces el mínimo se calcula resolviendo el sistema $Ap = -b$, o equivalentemente,

$$R^t v = -b, \qquad Rp = v$$

Entonces minimizar Ψ y resolver el sistema $Ax = -b$ son problemas equivalentes si A es simétrica positiva definida. El proceso de encontrar la factorización de Cholesky de A es el mas utilizado en la práctica para minimizar Ψ, pero para el caso de A con grandes dimensiones esto puede representar una cantidad inaceptable de trabajo, de aquí la necesidad de una técnica económica para el caso de un gran número de variables. El método de Gradiente Conjugado es de los mas utilizados en estos casos, describiremos a continuación las ideas detrás de este método.

Minimizaremos $\Psi(x)$ partiendo de que A es positiva definida para garantizar que tiene mínimo. Con los antecedentes del capítulo anterior minimizaremos $\Psi(x)$ con algún método de descenso con búsqueda lineal: Dado x_0 punto inicial, generamos

$$x_{k+1} = x_k + \alpha_k p_k$$

con p_k alguna dirección de descenso y α_k el *mínimo* de $\Psi(x)$ sobre la dirección p_k.

Veamos cómo escoger α_k y p_k. Como Ψ es cuadrática, podemos calcular α_k de forma exacta, tenemos que

$$\alpha_k = \arg\min_{\alpha} \Psi(x_k + \alpha p_k)$$

para encontrar el *mínimo* calculamos los puntos críticos, esto es,

$$\nabla\Psi(x_k + \alpha p_k)^t p_k = 0$$

de donde resulta

$$(A(x_k + \alpha p_k) + b)^t p_k = 0$$

lo cual implica

$$(Ax_k + b)^t p_k + \alpha p_k^t A p_k = 0$$

de donde obtenemos α_k,

$$\alpha_k = \frac{(-b - Ax_k)^t p_k}{p_k^t A p_k}$$

Notemos que $\nabla\Psi(x_k) = b + Ax_k$, entonces como notación introducimos el residual en x_k como

$$r_k = -b - Ax_k$$

Note que

$$r_k = -\nabla\Psi(x_k)$$

con esta notación entonces tenemos que

$$\alpha_k = \frac{r_k^t p_k}{p_k^t A p_k} \tag{3.1}$$

es el mínimo de Ψ sobre la dirección p_k. Considerando α_k de esta forma se obtiene

$$\Psi(x_k + \alpha_k p_k) = \Psi(x_k) - \frac{1}{2}\frac{(r_k^t p_k)^2}{p_k^t A p_k}$$

entonces para asegurar una reducción en Ψ la dirección p_k no puede ser ortogonal a r_k, o lo que es lo mismo, a $-\nabla\Psi(x)$, esto es, necesitamos

$$r_k^t p_k \neq 0 \tag{3.2}$$

Esto da lugar al siguiente algoritmo para encontrar x^*, el mínimo de Ψ:

$$
\begin{array}{l}
x_0 = \text{ punto inicial} \\
r_0 = -b - Ax_0 \\
k = 0 \\
\text{while } r_k \neq 0 \\
\quad \text{Escoger una dirección } p_k \text{ tal que } r_k^t p_k \neq 0 \\
\quad \alpha_k = r_k^t p_k / p_k^t A p_k \\
\quad x_{k+1} = x_k + \alpha_k p_k \\
\quad r_{k+1} = -b - Ax_{k+1} \\
\quad k = k + 1 \\
\text{end}
\end{array}
\tag{3.3}
$$

Ahora tenemos el problema de cómo escoger la dirección p_k en la práctica.

3.2 Método de Direcciones Conjugadas

La primera opción que puede considerarse es la dirección de menos el gradiente en el punto actual, o el residual en este caso, ésta es una dirección de descenso y satisface (3.2), pero esto conduciría al método de descenso más rápido con búsqueda lineal y como se sabe este método converge muy lentamente. Otra opción para p_k es la dirección de Newton y tendríamos convergencia en un solo paso, pero no es costeable cuando la dimensión es muy grande.

Entonces buscaremos una forma económica de calcular p_k de manera que se garantice convergencia, tratando de evitar los problemas del método de descenso más rápido.

Supongamos que estamos en la primera iteración, esto es, dados x_0 y p_0, calculamos α_0 por (3.1) y obtenemos

$$x_1 = x_0 + \alpha_0 p_0$$

Ahora, cómo obtener una buena dirección para avanzar, p_1? Tenemos que si $x_2 = x_1 + \alpha p_1$ entonces

$$
\begin{aligned}
\Psi(x_2) &= \Psi(x_1 + \alpha p_1) \\
&= \Psi(x_0 + \alpha_0 p_0 + \alpha p_1) \\
&= \Psi(x_0 + \alpha_0 p_0) + \alpha\alpha_0 p_0^t A p_1 + \frac{\alpha^2}{2} p_1^t A p_1 - \alpha p_1^t r_0
\end{aligned}
$$

El objetivo es minimizar $\Psi(x)$; en la primera iteración obtuvimos x_1 de manera que minimiza $\Psi(x)$ sobre $x_0 + span\{p_0\}$, entonces parece razonable escoger x_2 de manera

que minimice $\Psi(x)$ sobre $x_0 + span\{p_0, p_1\}$. Sabemos que

$$x_1 = x_0 + \alpha_0 p_0 = \arg \min_{x \in x_0 + span\{p_0\}} \Psi(x)$$

y que

$$\psi(\alpha) = \frac{\alpha^2}{2} p_1^t A p_1 - \alpha p_1^t r_0$$

se minimiza en

$$\alpha^* = \frac{p_1^t r_0}{p_1^t A p_1}$$

sobre la dirección p_1. Entonces si p_1 es tal que el término cruzado

$$p_0^t A p_1 = 0 \tag{3.4}$$

entonces tenemos que $x_2 = x_1 + \alpha^* p_1$ minimiza $\Psi(x)$ sobre $x_0 + span\{p_0, p_1\}$ y es nuestra siguiente iteración. Tenemos entonces otra condición para p_1, necesitamos que $p_1^t A p_0 = 0$.

Observe que si (3.4) se satisface, entonces

$$\begin{aligned}
\alpha^* &= \frac{p_1^t r_0}{p_1^t A p_1} \\
&= \frac{p_1^t(-b - Ax_0)}{p_1^t A p_1} \\
&= \frac{p_1^t(-b - Ax_0) - \alpha_0 p_1^t A p_0}{p_1^t A p_1} \\
&= \frac{p_1^t(-b - A(x_0 + \alpha_0 p_0))}{p_1^t A p_1} \\
&= \frac{p_1^t(-b - Ax_1)}{p_1^t A p_1} \\
&= \frac{p_1^t r_1}{p_1^t A p_1} \\
&= \alpha_1
\end{aligned}$$

esto es, el tamaño de paso es α_1 lo que habíamos obtenido en (3.1).

Resumiendo, una vez obtenido x_1, necesitamos una dirección p_1 tal que

$$p_1^t A p_0 = 0, \qquad p_1^t r_1 \neq 0$$

con esto generamos la nueva iteración en el método de descenso

$$x_2 = x_1 + \alpha_1 p_1$$

con α_1 calculada como en (3.1).

Para el caso general supongamos que estamos en el paso k y que obtuvimos x_k que minimiza $\Psi(x)$ en $x_0 + span\{p_0, p_1, \ldots, p_{k-1}\}$ queremos encontrar la dirección p_k. Tenemos que

$$x_{k+1} = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \cdots + \alpha p_k$$

entonces

$$\begin{aligned} \Psi(x_{k+1}) &= \Psi(x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \cdots + \alpha_{k-1} p_{k-1}) \\ &\quad + \alpha \textstyle\sum_{i=0}^{k-1} \alpha_i p_i^t A p_k + \frac{\alpha^2}{2} p_k^t A p_k - \alpha p_k^t r_0 \end{aligned} \tag{3.5}$$

queremos que x_{k+1} minimice $\Psi(x)$ sobre $x_0 + span\{p_0, p_1, \ldots, p_k\}$. Si

$$p_i^t A p_k = 0, \qquad i = 0, \ldots, k-1 \tag{3.6}$$

entonces

$$\sum_{i=0}^{k-1} \alpha_i p_i^t A p_k = 0$$

y la minimización de (3.5) se divide en dos minimizaciones con parámetros independientes, esto es,

$$\begin{aligned} \min_{x_0 + span\{p_0, p_1, \ldots, p_k\}} \Psi(x_{k+1}) &= \min_{\{\alpha_i\}_{i=0}^{k-1}} \Psi(x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \cdots + \alpha_{k-1} p_{k-1}) \\ &\quad + \min_\alpha \left(\frac{\alpha^2}{2} p_k^t A p_k - \alpha p_k^t r_0 \right) \end{aligned} \tag{3.7}$$

Sabemos que x_k minimiza el primer término de la derecha de (3.7) y que

$$\alpha^* = \frac{p_k^t r_0}{p_k^t A p_k}$$

minimiza el segundo término, entonces si (3.6) se satisface

$$x_{k+1} = x_k + \alpha^* p_k$$

minimiza $\Psi(x)$ sobre $x_0 + span\{p_0, p_1, \ldots, p_k\}$.

Notemos que debido a (3.6)

$$\begin{aligned} p_k^t r_k &= p_k^t(-b - Ax_k) \\ &= p_k^t(-b - A(x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \cdots \alpha_{k-1} p_{k-1}) \\ &= p_k^t(-b - Ax_0 - \sum_{i=0}^{k-1} \alpha_i A p_i) \\ &= p_k^t(-b - Ax_0) \\ &= p_k^t r_0 \end{aligned}$$

Entonces $\alpha^* = \alpha_k$ con α_k obtenido por (3.1). Tenemos entonces que si

$$p_i^t A p_k = 0, \qquad i = 0, \ldots, k-1$$

y

$$p_k^t r_k \neq 0$$

p_k es una buena dirección para avanzar en el algoritmo de minimización y

$$x_{k+1} = x_k + \alpha_k p_k$$

con α_k de (3.1) es nuestro siguiente elemento en la iteración.

Observemos que en cada iteración se busca la dirección p_k de tal forma que al resolver el problema de minimización unidimensional del tamaño de paso resolvamos también el problema de minimización k–dimensional del $\min \Psi(x)$ sobre $x_0 + span\{p_0, p_1, \ldots, p_k\}$.

Las direcciones que satisfacen (3.6) se denominan direcciones conjugadas.

Definición. Dada A simétrica positiva definida, un conjunto de vectores $\{p_0, p_1, \ldots p_k\}$ se dice que es conjugado con respecto a A o A–conjugados si

$$p_i^t A p_j = 0, \qquad \text{si } i \neq j$$

Esta es una generalización del concepto de ortogonalidad cuando $A = I$.

No es difícil demostrar que las direcciones A–conjugadas forman un conjunto de vectores linealmente independientes, esta propiedad la utilizaremos mas adelante.

Con esta definición hemos demostrado entonces la siguiente propiedad sobre las direcciones A–conjugadas.

Propiedad. *Si $\{p_0, \ldots, p_k\}$ es un conjunto de vectores A–conjugados, entonces satisfacen*

$$\min \Psi(x_k + \alpha p_k) = \min_{\{\alpha_i\}} \Psi\left(x_0 + \sum_{i=0}^{k-1} \alpha_i p_i\right)$$

Otra propiedad importante de escoger las direcciones de búsqueda A–conjugadas es que la convergencia del método de descenso está garantizada en a lo mas n pasos, esto ya que

$$x_n = \arg \min_{x \in x_0 + span\{p_0, \ldots, p_{n-1}\}} \Psi(x)$$

y como $\{p_0, \ldots, p_{n-1}\}$ son linealmente independientes, entonces x_n minimiza Ψ sobre todo $\mathbb{R}^n$ y el algoritmo termina.

Tenemos entonces todos los elementos para establecer el algoritmo del método de direcciones conjugadas, esto es, un método de descenso con búsqueda lineal donde las direcciones de búsqueda sean A–conjugadas.

$x_0 =$ punto inicial
$p_0 =$ dirección inicial
$r_0 = -b - Ax_0$
$\alpha_0 = r_0^t p_0 / p_0^t A p_0$
$x_1 = x_0 + \alpha_0 p_0$
$r_1 = -b - Ax_1$
$k = 1$
while $r_k \neq 0$
    Escoger una dirección p_k tal que
      $p_k \in span\{Ap_0, Ap_1, \ldots Ap_{k-1}\}^\perp$ con $r_k^t p_k \neq 0$
    $\alpha_k = r_k^t p_k / p_k^t A p_k$
    $x_{k+1} = x_k + \alpha_k p_k$
    $r_{k+1} = -b - Ax_{k+1}$
    $k = k + 1$
end

El siguiente resultado muestra que es posible encontrar las direcciones de búsqueda con las propiedades requeridas:

Lema. *Si $r_k \neq 0$, entonces existe un $p_k \in span\{Ap_0, \ldots, Ap_{k-1}\}^\perp$ tal que $p_k^t r_k \neq 0$.*

Demostración.
Procederemos por inducción. Para el caso $k = 0$, consideremos $p_0 = r_0$.

Si $k > 0$, entonces como $r_k \neq 0$ tenemos que

$$\begin{aligned} A^{-1}b \notin x_0 + span\{p_0, \ldots, p_{k-1}\} &\Rightarrow b \notin Ax_0 + span\{Ap_0, \ldots Ap_{k-1}\} \\ &\Rightarrow r_0 \notin span\{Ap_0, \ldots Ap_{k-1}\} \end{aligned}$$

Entonces existe un $p \in span\{Ap_0, \ldots Ap_{k-1}\}^{\perp}$ tal que $p^t r_0 \neq 0$.

Pero $x_k \in x_0 + span\{p_0, \ldots, p_{k-1}\}$ y entonces $r_k \in r_0 + span\{Ap_0, \ldots, Ap_{k-1}\}$.

Se sigue entonces que $p^t r_k = p^t r_0 \neq 0$. ■

Ya tenemos entonces las condiciones sobre las direcciones de búsqueda, veremos ahora la forma práctica de encontrarlas dando lugar al método de Gradiente Conjugado.

3.3 Método de Gradiente Conjugado

La forma usual de generar un conjunto de direcciones conjugadas es seleccionar un conjunto de vectores $\{z_0, z_1, \ldots, z_{n-1}\}$ linealmente independientes y aplicar el método de ortogonalización de Gram–Schmidt. El problema que se nos presenta ahora es de qué base partir.

El método de Gradiente Conjugado consiste en usar como base las direcciones del gradiente en cada iteración para generar las direcciones conjugadas. Entonces partiendo de la dirección de menos el gradiente, la de descenso mas rápido o el residual, r_k, lo ortogonalizamos con respecto a $\{Ap_0, \ldots, Ap_{k-1}\}$ y obtenemos las dirección p_k. Para la dirección inicial se parte del residual en el punto inicial, esto es, $p_0 = r_0$. Sea $S = span\{Ap_0, Ap_1, \ldots, Ap_{k-1}\}$ entonces geométricamente lo que tenemos es lo que se ilustra en la figura 3.1.

Entonces tenemos que p_k será el vector en $span\{Ap_0, Ap_1, \ldots, Ap_{k-1}\}^{\perp}$ mas cercano a r_k, en otras palabras, queremos encontar

$$\begin{gathered} p_k \text{ que minimice } \|p - r_k\|_2 \text{ sobre todos lo vectores} \\ p \in span\{Ap_0, Ap_1, \ldots Ap_{k-1}\}^{\perp} \end{gathered} \tag{3.8}$$

Esta dirección p_k la podemos caracterizar como la solución a un problema de mínimos cuadrados.

Antes de enunciar el resultado que lo justifica introducimos la siguiente notación: Sea P_k la matriz de $n \times k$ formada por todas las direcciones $\{p_i\}_{i=0}^{k-1}$, esto es,

$$P_k = [p_0, p_1, \ldots, p_{k-1}]_{n \times k}$$

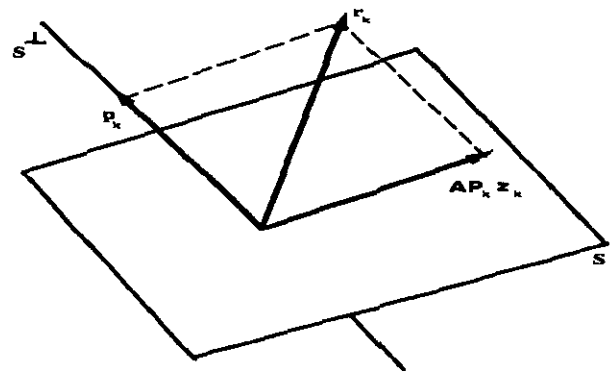


Figura 3.1: Obtención de la dirección p_k

con esta notación tenemos el siguiente.

Lema. *Para $k \geq 1$ los vectores que satisfacen (3.8) son tales que*

$$p_k = r_k - AP_k z_k$$

donde z_k resuelve el problema de $\min_{z \in \mathbb{R}^k} \|r_k - AP_k z\|_2$.

Demostración.
Supongamos que z_k es tal que

$$\min_{z \in \mathbb{R}^k} \|r_k - AP_k z\|_2 = \|r_k - AP_k z_k\|_2$$

entonces, si observamos la figura 3.1, tenemos que el residual mínimo para este problema de mínimos cuadrados es precisamente p_k, esto es,

$$p_k = r_k - AP_k z_k$$

y este vector es la proyección de r_k sobre $span\{Ap_0, Ap_1, \ldots, Ap_{k-1}\}^{\perp}$, por lo tanto satisface (3.8), lo cual completa la demostración. ■

Nos enfocaremos ahora a la tarea de buscar expresiones simples para p_k, para esto veremos algunas propiedades importantes que simplificarán los cálculos.

Notemos que como

$$x_{k+1} = x_k + \alpha_k p_k$$

entonces

$$Ax_{k+1} = A(x_k + \alpha_k p_k)$$

lo que implica que

$$r_{k+1} = r_k - \alpha_k A p_k \tag{3.9}$$

Esta propiedad de recurrencia nos servirá para demostrar la relación que se guarda entre las direcciones de búsqueda y los llamados subespacios de Krylov, los cuales están definidos por

$$\mathcal{K}(r_0, A, k) = span\{r_0, Ar_0, \ldots, A^k r_0\}$$

enunciamos estas propiedades en el siguiente resultado.

Teorema. *Después de k pasos de la iteración*

$$x_{k+1} = x_k + \alpha_k p_k$$

donde las direcciones p_k satisfacen (3.8) se satisfacen las siguientes relaciones

$$P_k^t r_k = 0 \tag{3.10}$$

$$span\{p_0, \ldots, p_k\} = span\{r_0, \ldots, r_k\} = \mathcal{K}(r_0, A, k) \tag{3.11}$$

y los residuales $\{r_0, r_1, \ldots, r_k\}$ son mutuamente ortogonales.

Demostración.
Demostraremos primero (3.10), esto es, queremos probar que

$$p_i^t r_k = 0, \qquad i = 0, \ldots, k-1$$

esto es, el residual actual es ortogonal a todas las direcciones anteriores. Procederemos por inducción. Para el caso $k = 1$, como el residual es en este caso menos el gradiente y α_1 es el minimizador unidimensional exacto, tenemos que

$$p_0^t r_1 = 0$$

Supondremos como hipótesis de inducción que $p_i^t r_{k-1} = 0$, para $i = 0, \ldots, k-2$. De (3.9) tenemos que

$$r_k = r_{k-1} - \alpha_{k-1} A p_{k-1}$$

y esto implica que

$$\begin{aligned} p_{k-1}^t r_k &= p_{k-1}^t r_{k-1} - \alpha_{k-1} p_{k-1}^t A p_{k-1} \\ &= p_{k-1}^t r_{k-1} - \frac{p_{k-1}^t r_k}{p_{k-1}^t A p_{k-1}} p_{k-1}^t A p_{k-1} \\ &= 0 \end{aligned}$$

Entonces solo falta para los vectores p_i, con $i = 0, \ldots, k-2$, pero también de (3.9) y por la A–conjugación de las direcciones de búsqueda tenemos que

$$\begin{aligned} p_i^t r_k &= p_i^t r_{k-1} - \alpha_{k-1} p_i^t A p_{k-1} \\ &= p_i^t r_{k-1} \\ &= 0 \end{aligned}$$

donde la última igualdad se da por la hipótesis de inducción. Entonces tenemos que

$$p_i^t r_k = 0, \qquad i = 0, \ldots, k-1$$

Nos enfocaremos ahora en la primera igualdad de (3.11). Tenemos que como

$$r_k = r_{k-1} - \alpha_{k-1} A p_{k-1}$$

entonces

$$\{Ap_0, \ldots, Ap_{k-1}\} \subseteq span\{r_0, \ldots, r_k\}$$

y por la forma de p_k tenemos que

$$p_k = r_k - [Ap_0, Ap_1, \ldots, Ap_{k-1}]z_k \in span\{r_0, \ldots r_k\}$$

Esto para cada k. Entonces

$$span\{p_0, \ldots, p_k\} \subseteq span\{r_0, \ldots, r_k\}$$

pero las direcciones A–conjugadas son linealmente independientes, entonces

$$span\{p_0, \ldots, p_k\} = span\{r_0, \ldots, r_k\}$$

Para la segunda igualdad procederemos por inducción. Es claro para $k = 0$ ya que estamos partiendo de que $p_0 = r_0$. Supongamos que se satisface para alguna k, esto es,

$$span\{r_0, \ldots, r_k\} = span\{r_0, Ar_0, \ldots, A^k r_0\}$$

veremos que se satisface para $k+1$.

Tenemos que

$$r_{k+1} = r_k - \alpha_k A p_k$$

de donde

$$r_{k+1} \in span\{r_k, Ap_k\}$$

pero por hipótesis de inducción

$$p_k \in span\{r_0, Ar_0, \ldots, A^k r_0\}$$

lo que implica que

$$r_{k+1} \in span\{r_k, Ar_0, \ldots, A^{k+1} r_0\}$$

por lo tanto

$$span\{r_0, \ldots, r_{k+1}\} \subseteq span\{r_0, Ar_0, \ldots, A^k r_0, A^{k+1} r_0\}$$

pero la parte derecha de esta igualdad tiene dimensión a lo mas $k+2$ y por la primera igualdad de (3.11) $span\{r_0, \ldots, r_{k+1}\}$ tiene dimensión $k+2$, ya que las $\{p_i\}_{i=0}^{k+1}$ son linealmente independientes, por lo tanto

$$span\{r_0, \ldots, r_{k+1}\} = span\{r_0, Ar_0, \ldots, A^k r_0, A^{k+1} r_0\}$$

que es lo que queríamos demostrar.

Falta solo la demostración de que los residuales son mutuamente ortogonales. Para esto, utilizando (3.10) tenemos que

$$r_k \in span\{p_0, \ldots, p_{k-1}\}^{\perp}$$

pero por la primera igualdad de (3.11), tenemos que

$$span\{p_0, \ldots, p_{k-1}\}^{\perp} = span\{r_0, \ldots, r_{k-1}\}^{\perp}$$

entonces tenemos que r_k es ortogonal a cada $r_0, r_1, \ldots, r_{k-1}$.

Esto termina la demostación del teorema. ■

Entonces, resumiendo las propiedades demostradas son

$$\begin{aligned} r_{k+1} &= r_k - \alpha_k A p_k \\ p_i^t r_k &= 0, \qquad i = 0, \ldots, k-1 \\ r_i^t r_k &= 0, \qquad i = 0, \ldots, k-1 \\ span\{p_0, p_1, \ldots, p_k\} &= span\{r_0, r_1, \ldots, r_k\} = span\{r_0, Ar_0, \ldots, A^k r_0\} \end{aligned}$$

Estas propiedades nos ayudarán a encontrar expresiones simples para p_k.

Tenemos ahora el resultado fundamental para el algoritmo de Gradiente Conjugado: toda dirección p_k depende solamente del residual actual y la dirección anterior.

Corolario. *Los residuales y las direcciones de búsqueda p_k tienen la propiedad de que $p_k \in span\{p_{k-1}, r_k\}$ para $k \geq 1$.*

Demostración.

Procederemos por inducción.

Si $k = 1$, entonces tenemos que

$$span\{p_0, p_1\} = span\{r_0, r_1\}$$

y $p_1 \in span\{r_0, r_1\}$, pero estamos partiendo de que $p_0 = r_0$, entonces p_1 es una combinación lineal de p_0 y r_1, esto es,

$$p_1 \in span\{p_0, r_1\}$$

por lo tanto si se satisface para $k = 1$.

Antes de hacer el caso general, analizaremos el caso particular $k = 2$, lo cual nos ayudará en la generalización.

Supongamos $k = 2$, sea $z_2 \in \mathbb{R}^2$ que resuelve el problema de mínimos cuadrados (3.8) asociado a la elección de p_2. Podemos particionar

$$z_2 = \begin{pmatrix} w \\ \mu \end{pmatrix}$$

con $w, \mu \in \mathbb{R}$ en este caso. Con lo que tenemos que

$$p_2 = r_2 - AP_2 z_2$$

donde $P_2 = [p_0, p_1]$. Tomando la identidad

$$r_2 = r_1 - \alpha_1 A p_1$$

tenemos que $Ap_1 = \frac{1}{\alpha_1}(r_1 - r_2)$, entonces

$$\begin{aligned} p_2 &= r_2 - Ap_0 w - Ap_1 \mu \\ &= r_2 - Ap_0 w - \frac{\mu}{\alpha_1}(r_1 - r_2) \\ &= \left(1 + \frac{\mu}{\alpha_1}\right) r_2 - \frac{\mu}{\alpha_1} r_1 - Ap_0 w \\ &= \left(1 + \frac{\mu}{\alpha_1}\right) r_2 + s_2 \end{aligned}$$

con

$$s_2 = -\frac{\mu}{\alpha_1} r_1 - Ap_0 w$$

Ahora, $s_2 \in span\{r_1, Ap_0\} \in span\{r_0, r_1\}$. Como los r_i son mutuamente ortogonales, entonces r_2 y s_2 son ortogonales.

El problema de mínimos cuadrados se reduce a escoger μ y w tales que se minimice

$$\|p_2\|_2^2 = \left(1 + \frac{\mu}{\alpha_1}\right)^2 \|r_2\|_2^2 + \|s_2\|_2^2$$

Ahora, tenemos que en el paso anterior la norma-2 de

$$r_1 - Ap_0 z$$

se minimiza en z_1 con residual p_1, entonces

$$s_2 = -\frac{\mu}{\alpha_1} r_1 - Ap_0 w$$

debe ser un múltiplo de p_1 ya que solo estamos variando la longitud de r_1.

Por lo tanto, p_2 es combinación lineal de r_2 y p_1, esto es,

$$p_2 \in span\{p_1, r_2\}$$

esto termina la demostración para el caso $k = 2$.

Haremos ahora la demostración para el caso general, supongamos que $k > 2$ y que z_k resuelve el problema de mínimos cuadrados para p_k. Demostraremos que

$$p_k \in span\{p_{k-1}, r_k\}$$

Se puede particionar z_k como

$$z_k = \begin{pmatrix} w \\ \mu \end{pmatrix} \begin{matrix} k-1 \\ 1 \end{matrix}$$

Como $r_k = r_{k-1} - \alpha_{k-1} A p_{k-1}$, entonces

$$Ap_{k-1} = \frac{1}{\alpha_{k-1}}(r_{k-1} - r_k)$$

como z_k resuelve el problema de mínimos cuadrados y p_k es el residual mínimo, tenemos

$$\begin{aligned} p_k &= r_k - AP_k z_k \\ &= r_k - AP_{k-1} w - Ap_{k-1}\mu \\ &= r_k - AP_{k-1} w - \frac{\mu}{\alpha_{k-1}}(r_{k-1} - r_k) \\ &= \left(1 + \frac{\mu}{\alpha_{k-1}}\right) r_k - \frac{\mu}{\alpha_{k-1}} r_{k-1} - AP_{k-1} w \\ &= \left(1 + \frac{\mu}{\alpha_{k-1}}\right) r_k + s_k \end{aligned}$$

donde

$$s_k = -\frac{\mu}{\alpha_{k-1}} r_{k-1} - AP_{k-1} w$$

de donde tenemos que

$$\begin{aligned} s_k &\in span\{r_{k-1}, AP_{k-1} w\} \\ &\in span\{r_{k-1}, Ap_0, Ap_1, \ldots, Ap_{k-2}\} \\ &\in span\{r_0, r_1, \ldots, r_{k-1}\} \end{aligned}$$

Como los residuales son mutuamente ortogonales, entonces r_k y s_k son ortogonales y entonces tenemos que encontrar w y μ tales que minimicen

$$\|p_k\|_2^2 = \left(1 + \frac{\mu}{\alpha_{k-1}}\right)^2 \|r_k\|_2^2 + \|s_k\|_2^2$$

Tenemos por hipótesis de inducción que z_{k-1} minimiza el problema de mínimos cuadrados

$$r_{k-1} - AP_{k-1} w$$

con residual mínimo p_{k-1} *y entonces se deduce que* s_k debe ser un múltiplo de p_{k-1}, por lo tanto, p_k es combinación lineal de r_k y p_{k-1}, es decir,

$$p_k \in span\{p_{k-1}, r_k\}$$

que es lo que queríamos demostrar. ■

Esto quiere decir que al hacer la proyección del residual r_k sobre

$$span\{Ap_0, Ap_1, \ldots, Ap_{k-1}\}^{\perp}$$

solo interviene al proyectar el vector p_{k-1}, esto es, podemos escribir

$$p_k = r_k + \beta_k p_{k-1}$$

con $\beta_k \in \mathbb{R}$. Para obtener la constante β_k notemos que por la A–conjugación de los $\{p_i\}$, tenemos que

$$0 = p_{k-1}^t A p_k = p_{k-1}^t A r_k + \beta_k p_{k-1}^t A p_{k-1}$$

lo que implica que

$$\beta_k = \frac{-p_{k-1}^t A r_k}{p_{k-1}^t A p_{k-1}}$$

Con esto tenemos todos los elementos para presentar una primera versión del algoritmo de Gradiente Conjugado:

$x_0 =$ punto inicial
$r_0 = -b - Ax_0$
$k = 0$
while $r_k \neq 0$
  if $k = 0$
    $p_0 = r_0$
  else
    $\beta_k = \frac{-p_{k-1}^t A r_k}{p_{k-1}^t A p_{k-1}}$
    $p_k = r_k + \beta_k p_{k-1}$
  end
  $\alpha_k = r_k^t p_k / p_k^t A p_k$
  $x_{k+1} = x_k + \alpha_k p_k$
  $r_{k+1} = -b - Ax_{k+1}$
  $k = k + 1$
end

En esta implementación el método requiere de tres multiplicaciones matriz-vector por separado. Sin embargo, calculando los residuales vía $r_k = r_{k-1} - \alpha_{k-1}Ap_{k-1}$, tenemos que

$$r_k^t r_k = r_k^t r_{k-1} - \alpha_{k-1} r_k^t A p_{k-1} = -\alpha_{k-1} r_k^t A p_{k-1}$$

Por otro lado, tenemos que $p_k = r_k + \beta_k p_{k-1}$ y utilizando el hecho de que el residual es ortogonal a las direcciones anteriores tenemos que

$$r_k^t p_k = r_k^t r_k + \beta_k r_k^t p_{k-1}$$

lo que implica que

$$r_k^t p_k = r_k^t r_k \tag{3.12}$$

Entonces, tenemos que

$$Ap_{k-1} = \frac{1}{\alpha_{k-1}}(r_{k-1} - r_k)$$

de donde obtenemos que

$$\begin{aligned} p_{k-1}^t A p_{k-1} &= \frac{1}{\alpha_{k-1}}(p_{k-1}^t r_{k-1} - p_{k-1}^t r_k) \\ &= \frac{1}{\alpha_{k-1}}(p_{k-1}^t r_{k-1}) \\ &= \frac{1}{\alpha_{k-1}}(r_{k-1}^t r_{k-1}) \end{aligned}$$

Esto nos ayudará a simplificar el cálculo de β_k. Tenemos que

$$\begin{aligned} \beta_k &= \frac{-p_{k-1}^t A r_k}{p_{k-1}^t A p_{k-1}} \\ &= \frac{\frac{r_k^t r_k}{\alpha_{k-1}}}{\frac{r_{k-1}^t r_{k-1}}{\alpha_{k-1}}} \\ &= \frac{r_k^t r_k}{r_{k-1}^t r_{k-1}} \end{aligned}$$

esto da lugar a un algoritmo mas económico ya que eliminamos dos multiplicaciones

matriz–vector.

$x_0 =$ punto inicial
$r_0 = -b - Ax_0$
$k = 0$
while $r_k \neq 0$
    if $k = 0$
        $p_0 = r_0$
    else
        $\beta_k = r_k^t r_k / r_{k-1}^t r_{k-1}$
        $p_k = r_k + \beta_k p_{k-1}$
    end
    $\alpha_k = r_k^t r_k / p_k^t A p_k$
    $x_{k+1} = x_k + \alpha_k p_k$
    $r_{k+1} = -b - Ax_{k+1}$
    $k = k + 1$
end

Este es esencialmente el algoritmo de Gradiente Conjugado que aparece originalmente en el artículo de Hestenes y Stiefel (1952).

Cuando se aplica el método de gradiente conjugado, usualmente n es muy grande y $O(n)$ iteraciones representa una cantidad inaceptable de trabajo. Como consecuencia, veremos el método como una técnica iterativa con un criterio de paro basado en un máximo de iteraciones k_{max} y la norma del residual. Esto da lugar al siguiente algoritmo práctico.

Algoritmo de Gradiente Conjugado

$$
\begin{array}{l}
x = \text{ punto inicial} \\
r = -b - Ax \\
\rho_0 = \|r\|^2 \\
k = 0 \\
\text{while } (\sqrt{\rho_k} > \epsilon\|b\|_2) \bigwedge (k < k_{max}) \\
\quad \text{if } k = 0 \\
\quad\quad p = r \\
\quad \text{else} \\
\quad\quad \beta_k = \frac{\rho_k}{\rho_{k-1}} \\
\quad\quad p = r + \beta_k p \\
\quad \text{end} \\
\quad w = Ap \\
\quad \alpha_k = \rho_k / p^t w \\
\quad x = x + \alpha_k p \\
\quad r = r - \alpha_k w \\
\quad \rho_{k+1} = \|r\|_2 \\
\quad k = k + 1 \\
\text{end}
\end{array}
\tag{3.13}
$$

Este algoritmo requiere sólo una multiplicación matriz–vector. Nótese que solo necesitamos almacenar cuatro vectores esenciales, x, r, p, w.

El punto de vista iterativo es útil pero la razón de convergencia es central para el éxito del método.

3.3.1 Propiedades de convergencia

Examinaremos la convergencia de las iteraciones de gradiente conjugado $\{x_k\}$. Daremos dos resultados los cuales dicen que el método funciona bien cuando A es cercana a la identidad ya sea en el sentido de una perturbación de rango pequeño o en el sentido de norma.

Teorema. *Si $A = I + B$ es una matriz simétrica positiva definida y rango$(B) = r$ entonces el algoritmo (3.13) converge en a lo más $r + 1$ pasos.*

Demostración.

La dimensión de

$$span\{r_0, Ar_0, \ldots, A^k r_0\} = span\{r_0, Br_0, \ldots, B^k r_0\}$$

no puede exceder $r+1$. Como $p_0, \ldots, p_k$ generan este subespacio y son independientes, la iteración no puede avanzar mas alla de $r+1$ pasos. ■

Un resultado importante se desprende de este teorema.

Si A es cercana a una corrección de rango r a la identidad, entonces el algoritmo (3.13) converge en a lo más $r+1$ pasos.

Se puede obtener una cota del error distinta en términos de la A–norma definida como

$$\|w\|_A = \sqrt{w^t A w}$$

Teorema. *Suponga que $A \in \mathbb{R}^{n\times n}$ es simétrica y positiva definida y $b \in \mathbb{R}^n$. Si el algoritmo (3.13) produce iterandos $\{x_k\}$ y $\kappa = \kappa_2(A)$ entonces*

$$\|x - x_k\|_A \leq 2\|x - x_0\|_A \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \tag{3.14}$$

La demostración se puede consultar en [26].

Esto conduce a al resultado de que el método de gradiente conjugado converge muy rápido en la A–norma si $\kappa_2(A) \approx 1$, recordemos que $\kappa_2(A) = \|A\|_2\|A^{-1}\|_2 = \lambda_{\max}/\lambda_{\min}$ donde $\lambda_{\max}$ y $\lambda_{\min}$ son el mas grande y mas pequeño eigenvalores de A. Entonces si A está bien condicionada, esto es, $\kappa_2(A) \approx 1$ la convergencia va a ser muy rápida, pero si $\kappa_2(A)$ es muy grande, A mal condicionada, entonces el algoritmo va a ser muy lento.

3.3.2 Precondicionamiento

La cota (3.14) sugiere que el algoritmo de gradiente conjugado puede acelerarse transformando el problema para mejorar la condición de A. Esto puede hacerse por medio del cambio de variable

$$\hat{x} = Cx$$

donde C es simétrica y positiva definida, esto conduce a resolver la ecuación cuadrática transformada

$$\hat{\Psi}(\hat{x}) = \frac{1}{2}\hat{x}^t(C^{-1}AC^{-1})\hat{x} - (C^{-1}b)^t\hat{x} \tag{3.15}$$

La razón de convergencia del método de gradiente conjugado aplicado a esta cuadrática está en función de $\kappa_2(C^{-1}AC^{-1})$ en lugar de $\kappa_2(A)$. Entonces, precondicionamos el problema si escogemos una matrix C tal que $(C^{-1}AC^{-1})$ esté mejor condicionada que A.

No es necesario conocer la transformación explícitamente, se escribe el algoritmo de gradiente conjugado para minimizar (3.15) en términos de las variables $\hat{x}$ y entonces se transforman las iteraciones obtenidas a las variables originales x. Enseguida presentamos el método de gradiente conjugado Precondicionado, donde M usualmente es llamada el precondicionador, y está dada por $M = C^2$ y es simétrica y positiva definida.

Algoritmo de Gradiente Conjugado Lineal Precondicionado

$$
\begin{array}{l}
x_0 = \text{ punto inicial} \\
M = \text{ precondicionador} \\
r_0 = -b - Ax_0 \\
k = 0 \\
\text{while } (r_k \neq 0) \\
\quad \text{Resolver } Mz_k = r_k \\
\quad \text{if } k = 0 \\
\quad\quad p_0 = z_0 \\
\quad \text{else} \\
\quad\quad \beta_k = r_k^t z_k / r_{k-1}^t z_{k-1} \\
\quad\quad p_k = z_k + \beta_k p_{k-1} \\
\quad \text{end} \\
\quad \alpha_k = r_k^t z_k / p_k^t A p_k \\
\quad x_{k+1} = x_k + \alpha_k p_k \\
\quad r_{k+1} = r_k - \alpha_k A p_k \\
\quad k = k + 1 \\
\text{end}
\end{array}
\tag{3.16}
$$

Sobre este procedimiento podemos hacer algunas observaciones:

- Se puede demostrar que los residuales y las direcciones de búsqueda satisfacen

$$r_j^t M^{-1} r_i = 0, \qquad i \neq j$$

$$p_j^t (C^{-1} A C^{-1}) p_i = 0, \qquad i \neq j$$

- Los denominadores $r_{k-1}^t z_{k-1} = z_{k-1}^t M z_{k-1}$ no puden anularse ya que M es positiva definida.
- C no actúa directamente en el algoritmo, solo necesitamos M.
- Para que el algoritmo sea una técnica efectiva para matrices sparse, los sistemas lineales de la forma $Mz = r$ deben ser fáciles de resolver.

La elección de un buen precondicionador es crucial para determinar la razón de convergencia del método.

3.4 Método de Gradiente Conjugado de Beale

Se ha observado en la práctica que es posible iniciar con otra dirección distinta de $-r_0$ el método de direcciones conjugadas, Beale [10] propone un método de Gradiente Conjugado donde la dirección inicial es una dirección de descenso distinta a la opuesta al gradiente.

La técnica de Beale permite iniciar con cualquier dirección de descenso. La idea es iniciar con cualquier dirección de descenso, pero que el método conserve la propiedad de convergencia en n pasos para cuadráticas. La técnica de Beale consiste en lo siguiente: Consideremos p_t una dirección de descenso arbitraria, $f(x)$ una cuadrática y la dirección p_{k+1}, $(k+1 > t)$, como una combinación lineal de p_t y los gradientes $g_{t+1}, g_{t+2}, \ldots, g_{k+1}$, entonces cómo debe ser la combinación lineal para que las direcciones de búsqueda sean conjugadas? Beale [10] demuestra que p_{k+1} tiene que ser de la forma

$$p_{k+1} = -r_{k+1} + \beta_{k+1} p_k + \gamma_k p_t$$

Esto es, la dirección de Gradiente Conjugado que ya teníamos mas un múltiplo de la dirección p_t. β_{k+1} y γ_k se escogen de manera que p_{k+1} sea conjugado a p_k y p_t. La propuesta es considerar el siguiente algoritmo de Gradiente Conjugado.

Algoritmo de Gradiente Conjugado de Beale
Escoger un punto inicial x_0
evaluar r_0
escoger p_0 una dirección de descenso
sea $k = 0$, $t = 0$.
while $r_k \neq 0$
calcular α_k
$x_{k+1} = x_k + \alpha_k p_k$
Evaluar r_{k+1}
$$\beta_{k+1} = \frac{r_{k+1}^t (r_{k+1} - r_k)}{p_k^t (r_{k+1} - r_k)}$$
if $k = t$ o $t = 0$
$\gamma_k = 0$
else
$$\gamma_k = \frac{r_{k+1}^t (r_{t+1} - r_t)}{p_k^t (r_{t+1} - r_t)}$$
end
$p_{k+1} = -g_{k+1} + \beta_{k+1} p_k + \gamma_k p_t$
$k = k + 1$
end while

Referencias

G. Golub, Ch Van Loan (1989). *Matrix Computations* The Johns Hopkins University Press, 2nd. Edition.

C. Kelley (1995) *Iterative Methods for linear and non linear equation* SIAM Publications.

J. Nocedal and J. Wright (1999). *Numerical Optimization*, Springer Series in Operations Research.

Capítulo 4

Mínimos de cuadráticas restringidos a una región

Una familia de métodos de optimización se basan en la construcción de aproximaciones cuadráticas en la vecindad de un punto, y entonces lo que hacen es tomar como aproximaciones al óptimo de la función el mínimo de la cuadrática, debido a que las aproximaciones son locales es necesario hacer algunas restricciones y en particular una idea muy exitosa es la de elegir un punto donde la aproximación cuadrática sea lo mas pequeña en una vecindad del punto en consideración. Estos métodos son la base de técnicas conocidas como Región de Confianza que se estudiarán mas adelante, es por esto que nos interesa poder resolver el problema de minimizar una cuadrática dentro de una vecindad. En este capítulo presentamos algunos algoritmos efectivos para resolver este problema y los resultados teóricos que los respaldan.

4.1 Problema

El problema al que nos enfocaremos en este capítulo es: minimizar una función cuadrática restringida a una vecindad, esto es, queremos encontrar

$$\min_{\|w\| \leq \Delta} \Psi(w) \tag{4.1}$$

con $\Psi(w) = \frac{1}{2} w^t B w + g^t w$, $\Delta \geq 0$ y B simétrica.

Lo primero que haremos es caracterizar la solución de nuestro problema para después resolverlo de manera práctica.

4.2 Caracterización de la solución

Antes de enunciar el resultado que caracteriza la solución de (4.1) veremos dos ejemplos de funciones cuadráticas para $n = 2$ y encontraremos el mínimo restringido a una región, para dar una visualización geométrica en el plano y después enunciar el resultado en general.

Ejemplo. Supongamos que queremos calcular el mínimo de la cuadrática

$$\Psi(w) = g^t w + \frac{1}{2} w^t B w$$

restringido a la región $\{w \in \mathbb{R}^2 : \|w\| \leq \delta\}$.

Para ver cómo son las soluciones debemos de considerar 2 casos, si la matriz es positiva definida o no lo es.

Supongamos que la matriz B es positiva definida, entonces, la cuadrática tiene un único mínimo sin restricciones, $-B^{-1}g$ si este mínimo sin restricciones está dentro de la región considerada, entonces este es el punto que resuelve nuestro problema.

Supongamos que no es así, esto es que $\|B^{-1}g\| > \Delta$. Como $B > 0$ entonces las curvas de nivel de esta función serán elipses, conforme los ejes de las elipses aumenten, aumentará el valor de la curva de nivel a la que correspondan, entonces, el punto mínimo de la cuadrática en la región, será el primer punto en donde una curva de nivel toque la región, esto es, si partimos del mínimo de la cuadrática sin restricciones y vamos recorriendo las curvas de nivel, existirá una primera curva de nivel que sea tangente a la región, el punto de tangencia será el punto que estamos buscando ya que para los siguientes puntos les corresponderá una curva de nivel de valor más grande. Geométricamente lo que tenemos en este caso es lo que se aprecia en la figura 4.1.

Consideremos ahora el caso de B indefinida. En este caso la cuadrática ya no tiene mínimo, las curvas de nivel ahora son hiperbólas las cuales crecen de un lado y decrecen del lado contrario, igual que en el caso anterior si seguimos las curvas de nivel hacia donde decrecen hay un momento donde una curva de nivel es tangente a la región donde queremos minimizar la cuadrática, entonces para la curva de nivel de valor mas pequeño que sea tangente a la región, el punto de tangencia es el punto solución de nuestro

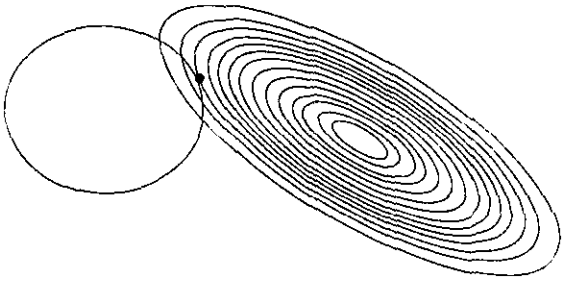


Figura 4.1: Caso B Positiva Definida

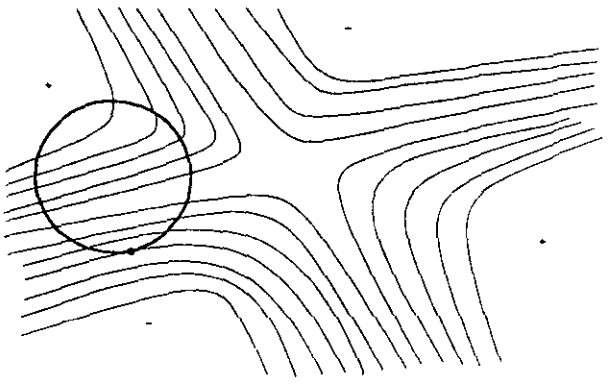


Figura 4.2: Caso de B indefinida

problema, en este caso geométricamente lo que tenemos es lo que aparece en la figura 4.2.

Entonces si la matriz es positiva definida calculamos el mínimo sin restricciones, si éste cae dentro de la región, entonces es la solución que buscamos, si no es así, entonces la solución se encuentra en la frontera de la región y si la matriz es indefinida, también la solución la encontraremos en la frontera de la región. Esto nos ayudará a entender el caso n–dimensional.

■

La caracterización de la solución de (4.1) para el caso general es nuestro siguiente resultado.

Lema . *Sea* $\Psi : \mathbb{R}^n \to \mathbb{R}$ *una función cuadrática y sea* $\Delta > 0$ *dado. Un punto* $p \in \mathbb{R}^n$ *resuelve el problema*

$$\min_{\|w\| \leq \Delta} \Psi(w)$$

si y sólo si $\|p\| \leq \Delta$ *y existe un* $\lambda \geq 0$ *tal que*

$$(B + \lambda I)p = -g \qquad (a)$$
$$\lambda(\Delta - p) = 0 \qquad (b)$$
$$(B + \lambda I) \text{ positiva semidefinida} \qquad (c)$$

Demostración.

$\Longleftarrow$) Supongamos que existen $p^* \in \mathbb{R}^n$ y $\lambda \geq 0$ tales que (a), (b) y (c) se satisfacen. De (a) y (c) tenemos que por la caracterización de mínimos de cuadráticas sin restricciones p^* es un mínimo global de la cuadrática

$$\begin{aligned} \hat{\Psi}(w) &= g^t p + \frac{1}{2} w^t (B + \lambda I) w \\ &= \Psi(w) + \frac{\lambda}{2} w^t w \end{aligned}$$

Como $\hat{\Psi}(p) \geq \hat{\Psi}(p^*)$ ya que p^* es un mínimo, entonces

$$\begin{aligned} & & \hat{\Psi}(p) &\geq \hat{\Psi}(p^*) \\ \Rightarrow & & \Psi(p) + \frac{\lambda}{2} p^t p &\geq \Psi(p^*) + \frac{\lambda}{2} p^{*t} p^* \\ & \Rightarrow & \Psi(p) &\geq \Psi(p^*) + \frac{\lambda}{2}(p^{*t} p^* - p^t p) \end{aligned}$$

De la última desigualdad se tiene que si $\lambda = 0$ entonces p^* es un mínimo. Veremos entonces el caso con $\lambda > 0$.

Se tiene que $\lambda(\Delta - \|p^*\|) = 0$,

$$\Rightarrow \lambda(\Delta^2 - p^{*t} p^*) = 0$$

Entonces como $\lambda > 0$, se tenemos que $p^{*t} p^* = \Delta^2$,

$$\Rightarrow \Psi(p) \geq \Psi(p^*) + \frac{\lambda}{2}(\Delta^2 - p^t p)$$

Como tenemos que $\|p\| \leq \Delta$, entonces $\Delta^2 - p^t p \geq 0$

$$\Rightarrow \Psi(p) \geq \Psi(p^*) \qquad \forall p \text{ tal que } \|p\| \leq \Delta$$

entonces p^* es un mínimo de Ψ en la región dada.

$\Longrightarrow$) Supongamos que p^* es una solución global de (4.1), mostraremos que existe un $\lambda \geq 0$ que satisface (a), (b) y (c).

En el caso $\|p^*\| \leq \Delta$, p^* es un mínimo sin restricciones de Ψ y se tiene

$$\nabla\Psi(p^*) = Bp^* + g = 0 \quad \Rightarrow \quad Bp^* = -g$$

y

$$\nabla^2\Psi(p^*) = B \geq 0$$

Entonces (a), (b) y (c) se satisfacen con $\lambda = 0$.

Resta entonces el caso $\|p^*\| = \Delta$. (b) se satisface inmediatamente y se tiene que p^* resuelve el problema

$$\min_{\|w\|=\Delta} \Psi(w)$$

De la teoría de minimización sin restricciones se tiene que existe un λ tal que la función Lagrangiana definida por

$$L(w, \lambda) = \Psi(w) + \frac{\lambda}{2}(w^t w - \Delta^2)$$

tiene un punto crítico en p^*, esto es,

$$\nabla L(w, \lambda) = \begin{pmatrix} \nabla L_w \\ \nabla L_\lambda \end{pmatrix} = \begin{pmatrix} Bp^* + g + \lambda p^* \\ \frac{1}{2}(p^{*t} p^* - \Delta^2) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

De la primera componente de este vector se tiene

$$Bp^* + \lambda p^* = -g$$

$$\Rightarrow (B + \lambda I)p^* = -g$$

y se tiene (a).

Sólo falta mostrar (c). Como p^* es un mínimo se tiene que

$$\Psi(p) \geq \Psi(p^*) \qquad \forall p \text{ tal que } p^t p = p^{*t} p^* = \Delta^2$$

entonces para tales vectores p se tiene

$$p^{*t}p^* - p^t p = 0$$

$$\Rightarrow \Psi(p) \geq \Psi(p^*) + \frac{\lambda}{2}(p^{*t}p^* - p^t p)$$

sustituyendo g en esta última ecuación y reordenando se obtiene

$$\frac{1}{2}(p - p^*)^t(B + \lambda I)(p - p^*) \geq 0$$

Ahora, esto se satisface para cualquier vector p tal que $\|p\| = \Delta$, y el conjunto

$$\left\{ w : w = \pm \frac{p - p^*}{\|p - p^*\|} \text{ para algún } p \text{ con } \|p\| = \Delta \right\}$$

es un conjunto denso en la esfera unitaria. Cualquier vector en $\mathbb{R}^n$ puede expresarse como un múltiplo de $p-p^*$ para alguna p con $\|p\| = \Delta$, entonces se tiene que $(B + \lambda I) \geq 0$ y queda demostrado (c).

Sólo falta mostrar que $\lambda \geq 0$. Como (a) y (c) se satisfacen en p^* entonces se tiene que p^* minimiza $\tilde{\Psi}$ y tenemos que

$$\Psi(p) \geq \Psi(p^*) + \frac{\lambda}{2}(p^{*t}p^* - p^t p) \tag{4.2}$$

Supongamos que sólo existen $\lambda < 0$ que satisfacen (a) y (c). Entonces de (4.2) se tiene que

$$\Psi(p) \geq \Psi(p^*)$$

cuando $\|p\| \geq \|p^*\| = \Delta$ ya que $p^{*t}p^* - p^t p < 0$ entonces $\frac{\lambda}{2}(p^{*t}p^* - p^t p) > 0$ y entonces

$$\Psi(p) \geq \Psi(p^*) + \alpha$$

con $\alpha > 0$, entonces se tiene que

$$\Psi(p) \geq \Psi(p^*)$$

ahora, ya teníamos que p^* minimiza Ψ para $\|p\| \leq \Delta$ y con la ecuación anterior se tiene que p^* es un mínimo global sin restricciones de Ψ. Entonces se tiene que

$$Bp = -g, \qquad B \geq 0$$

Entonces (a) y (c) se satisfacen para $\lambda = 0$ lo cual contradice la suposición de que sólo valores negativos de λ satisfacían estas condiciones. Por lo tanto $\lambda \geq 0$. ■

Ya tenemos entonces la caracterización de la solución a nuestro problema. Veremos ahora qué propiedades satisface nuestra solución para entonces ver la forma práctica de resolverlo.

4.3 Propiedades de la Solución

De acuerdo al Lema de la sección anterior, p^* es solución de (4.1) si y sólo si existe un $\lambda \geq 0$ tal que

$$\begin{array}{rcll} (B+\lambda I)p^* & = & -g & \text{(a)} \\ \lambda(\Delta - \|p^*\|) & = & 0 & \text{(b)} \\ (B+\lambda I) & \geq & 0 & \text{(c)} \end{array}$$

La condición (b) establece que al menos una de las dos cantidades no negativas debe ser cero. Cuando la solución cae estrictamente dentro de la región, debemos tener $\lambda = 0$ y entonces $Bp^* = -g$ y $B > 0$ de (a) y (c). En el otro caso, tenemos $\|p^*\| = \Delta$ y λ puede tomar un valor positivo.

Entonces de acuerdo al Lema necesitamos un algoritmo para encontrar la solución p de (4.1). Definimos

$$p(\lambda) = -(B+\lambda I)^{-1}g$$

y buscamos el valor $\lambda \geq 0$ tal que $\|p(\lambda)\| = \Delta$.

Para ver que existe un λ con estas propiedades, veremos la descomposición en eigenvalores de B y estudiaremos propiedades de $\|p(\lambda)\|$.

Como B es simétrica existe Q ortogonal y Λ diagonal tales que $B = Q\Lambda Q^t$, donde

$$\Lambda = \text{ diag } (\lambda_1, \lambda_2, \ldots, \lambda_n)$$

con $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ eigenvalores de B. Entonces tenemos que

$$\begin{array}{rcl} B+\lambda I & = & Q\Lambda Q^t + \lambda I \\ & = & Q\Lambda Q^t + \lambda I Q Q^t \\ & = & Q(\Lambda + \lambda I)Q^t \end{array}$$

y si $\lambda \neq \lambda_j$, tenemos

$$\begin{array}{rcl} p(\lambda) & = & -(B+\lambda I)^{-1}g \\ & = & -Q(\Lambda+\lambda I)^{-1}Q^t g \\ & = & -\displaystyle\sum_{j=1}^{n} \frac{q_j^t g}{\lambda_j + \lambda} q_j \end{array}$$

con q_j la j–ésima columna de Q. Entonces

$$\|p(\lambda)\|^2 = \sum_{j=1}^{n} \frac{(q_j^t g)^2}{(\lambda_j + \lambda)^2} \tag{4.3}$$

Analizaremos esta expresión. Nótese que si $\lambda > -\lambda_1$ entonces $\lambda_j + \lambda \geq 0$ para toda $j = 1, \ldots, n$ y $\|p(\lambda)\|$ es una función continua, decreciente en el intervalo $(-\lambda_1, \infty)$. De hecho, tenemos

$$\lim_{\lambda \to \infty} \|p(\lambda)\| = 0 \tag{4.4}$$

Nótese también que cuando $q_1^t g \neq 0$, entonces

$$\lim_{\lambda \to -\lambda_j} \|p(\lambda)\| = \infty \tag{4.5}$$

Geométricamente, en este caso tenemos lo que aparece en la figura 4.3.

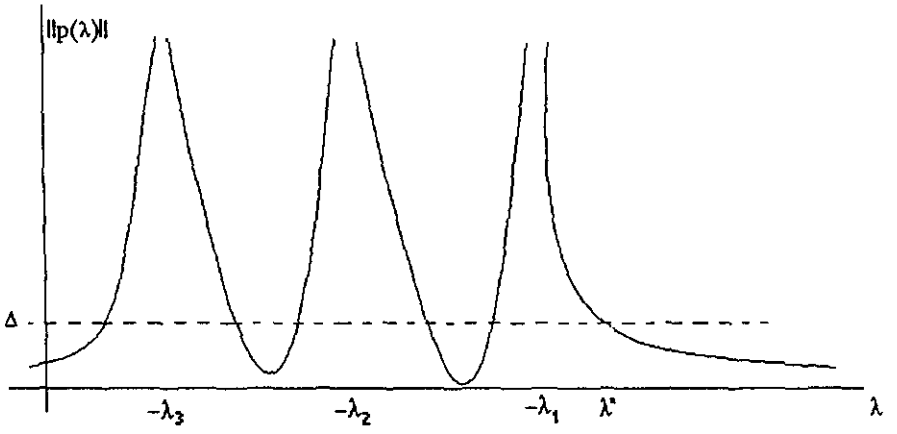


Figura 4.3: Gráfica de $\|p(\lambda)\|$

Como la función es continua y decreciente, $\|p(\lambda)\|$ alcanza un valor de Δ en exactamente un punto en el intervalo $(-\lambda_1, \infty)$.

Veamos cómo es la trayectoria de p conforme varía λ. Cuando $B > 0$, tenemos que

$$p(\lambda) = -(B + \lambda I)^{-1} g$$

Nótese que si $\lambda = 0$ entonces $p(\lambda)$ es el mínimo de la cuadrática sin restricciones, denotaremos este punto por $p^B = -B^{-1}g$, entonces si hacemos tender λ a cero, $p(\lambda)$ tenderá a p^B, esto es,

$$\lim_{\lambda \to 0} p(\lambda) = -B^{-1} g = p^B$$

Ahora, conforme λ aumente $p(\lambda)$ tenderá a cero, esto se ve si consideramos que p satisface

$$(B + \lambda I)p = -g$$

dividiendo por λ esta ecuación resulta

$$\left(\frac{B}{\lambda} + I\right) p = -\frac{g}{\lambda}$$

entonces si λ tiende a infinito, $p(\lambda)$ tiende a cero acercándose de forma tangente sobre la dirección de $-g$. Entonces podemos trazar la trayectoria curva de $p(\lambda)$ partiendo de $\lambda = 0$ y haciendo tender λ a infinito como se ilustra en la figura 4.4.

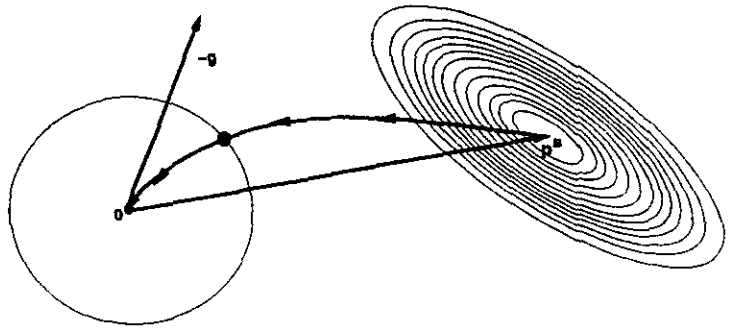


Figura 4.4: p en función de λ

Ahora, esta misma trayectoria pero en sentido contrario puede obtenerse de la siguiente manera: Consideremos ahora a p como función del radio de la región de minimización, Δ. Si $\Delta = 0$ entonces $p = 0$ es la solución de (4.1) y si Δ aumenta, llegará un momento en que la región de restricción contendrá a p^B, el mínimo sin restricciones, esto es, para $\Delta \geq \|p^B\|$ la solución de (4.1) es p^B, entonces la trayectoria de p como función de Δ partiendo de $\Delta = 0$ y haciendo tender Δ a infinito es la que se ilustra en la figura 4.5.

Entonces si $B > 0$ y $\|B^{-1}g\| \leq \Delta$ tendremos que $\lambda = 0$ es el punto buscado y no hay necesidad de hacer búsqueda. Si $B > 0$ y $\|B^{-1}g\| > \Delta$ entonces existe un $\lambda > 0$ para el cual $\|p(\lambda)\| = \Delta$ y se busca la solución en el intervalo $(0, \infty)$.

Si B es indefinida y $q_1^t q \neq 0$ (4.4) y (4.5) garantizan que podemos encontrar una solución en el intervalo $(-\lambda_1, \infty)$.

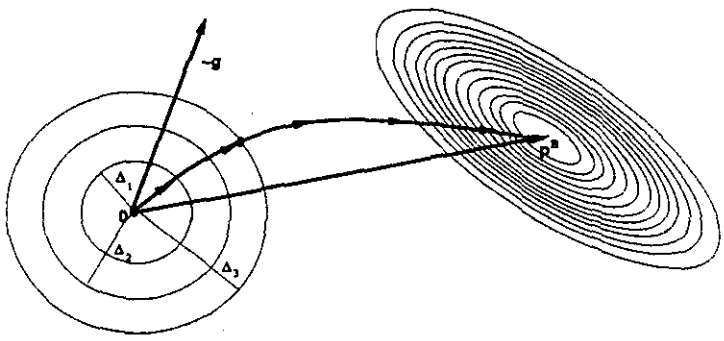


Figura 4.5: p en función de Δ

4.3.1 El caso duro

En la discusión anterior supusimos que $q_1^t g \neq 0$ para el caso B indefinida. Lo anterior también as aplicable aún cuando el eigenvalor más negativo de B sea un eigenvalor múltiple, esto es, $0 > \lambda_1 = \lambda_2 = \cdots$ siempre que $q_i^t g \neq 0$ para al menos uno de los índices i para los cuales $\lambda_i = \lambda_1$.

Cuando $q_i^t g = 0$ la situación se complica ya que el límite de (4.5) no se satisface para $\lambda_j = \lambda_1$ y puede no existir un valor $\lambda \in (-\lambda_1, \infty)$ tal que $\|p(\lambda)\| = \Delta$. Esto es

$$\|(B + \lambda I)^{-1} g\| = \Delta$$

no tiene solución con $B + \lambda I$ positiva definida. Moré y Sorensen [32] se refieren a este caso como el caso duro.

La gráfica es este caso es la figura 4.6.

La dificultad característica del caso duro es que $\|p(\lambda)\| < \Delta$ cuando $(B + \lambda I)$ es positiva definida. Por ejemplo, si consideramos

$$B = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad g = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Se tiene que B no es positiva definida, $\lambda_1 = -1$, ahora si $B + \lambda I$ es positiva definida, entonces $\|p(\lambda)\| \leq 1/2$.

Ahora no existe solución para λ en el intervalo abierto $(-\lambda_1, \infty)$. Pero el Lema garantiza que existe solución en el intervalo $[-\lambda_1, \infty)$ entonces solo queda una posibilidad $\lambda = -\lambda_1$.

Notemos que $(B - \lambda_1 I)$ es singular, entonces existe un z tal que $(B - \lambda_1 I)z = 0$. De hecho, z es un eigenvector de B asociado al eigenvalor λ_1. En el caso duro podemos

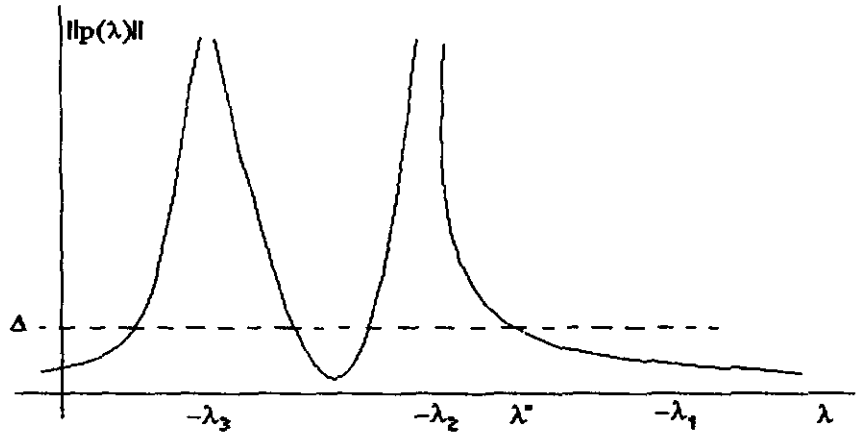


Figura 4.6: Caso Duro: $\lambda^* < -\lambda_1$

obtener una solución resolviendo

$$(B - \lambda_1 I)p = -g$$

para p con $\|p\| \leq \Delta$ y determinando un vector $z \in S_1$, donde

$$S_1 = \{z : Bz = \lambda_1 z, z \neq 0\}$$

Entonces

$$(B - \lambda_1 I)(p + \tau z) = -g$$

y $\|p + \tau z\| = \Delta$ para alguna τ. Por el Lema se tiene que $(p + \tau z)$ resuelve (4.1).

Nótese que la solución siempre tiene la forma

$$p + \tau z$$

en el caso duro $z \in S_1$ y si no se está en este caso $z = 0$. Ahora, en el caso duro es necesario conocer z, eigenvector de B asociado a λ_1 para resolver nuestro problema; determinar z puede requerir mas esfuerzo computacional que el necesitado para resolver nuestro problema de optimización, además se requiere tiempo para reconocer que la solución está dada de esta forma. Entonces los algoritmos prácticos van a tratar de evitar el cálculo del eigenvector z de forma exacta, en su lugar tratarán de aproximarlo de forma económica. La aproximación a z se hace de tal manera que se encuentre una solución aproximada a (4.1) en el sentido de que esté en una vecindad de la frontera de la bola y tal que el valor de la cuadrática no diste mucho del valor

de la cuadrática en el óptimo, esto es, trataremos de encontrar una solución de la forma $p + z$ tal que si $0 < \sigma < 1$, entonces

$$|\Psi(p+z) - \Psi^*| \leq \sigma|\Psi^*| \tag{4.6}$$

donde Ψ^* es el valor de la cuadrática en el óptimo. De esta forma $p + z$ estará cerca de la solución óptima de (4.1).

El siguiente resultado establece cuáles son las condiciones para z de manera que satisfaga (4.6).

Lema. *Sea* $0 < \sigma < 1$ *dado y suponga que*

$$B + \lambda I = R^t R, \qquad (B + \lambda I)p = -g, \qquad \lambda \geq 0 \tag{4.7}$$

Sea $z \in \mathbb{R}^n$ *que satisface*

$$\|p + z\| = \Delta, \qquad \|Rz\|^2 \leq \sigma(\|Rp\|^2 + \lambda\Delta^2) \tag{4.8}$$

Entonces

$$\begin{aligned} -\Psi(p+z) &\geq (1/2)(1-\sigma)(\|Rp\|^2 + \lambda\Delta^2) \\ &\geq (1-\sigma)|\Psi^*| \end{aligned}$$

donde Ψ^* *es el valor óptimo de (1.1).*

Demostración.

Notemos primero que para cualquier $z \in \mathbb{R}^n$

$$\Psi(p+z) = g^t(p+z) + \frac{1}{2}(p+z)^t B(p+z)$$

por otro lado, tenemos que

$$\begin{aligned} g^t(p+z) &= -p^t(B+\lambda I)(p+z) \\ &= -p^t R^t R(p+z) \\ &= -p^t R^t Rp - p^t R^t Rz \\ &= -\|Rp\|^2 - p^t R^t Rz \end{aligned}$$

Entonces

$$\begin{aligned}\frac{1}{2}(p+z)^tB(p+z) &= \frac{1}{2}(p+z)^tB(p+z)+\frac{1}{2}\lambda I(p+z)^t(p+z)-\frac{1}{2}\lambda I(p+z)^t(p+z)\\ &= \frac{1}{2}(p+z)^t(B+\lambda I)(p+z)-\frac{1}{2}\lambda\|p+z\|^2\\ &= \frac{1}{2}(p^tR^tRp+z^tR^tRp+p^tR^tRz+z^tR^tRz)-\frac{1}{2}\lambda\|p+z\|^2\\ &= \frac{1}{2}(\|Rp\|^2+2p^tR^tRz+\|Rz\|^2)-\frac{1}{2}\lambda\|p+z\|^2\end{aligned}$$

lo que implica que

$$\begin{aligned}\Psi(p+z) &= -\|Rp\|^2-p^tR^tRz+\frac{1}{2}(\|Rp\|^2+2p^tR^tRz+\|Rz\|^2)-\frac{1}{2}\lambda\|p+z\|^2\\ &= -\frac{1}{2}(\|Rp\|^2+\lambda\|p+z\|^2)+\frac{1}{2}\|Rz\|^2 \qquad (4.9)\end{aligned}$$

Ahora, por (4.8) tenemos que

$$\begin{aligned}\Psi(p+z) &\leq -\frac{1}{2}(\|Rp\|^2+\lambda\Delta^2)+\frac{1}{2}\sigma(\|Rp\|^2+\lambda\Delta^2)\\ &= \frac{1}{2}(\sigma-1)(\|Rp\|^2+\lambda\Delta^2)\end{aligned}$$

Esto implica que

$$-\Psi(p+z)\geq\frac{1}{2}(1-\sigma)(\|Rp\|^2+\lambda\Delta^2) \qquad (4.10)$$

Si $\Psi^*=\Psi(p+z^*)$ con $\|p+z^*\|\leq\Delta$ entonces por (4.9) se tiene que

$$\begin{aligned}-\Psi(p+z^*) &= \frac{1}{2}(\|Rp\|^2+\lambda\|p+z^*\|^2)-\frac{1}{2}\|Rz^*\|^2\\ &\leq \frac{1}{2}(\|Rp\|^2+\lambda\Delta^2) \qquad (4.11)\end{aligned}$$

Entonces

$$|\Psi^*|\leq\frac{1}{2}(\|Rp\|^2+\lambda\Delta^2) \qquad (4.12)$$

Y tenemos de (4.10) y (4.12)

$$\begin{aligned}-\Psi(p+z) &\geq \frac{1}{2}(1-\sigma)(\|Rp\|^2+\lambda\Delta^2)\\ &\geq (1-\sigma)|\Psi^*|\end{aligned}$$

que es lo que queríamos demostrar. ■

Una consecuencia de este Lema es que

$$|\Psi(p+z) - \Psi^*| \leq \sigma|\Psi^*|$$

Esto es, si (4.8) se satisface entonces $(p+z)$ está cerca de la solución óptima de (4.1).

Esto nos caracteriza una solución aproximada en todos los casos, debemos buscar la solución de la forma $p+z$ con p que satisfaga (4.7), esto es, p tal que

$$B + \lambda I = R^t R, \qquad (B+\lambda I)p = -g, \qquad \lambda > 0$$

y z como en (4.8), es decir

$$\|p+z\| = \Delta, \qquad \|Rz\|^2 \leq \sigma(\|Rp\|^2 + \lambda\Delta^2)$$

En el caso no duro $z = 0$ satisface (4.8) y para el caso duro debemos encontrar z que satisfaga (4.8).

4.4 Criterios de Convergencia

Veremos ahora los criterios de convergencia para terminar el proceso con una solución aproximada. La idea es terminar cuando estemos muy cerca de la solución óptima de (4.1). Dados σ_1 y $\sigma_2 \in (0,1)$ y un $\lambda > 0$ tales que $B + \lambda I$ sea positiva definida, se calcula un vector $p = p(\lambda)$. Si

$$|\Delta - \|p\|| \leq \sigma_1\Delta \qquad \text{o} \qquad \|p\| \leq \Delta,\ \lambda = 0 \tag{4.13}$$

entonces $s = p$ es una solución aproximada. El caso duro se toma en cuenta calculando p y $\hat{z}$ cuando $\|p\| < \Delta$ y si

$$\|R(\tau\hat{z})\|^2 \leq \sigma_1(2-\sigma_1)\max\{\sigma_2, \|Rp\|^2 + \lambda\Delta^2\} \tag{4.14}$$

entonces $s = p + \tau\hat{z}$ se considera como solución aproximada.

Una prueba de convergencia adicional es que si ambos, (4.13) y (4.14) se satisfacen entonces se escoge la solución aproximada para la cual $\Psi(s)$ sea mas pequeña. Esto no es muy difícil de hacer ya que de (4.9) teníamos que

$$\Psi(p+z) = -\frac{1}{2}(\|Rp\|^2 + \lambda\|p+z\|^2) + \frac{1}{2}\|Rz\|^2$$

Entonces $\Psi(p+\tau\hat{z}) \leq \Psi(p)$ si y sólo si

$$-\frac{1}{2}(\|Rp\|^2 + \lambda\|p+\tau\hat{z}\|^2) + \frac{1}{2}\|R(\tau\hat{z})\|^2 \leq -\frac{1}{2}(\|Rp\|^2 + \lambda\|p\|^2)$$

$$\iff \quad -\lambda\|p+\tau\hat{z}\|^2 + \|R(\tau\hat{z})\|^2 \leq -\lambda\|p\|^2$$

$$\iff \quad \|R(\tau\hat{z})\|^2 \leq \lambda(\Delta^2 - \|p\|^2)$$

Como conocemos ambas cantidades, entonces elegimos como solución aquella para la cual $\Psi(s)$ sea de menor magnitud, en el caso que se satisfagan (4.13) y (4.14) al mismo tiempo.

Mostraremos ahora que (4.13) y (4.14) garantizan que si el algoritmo termina entonces la solución aproximada s satisface

$$\Psi(s) - \Psi^* \leq \sigma_1(2-\sigma_1)\max\{|\Psi^*|, \sigma_2\}, \qquad \|s\| \leq (1+\sigma_1)\Delta \tag{4.15}$$

y entonces s está cerca de la solución óptima de (4.1) en el sentido que lo establece el Lema de la sección anterior.

Consideremos primero (4.14). Si $\|Rp\|^2 + \lambda\Delta^2 < \sigma_2$ entonces se satisfacen las hipótesis del Lema anterior cuando σ se reemplaza por $\sigma_1(2-\sigma_1)$ y entonces tenemos que (4.15) se satisface para $s = p + \tau\hat{z}$.

Ahora supongamos que $\|Rp\|^2 + \lambda\Delta^2 \geq \sigma_2$ para ver que (4.15) se satisfacen en este caso primero note que si $\Psi^* = \Psi(p+z^*)$ donde $\|p+z^*\| \leq \Delta$ entonces (4.9) implica que

$$\begin{aligned}\Psi(p+z^*) &= -\frac{1}{2}(\|Rp\|^2 + \lambda\|p+z^*\|^2) + \frac{1}{2}\|Rz^*\|^2 \\ &\geq -\frac{1}{2}(\|Rp\|^2 + \lambda\Delta^2)\end{aligned}$$

entonces

$$\begin{aligned}|\Psi^*| &= -\Psi^* \\ &\leq \frac{1}{2}(\|Rp\|^2 + \lambda\Delta^2) \\ &\leq \frac{1}{2}\sigma_2\end{aligned}$$

Usando este resultado tenemos que

$$\begin{aligned}\Psi(s) &= -\frac{1}{2}(\|Rp\|^2 + \lambda\|p+\tau\hat{z}\|^2) + \frac{1}{2}\|R(\tau\hat{z})\|^2 \\ &\leq \Psi^* + \frac{1}{2}\|R(\tau\hat{z})\|^2 \\ &\leq \Psi^* + \frac{1}{2}\sigma_1(2-\sigma_1)\sigma_2 \qquad \text{(por 4.14)}\end{aligned}$$

entonces

$$\Psi(s) - \Psi^* \leq \sigma_1(2 - \sigma_1)\max\{|\Psi^*|, \sigma_2\}$$

Y entonces (4.15) también se satisface en este caso.

El siguiente lema muestra que cuando se cumple (4.13) entonces (4.15) también se satisface con $s = p$.

Lema. *Sea* $0 < \sigma < 1$ *dado y suponga que*

$$B + \lambda I = R^t R, \qquad (B + \lambda I)p = -g, \qquad \lambda \geq 0$$

Si Ψ^* *es el valor óptimo de (4.1) y si*

$$\|p\| \geq (1 - \sigma)\Delta$$

entonces

$$-\Psi(p) \geq \frac{1}{2}(1 - \sigma)^2(\|Rp\|^2 + \lambda\Delta^2) \geq (1 - \sigma)^2|\Psi^*|$$

Demostración.

Como en la demostración del Lema anterior tenemos que (4.9) se satisface para cualquier $z \in \mathbb{R}^n$ y entonces

$$\Psi^* = \Psi(p + z^*) = -\frac{1}{2}(\|Rp\|^2 + \lambda\|p + z^*\|^2) + \frac{1}{2}\|Rz^*\|^2$$

de donde

$$\Psi^* \geq -\frac{1}{2}(\|Rp\|^2 + \lambda\|p + z^*\|^2)$$

y esto implica

$$-\Psi^* \leq \frac{1}{2}(\|Rp\|^2 + \lambda\|p + z^*\|^2)$$

Y de (4.9) con $z = 0$ tenemos

$$\Psi(p) = -\frac{1}{2}(\|Rp\|^2 + \lambda\|p\|^2)$$

entonces

$$\begin{aligned} -\Psi(p) &= \frac{1}{2}(\|Rp\|^2 + \lambda\|p\|^2) \\ &\geq \frac{1}{2}(\|Rp\|^2 + \lambda(1-\sigma)^2\Delta^2) \\ &\geq \frac{1}{2}((1-\sigma)^2\|Rp\|^2 + \lambda(1-\sigma)^2\Delta^2) \\ &= \frac{1}{2}(1-\sigma)^2(\|Rp\|^2 + \lambda\Delta^2) \end{aligned}$$

de aquí se deduce que

$$\begin{aligned} -\Psi(p) &\geq \frac{1}{2}(1-\sigma)^2(\|Rp\|^2 + \lambda\Delta^2) \\ &\geq (1-\sigma)^2(-\Psi^*) \\ &= (1-\sigma)^2|\Psi^*| \end{aligned}$$

lo cual termina la demostración del lema. ■

Entonces si $p(\lambda)$ satisface (4.13) o (4.14) tendremos una solución aproximada de (4.1).

Ya tenemos la caracterización y propiedades de la solución veremos ahora algoritmos prácticos para encontrarla.

Empezaremos con el método de la *"pata de perro"*, el cual es apropiado cuando $B > 0$. Después veremos el método de Steihaug, apropiado cuando B es grande y sparse. También describiremos una estrategia para aproximar la solución exacta que busca un valor de λ modificando el problema a encontrar la raiz de una ecuación de una variable utilizando el método de Newton en una dimensión. Enseguida se enuncian los detalles al respecto.

Queremos resolver (4.1). Si $B > 0$ y $p = -B^{-1}g$ es tal que $\|p\| \leq \Delta$, entonces p es el punto solución, esto es, la solución es la misma que si minimizamos sin restricciones.

En caso de que $B > 0$, $p = -B^{-1}g$ con $\|p\| > \Delta$, entonces se tiene que el mínimo sin restricciones está fuera de la región de radio Δ, entonces la solución a (4.1) se encuentra sobre la frontera de la región, esto es, queremos resolver

$$\min_{\|w\|=\Delta} \Psi(w)$$

Si B es indefinida, entonces la cuadrática no está acotada inferiormente, no tiene mínimo y el punto que minimiza Ψ sobre la región, se encuentra, al igual que en el caso anterior, en la frontera de la región considerada.

Entonces los métodos que aproximan la solución se diseñan tal que el punto $p = -B^{-1}g$ con $\|p\| \leq \Delta$ se escoja cuando $B > 0$ y en los otros casos se busca el mínimo sobre la frontera de la región.

4.5 Método de la *"pata de perro"*

El método de la *"pata de perro"* es uno de los métodos más utilizados para resolver este problema, ya resulta muy económico, solo que tiene la desventaja de que solo puede ser aplicado cuando la matriz B es positiva definida.

Supongamos que $B > 0$ entonces el mínimo sin restricciones de Ψ es $p^B = -B^{-1}g$. Si este punto es factible para (4.1), entonces es la solución buscada, esto es,

$$p^* = p^B, \qquad \Delta \geq \|p^B\|$$

Si $\Delta < \|p^B\|$, entonces el mínimo sin restricciones está fuera de la región y el punto que buscamos se encuentra sobre al frontera.

En la figura (4.5) graficamos la trayectoria que sigue p conforme variamos el radio de la región de restricción, entonces el punto de la curva donde se minimice Ψ dentro de la región de radio Δ es nuestro punto solución. El método de la *"pata de perro"* consiste en reemplazar esta trayectoria curva para p con una trayectoria que consta de dos segmentos de línea. El primer segmento parte del origen al minimizador sobre la dirección de máximo descenso, esto es, sobre la dirección de $-g$, sujeto a la restricción de pertenecer a la región, este punto se denota por p^u. El segundo segmento es la línea que une p^u y p^B, entonces el punto sobre la trayectoria de los dos segmentos donde se alcance el *mínimo* de Ψ *dentro* de la región, esa es la solución aproximada de (4.1).

El punto p^u es llamado el punto de Cauchy.

Definición. El punto de Cauchy p^C es el mínimo de Ψ sobre la dirección de descenso más rápido, $-g$, sujeto a la restricción $\|p\| \leq \Delta$. Se tiene que $p^C = -\tau g$ con

$$\tau = \begin{cases} \Delta/\|g\| & \text{si } g^tBg \leq 0 \\ \min\{\|g\|^2/g^tBg, \Delta/\|g\|\} & \text{en otro caso} \end{cases}$$

La forma de τ se sigue de los siguiente: Cuando $g^tBg \leq 0$ entonces la cuadrática es *no* acotada y el *mínimo* sobre *cualquier dirección* se alcanza en la frontera de la región. Si $B > 0$ entonces aparecen dos casos si el mínimo sobre esta dirección está o no dentro de la región de restricción.

El punto de Cauchy jugará un papel muy importante en los métodos de región de confianza que se describirán en capítulos posteriores.

Como estamos suponiendo que $B > 0$ y p^B está fuera de la región, entonces

$$p^u = -\frac{g^t g}{g^t B g} g$$

Formalmente, la trayectoria de la *"pata de perro"* está dada por $\tilde{p}(\tau)$ con $\tau \in [0, 2]$ donde

$$\tilde{p}(\tau) = \begin{cases} \tau p^u & 0 \leq \tau \leq 1 \\ p^u + (\tau - 1)(p^B - p^u) & 1 \leq \tau \leq 2 \end{cases}$$

Geométricamente, es lo que tenemos en la figura 4.7

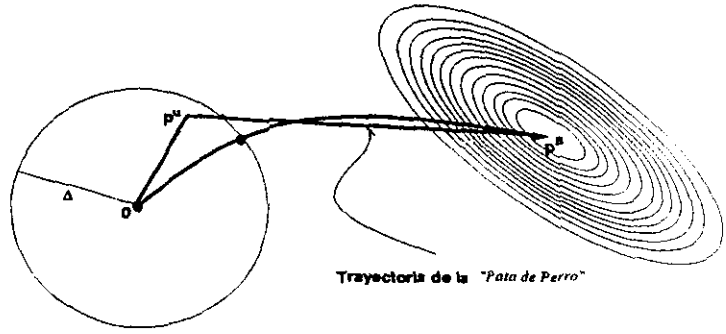


Figura 4.7: Trayectoria de la *"pata de perro"*

Entonces nuestro problema se traduce en minimizar Ψ a lo largo de $\tilde{p}(\tau)$ sujeto a la restricción $\|p\| \leq \Delta$. De hecho, no es necesario hacer una búsqueda, ya que como mostraremos, la trayectoria intersecta la frontera de la región en solo un punto y éste puede ser calculado analíticamente. Demostramos esto en el siguiente Lema.

Lema. *Sea $B > 0$, entonces*

i) $\|\tilde{p}(\tau)\|$ *es una función creciente de τ; y*

ii) $\Psi(\tilde{p}(\tau))$ *es una función decreciente de τ.*

Demostración.

Por la definición de p^u se ve que i) y ii) se satisfacen para $\tau \in [0, 1]$. Así que centraremos nuestra atención en el caso $\tau \in [1, 2]$.

Para i) definimos $h(\alpha) = \frac{1}{2}\|\tilde{p}(1+\alpha)\|^2$, i) quedará demostrado si se satisface que $h'(\alpha) \geq 0$ para $\alpha \in (0,1)$. Se tiene que

$$\begin{aligned} h(\alpha) &= \frac{1}{2}\|\tilde{p}(1+\alpha)\|^2 \\ &= \frac{1}{2}(\|p^u + \alpha(p^B - p^u)\|^2) \\ &= \frac{1}{2}\|p^u\|^2 + \alpha p^{u^t}(p^B - p^u) + \frac{1}{2}\alpha^2\|p^B - p^u\|^2 \end{aligned}$$

De aquí obtenemos

$$\begin{aligned} h'(\alpha) &= p^{u^t}(p^B - p^u) + \alpha\|p^B - p^u\|^2 \\ &= -p^{u^t}(p^u - p^B) + \alpha\|p^B - p^u\|^2 \\ &\geq -p^{u^t}(p^u - p^B) \\ &= \frac{g^t g}{g^t B g} g^t \left(-\frac{(g^t g)^2}{g^t B g} g + B^{-1} g \right) \\ &= g^t g \frac{g^t B^{-1} g}{g^t B g} \left(-\frac{(g^t g)^2}{(g^t B g)(g^t B^{-1} g)} + 1 \right) \\ &\geq 0 \end{aligned}$$

La última desigualdad se tiene ya que B y B^{-1} son positivas definidas y por la desiguladad de Cauchy–Schwarz se tiene que

$$\frac{\|g\|^4}{(g^t B g)(g^t B^{-1} g)} \leq 1$$

($|u^t v|^2 \leq \|u\|^2\|v\|^2$ sea $u = B^{1/2}g$, $v = B^{-1/2}g$, $B^{1/2}$ y $B^{-1/2}$ existen ya que $B > 0$.).

Por lo tanto se tiene que $h'(\alpha) \geq 0$, entonces $h(\alpha)$ es una función creciente y esto implica que $\|\tilde{p}(\tau)\|$ es una función creciente para $\tau \in [1,2]$.

Para ii) procedemos de manera similar, definimos $\hat{h}(\alpha) = \Psi(\tilde{p}(1+\alpha))$ y mostraremos que $\hat{h}'(\alpha) \leq 0$ con $\alpha \in (0,1)$. Tenemos que

$$\hat{h}(\alpha) = g^t(p^u + \alpha(p^B - p^u)) + \frac{1}{2}(p^u + \alpha(p^B - p^u))^t B(p^u + \alpha(p^B - p^u))$$

y de aquí se obtiene que

$$\hat{h}'(\alpha) = (p^B - p^u)^t(g + Bp^u) + \alpha(p^B - p^u)^t B(p^B - p^u)$$

$$\begin{aligned} &\leq (p^B - p^u)^t(g + Bp^u + B(p^B - p^u)) \\ &= (p^B - p^u)^t(g + Bp^B) \\ &= 0 \end{aligned}$$

Entonces $\hat{h}'(\alpha) \leq 0$ y por lo tanto $\Psi(\tilde{p}(\tau))$ es una función decreciente para $\tau \in [1, 2]$, que es lo que queríamos demostrar. Con esto termina la demostración del Lema. ■

Como una consecuencia del Lema anterior se tiene lo siguiente: los segmentos que definen la trayectoria de la *"pata de perro"* intersectan la frontera de la región $\|p\| = \Delta$ en exactamente un punto si $\|p^B\| > \Delta$ y en ningún punto en otro caso. Como Ψ es decreciente a lo largo de la trayectoria, el valor buscado de p es p^B si $\|p^B\| \leq \Delta$ y en otro caso p es el punto de intersección de la *"pata de perro"* con la frontera de la región. En este último caso encontramos el valor de τ resolviendo la ecuación cuadrática

$$\|p^u + (\tau - 1)(p^B - p^u)\|^2 = \Delta^2$$

y esta τ nos define el punto solución aproximado.

Esta estrategia de la *"pata de perro"* trabaja con $B > 0$, si B es indefinida ya no puede aplicarse directamente, ya que p^B no es el mínimo sin restricciones de Ψ. Enseguida describimos otro método que permite trabajar con B indefinida.

4.6 Método de Steihaug

El método anterior, requiere la solución de un sistema lineal que involucra B. Cuando B es grande, esta operación puede ser muy costosa, entonces buscamos una técnica que encuentre una solución aproximada de (4.1) que no requiera la solución exacta de un sistema lineal. Una técnica de este tipo fue propuesta por Steihaug y es la que describiremos a continuación. El método está basado en el algoritmo de gradiente conjugado.

La diferencia con el método de Gradiente Conjugado clásico es que Gradiente Conjugado encuentra solución a $Bp = -g$ y el algoritmo de Steihaug intenta resolver el sistema lineal pero el algoritmo termina si encontramos un p tal que $\|p\| \geq \Delta$ o si encuentra una dirección de curvatura negativa, en estos casos, el punto buscado se encuentra intersectando la dirección de búsqueda actual con la frontera de la región.

El algoritmo es el siguiente:

Algoritmo de Steihaug

dada $\epsilon > 0$, $p_0 = 0$, $r_0 = g$, $d_0 = -r_0$
if $\|r_0\| < \epsilon$, return, $p = p_0$, end
for $j = 0, 1, 2, \ldots$
  if $d_j^t B d_j \leq 0$ encontrar τ tal que $p = p_j + \tau d_j$ minimice
    $\Psi(p)$ en 4.1 y satisfaga $\|p\| = \Delta$.
    return p.
  end
  $\alpha_j = r_j^t r_j / d_j^t B d_j$
  $p_{j+1} = p_j + \alpha_j + d_j$
  if $\|p_{j+1}\| \geq \Delta$: encontrar $\tau > 0$ tal que $p = p_j + \tau d_j$ satisfaga $\|p\| = \Delta$.
    return p
  end
  $r_{j+1} = r_j + \alpha_j B d_j$
  if $\|r_{j+1}\| < \epsilon \|r_0\|$: return $p = p_{j+1}$, end
  $\beta_{j+1} = r_{j+1}^t r_{j+1} / r_j^t r_j$
  $d_{j+1} = r_{j+1} + \beta_{j+1} d_j$
end

La inicialización de $p_0 = 0$ es crucial, ya que después de la primera iteración si $\|r_0\| > \epsilon$ tenemos que

$$p_1 = \alpha_o d_0 = \frac{r_0^t r_0}{d_0^t B d_0} = -\frac{g^t g}{g^t B g} g$$

que es el punto de Cauchy. Esta propiedad será de gran utilidad cuando se vean los algoritmos de Región de Confianza.

Una propiedad importante del método es que cada iterando p_j es mayor en norma que su predecesor. Esta es una consecuencia de la inicialización $p_0 = 0$. La principal implicación es que es aceptable parar la iteración tan pronto como lleguemos a la frontera de la región ya que ningún iterando posterior estará dentro de la región y tendrá un menor valor de Ψ.

Esto se establece en el siguiente teorema.

Teorema *La sucesión de vectores generada por el método de Steihaug iniciando con* $p_0 = 0$ *satisface*

$$\|p_j\|_2 < \|p_{j+1}\|_2$$

Demostración.

Mostraremos primero que la sucesión de vectores generada satisface $p_j^t r_j =$

0 para $j \geq 0$ y $p_j^t d_j > 0$ para $j \geq 1$.

El algoritmo es recursivo y se tiene que

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

entonces

$$\begin{aligned} p_j &= p_0 + \sum_{i=1}^{j-1} \alpha_i d_i \\ &= \sum_{i=1}^{j-1} \alpha_i d_i \end{aligned}$$

ya que incializamos $p_0 = 0$. Multiplicando esta expresión por r_j y utilizando la propiedad de gradiente conjugado de que las direcciones de búsqueda son ortogonales a los residuales, tenemos

$$\begin{aligned} p_j^t r_j &= \sum_{i=1}^{j-1} \alpha_i d_i^t r_j \\ &= 0 \end{aligned}$$

Mostraremos ahora por inducción que $p_j^t d_j > 0$. Tenemos que

$$\begin{aligned} p_1^t d_1 &= (\alpha_0 d_0)^t (r_1 + \beta_1 d_0) \\ &= \alpha_0 d_0^t r_1 + \alpha_0 \beta_1 d_0^t d_0 \\ &= \alpha_0 \beta_1 d_0^t d_0 \end{aligned}$$

esta última igualdad resulta de aplicar nuevamente la propiedad de gradiente conjugado. Ahora, por la definición de α_i y β_i se tiene que $\alpha_i \geq 0$ y $\beta_i \geq 0$, entonces

$$p_1^t d_1 = \alpha_0 \beta_1 d_0^t d_0 > 0$$

Por lo tanto sí se cumple para $j = 1$, suponemos ahora la hipótesis de inducción, que $p_j^t d_j > 0$ y mostraremos que se satisface para $j + 1$.

Tenemos que $p_{j+1}^t r_{j+1} = 0$, entonces

$$\begin{aligned} p_{j+1}^t d_{j+1} &= p_{j+1}^t (r_{j+1} + \beta_{j+1} d_j) \\ &= p_{j+1}^t r_{j+1} + \beta_{j+1} p_{j+1}^t d_j \end{aligned}$$

$$\begin{aligned} &= \beta_{j+1} p_{j+1}^t d_j \\ &= \beta_{j+1}(p_j + \alpha_j d_j)^t d_j \\ &= \beta_{j+1} p_j^t d_j + \beta_{j+1}\alpha_j d_j^t d_j \\ &> 0 \end{aligned}$$

La última expresión es positiva utilizando la hipótesis de inducción. Entonces tenemos ya demostrado que $p_j^t d_j > 0$ para $j \geq 1$. Demostraremos ahora el teorema.

Si el algoritmo para debido a que $d_j^t B d_j \leq 0$ o $\|p_{j+1}\|_2 \geq \Delta$, entonces el punto final p se escoge tal que $\|p\|_2 = \Delta$ que es la longitud más grande que puede tener cualquier punto. Para cubrir todas las otras posibilidades en el algoritmo debemos mostrar que $\|p_j\|_2 < \|p_{j+1}\|_2$ cuando $p_{j+1} = p_j + \alpha_j d_j$ y $j \geq 1$.

Veamos que

$$\begin{aligned} \|p_{j+1}\|_2^2 &= (p_j + \alpha_j d_j)^t (p_j + \alpha_j d_j) \\ &= \|p_j\|_2^2 + 2\alpha_j p_j^t d_j + \alpha^2 \|d_j\|_2^2 \end{aligned}$$

como $p_j^t d_j > 0$ se tiene que

$$\|p_j\|_2 < \|p_{j+1}\|_2$$

lo cual completa la demostración. ■

Este teorema nos dice que el algoritmo de Steihaug genera una trayectoria desde 0 hasta p^B en donde cada paso aumenta la distancia total al punto inicial. Cuando $B > 0$ esta trayectoria puede compararse con la *"pata de perro"* ya que ambos métodos se mueven de 0 a p^B hasta llegar a la frontera de la región.

4.7 Método para el cálculo de la solución exacta

Los métodos anteriores son bastante económicos pero por ejemplo el de la *"pata de perro"* no se pueden aplicar en general, requiere que $B > 0$. Veremos ahora una técnica para calcular una solución aproximada usando el método de Newton en una dimensión.

Tenemos entonces que de acuerdo a la caracterización, estamos buscando $\lambda \in \mathbb{R}$ tal que

$$\|p(\lambda)\| = \Delta$$

donde

$$p(\lambda) = -\sum_{j=1}^{n} \frac{q_j^t g}{\lambda_j + \lambda} q_j$$

con $B = Q\Lambda Q^t$ y $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$, $\lambda_1 \leq \cdots\cdots \lambda_n$, eigenvalores de B.

Entonces podemos transformar el problema a encontar una raiz de una función unidimensional.

Podemos utilizar el método de Newton para encontrar ceros de funciones unidimensionales y asi encontraremos el valor de $\lambda > \lambda_1$ que resuelve

$$\Phi_1(\lambda) = \|p(\lambda)\| - \Delta = 0$$

Esta ecuación tiene una desventaja, si consideramos λ ligeramente mayor que $-\lambda_1$, entonces

$$\begin{aligned} \Phi_1(\lambda) &= \|p(\lambda)\| - \Delta \\ &= \left\| \sum_{j=1}^{n} \frac{q_j^t g}{\lambda + \lambda_j} q_j \right\| - \Delta \\ &\approx \frac{C_1}{\lambda_1 + \lambda} + C_2 \end{aligned}$$

con $C_1 > 0$ y C_2 constantes. Para estos valores de λ la función es muy no lineal y el método de Newton en estos casos no es muy confiable o el proceso puede ser muy lento.

Lo que se hace entonces el reformular el problema de tal forma que la función sea cercana a ser lineal cerca de λ óptima. Definimos

$$\Phi_2(\lambda) = \frac{1}{\Delta} - \frac{1}{\|p(\lambda)\|}$$

para esta nueva función si λ es ligeramente mayor que $-\lambda_1$, tenemos

$$\begin{aligned} \Phi_2(\lambda) &= \frac{1}{\Delta} - \frac{1}{\|p(\lambda)\|} \\ &= \frac{1}{\Delta} - \frac{1}{\left\| \sum_{j=1}^{n} \frac{q_j^t g}{\lambda + \lambda_j} q_j \right\|} \\ &\approx \frac{1}{\Delta} + \frac{\lambda_1 + \lambda}{C_3} \end{aligned}$$

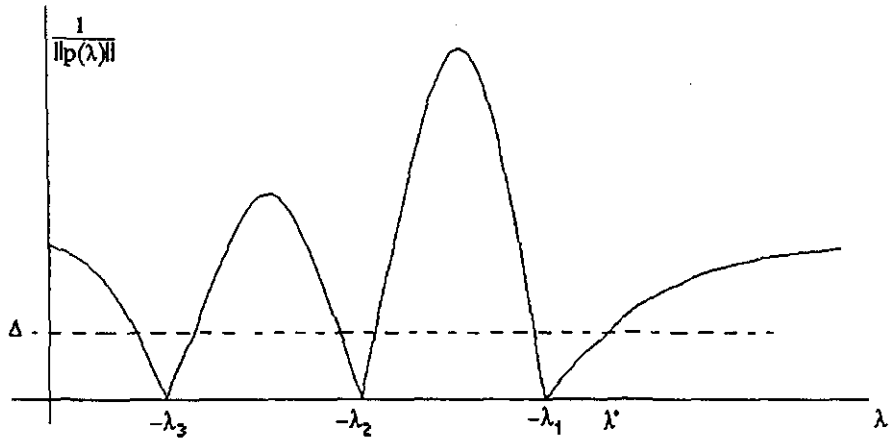


Figura 4.8: Gráfica de $\|p(\lambda)\|^{-1}$

Con $C_3 > 0$. Entonces $\Phi_2(\lambda)$ es casi lineal en el rango que estamos considerando, y el método de Newton en este caso funciona bien siempre que $\lambda > -\lambda_1$. La gráfica de $\Phi_2(\lambda)$ es la figura 4.8.

El método de Newton aplicado a $\Phi_2(\lambda)$ genera una sucesión de iteraciones $\lambda^{(l)}$ tales que

$$\lambda^{(l+1)} = \lambda^{(l)} - \frac{\Phi_2(\lambda^{(l)})}{\Phi_2'(\lambda^{(l)})}$$

Pasemos ahora a la implementación. Necesitamos $\Phi_2'(\lambda)$:

$$\begin{aligned}\Phi_2'(\lambda) &= \frac{d}{d\lambda}\left(\frac{1}{\Delta} - \frac{1}{\|p(\lambda)\|}\right) \\ &= \frac{d}{d\lambda}\left(-\frac{1}{\|p(\lambda)\|}\right)\end{aligned}$$

Y tenemos que

$$\begin{aligned}\frac{d}{d\lambda}\left(\frac{1}{\|p(\lambda)\|}\right) &= \frac{d}{d\lambda}(\|p(\lambda)\|^2)^{-1/2} \\ &= -\frac{1}{2}(\|p(\lambda)\|^2)^{-3/2}\frac{d}{d\lambda}(\|p(\lambda)\|^2) \\ &= -\frac{1}{2}\|p(\lambda)\|^{-3}\frac{d}{d\lambda}\left(\sum_{j=1}^{n}\frac{(q_j^t g)^2}{(\lambda_j+\lambda)^2}\right)\end{aligned}$$

$$= -\frac{1}{2}\|p(\lambda)\|^{-3}\left(-2\sum_{j=1}^{n}\frac{(q_j^t g)^2}{(\lambda_j+\lambda)^3}\right)$$
$$= \|p(\lambda)\|^{-3}\left(\sum_{j=1}^{n}\frac{(Q^t g)^2}{(\lambda_j+\lambda)^3}\right)$$

Ahora, se tiene que como $p = -Q(\Lambda + \lambda I)^{-1}Q^t g$ entonces

$$\sum_{j=1}^{n}\frac{(Q^t g)^2}{(\lambda_j+\lambda)^3} = p^t(B+\lambda I)^{-1}p$$

Si tenemos $B + \lambda I = R^t R$ entonces $(B+\lambda I)^{-1} = R^{-1}R^{-t}$, entonces

$$p^t(B+\lambda I)^{-1}p = p^t R^{-1}R^{-t}p = \|R^{-t}p\|^2$$

Sea v tal que $R^t v = p$ entonces

$$\|v\|^2 = \|R^{-t}p\|^2 = \sum_{j=1}^{n}\frac{(Q^t g)^2}{(\lambda_j+\lambda)^3}$$

de donde

$$\Phi_2'(\lambda) = -\|p(\lambda)\|^{-3}\|v\|^2$$

y esto implica que

$$\frac{\Phi_2(\lambda)}{\Phi_2'(\lambda)} = \frac{\frac{1}{\Delta}-\frac{1}{\|p(\lambda)\|}}{-\|p(\lambda)\|^{-3}\|v\|^2}$$
$$= -\left(\frac{\|p(\lambda)\|}{\|v\|}\right)^2\left(\frac{\|p(\lambda)\|-\Delta}{\Delta}\right)$$

tenemos entonces todos los elementos para presentar el algoritmo de Newton para el cálculo de λ.

4.7.1 Caso no duro

En caso que no se presente en niguna iteración un caso duro, podemos presentar un algoritmo de actualización de λ por el método de Newton como sigue:

Algoritmo (4.16)

Dados $\lambda, \Delta > 0$
for $l = 0, 1, 2, \ldots$

1. Factorizar $B + \lambda^{(l)} I = R^t R$
2. Resolver $Rp^{(l)} = -g$
3. Resolver $R^t v^{(l)} = p^{(l)}$
4. $\lambda^{(l+1)} = \lambda^{(l)} + \left(\frac{\|p(\lambda)\|}{\|v\|}\right)^2 \left(\frac{\|p(\lambda)\| - \Delta}{\Delta}\right)$

end

4.7.2 Caso duro

El algoritmo anterior no funciona en todos los casos ya que no sabemos en que momento se puede presentar un caso duro, presentamos ahora un primer algoritmo para el caso general, en donde por el Lema de la sección 4.3.1 la solución buscada es de la forma $p + z$, si no se está en un caso duro $z = 0$, en caso contrario se busca un z que satisfaga (4.8), podemos presentar el algoritmo como sigue:

Algoritmo (4.17)

Dados $\lambda, \Delta > 0$
for $l = 0, 1, 2, \ldots$

1. Factorizar $B + \lambda^{(l)} I = R^t R$
2. Resolver $Rp^{(l)} = -g$
3. Resolver $R^t v^{(l)} = p^{(l)}$
4. Si $\|p\| < \Delta$ calcular z
5. $\lambda^{(l+1)} = \lambda^{(l)} + \left(\frac{\|p(\lambda)\|}{\|v\|}\right)^2 \left(\frac{\|p(\lambda)\| - \Delta}{\Delta}\right)$

end

A este algoritmo se le deben añadir ciertas medidas de seguridad, por ejemplo, cuando $\lambda^{(l)} < -\lambda_1$ la factorización de Cholesky $B + \lambda^{(l)} I = R^t R$ no existe, por otro lado no hemos mencionado cómo hacer el cálculo de z, nos enfocaremos ahora a estos aspectos.

4.7.3 Cálculo de z

De acuerdo al Lema de la sección 4.3.1 la solución aproximada es de la forma

$$p + z$$

donde

$$B + \lambda I = R^t R, \qquad (B + \lambda I)p = -g, \qquad \lambda \geq 0$$

y z es tal que

$$\|p + z\| = \Delta, \qquad \|Rz\|^2 \leq \sigma(\|Rp\|^2 + \lambda\Delta^2)$$

El principal uso de este resultado es en el caso duro, donde debemos encontrar un $z = \tau\hat{z}$ que satisfaga la ecuación anterior considerando τ tal que $\|p + \tau\hat{z}\| = \Delta$ y escogiendo un $\hat{z}$ con $\|\hat{z}\| = 1$ tal que $\|R\hat{z}\|$ sea tan pequeña como sea posible.

Dado $\hat{z}$ con $\|\hat{z}\| = 1$ y p con $\|p\| < \Delta$ hay usualmente dos elecciones de τ que satisfacen $\|p + \tau\hat{z}\| = \Delta$ y de (4.9) debemos escoger la de menor magnitud que es la que minimiza $\Psi(p + \tau z)$. La τ buscada es

$$\tau = \frac{\Delta^2 - \|p\|^2}{p^t\hat{z} + \text{sign}(p^t\hat{z})[(p^t\hat{z})^2 + (\Delta^2 - \|p\|^2)]^{1/2}}$$

Un vector $\hat{z}$ con $\|\hat{z}\| = 1$ y tal que $\|R\hat{z}\|$ sea tan pequeña como sea posible puede obtenerse de varias formas, lo único que requiere el algoritmo es que $\|\hat{z}\|$ tienda a cero cuando λ tienda a $-\lambda_1$. Una forma económica de hacerlo es por medio de la iteración inversa, aunque existen otros caminos.

4.7.4 Protección de λ

Un elemento muy importante en la iteración es la protección necesaria para asegurar que se encuentre la solución buscada. Esta protección va a depender en este caso de la forma de Φ, recordemos que

$$\Phi(\lambda) = \|p(\lambda)\| - \Delta$$

con $(B + \lambda I)p(\lambda) = -g$, ésta es una función convexa y estrictamente decreciente en $(-\lambda_1, \infty)$, lo cual implica que el método de Newton iniciando en $\lambda \in (-\lambda_1, \infty)$ con $\Phi(0) > 0$ produce una sucesión monótonamente creciente que converge a la solución $\Phi(\lambda^*) = 0$. Además, si $\lambda \in (-\lambda_1, \infty)$ y $\Phi(\lambda) < 0$ la siguiente iteración de Newton, λ_+, es tal que o $\lambda_+ \leq -\lambda_1$ o $\Phi(\lambda_+) \geq 0$.

El esquema de protección utiliza parámetros λ_L, λ_U y λ_S tales que $[\lambda_L, \lambda_U]$ es un intervalo de incertidumbre que contiene a λ^* y λ_S es una cota inferior de $-\lambda_1$, el algoritmo de protección es el siguiente:

Protección de λ

a) $\lambda = \max(\lambda, \lambda_L)$

b) $\lambda = \min(\lambda, \lambda_U)$

c) Si $\lambda \leq \lambda_S$, entonces $\lambda = \max(.001\lambda_U, (\lambda_L\lambda_U)^{1/2})$

Esquemas de protección de este tipo son utilizados por Moré [31] para el caso en que B sea positiva semidefinida y por Gay para el caso general. Los primeros dos pasos del algoritmo aseguran que $\lambda \in [\lambda_L, \lambda_U]$ y el tercer paso garantiza la reducción de la longitud del intervalo de incertidumbre. El tercer paso el crucial: si la longitud del intervalo de incertidumbre permanece acotada lejos de cero entonces el tercer paso sólo puede ser ejecutado un número finito de veces. Esto se verá más claro cuando se vean las reglas de actualización de los parámetros de protección.

Dado λ_S y un λ de prueba las reglas para λ_S son las siguientes: Si $\lambda \in (-\lambda_1, \infty)$ y $\Phi(\lambda) < 0$, entonces $\|p(\lambda)\| < \Delta$ y calculamos τ y $\hat{z}$ por el algoritmo del caso duro y hacemos

$$\lambda_S = \max(\lambda_S, \lambda - \|R\hat{z}\|^2) \tag{4.18}$$

Como R es el factor de Cholesky de $(B+\lambda I)$ entonces para cualquier $\hat{z}$ tal que $\|\hat{z}\| = 1$ tenemos

$$\lambda - \|R\hat{z}\|^2 \leq -\lambda_1$$

Entonces si λ_S es una cota inferior en $-\lambda_1$, (4.18) garantiza que λ_S actualizado es también una cota para $-\lambda_1$. Si $\lambda \leq -\lambda_1$ entonces podemos actualizar λ_S notando que durante la descomposición de Cholesky de $B + \lambda I$ es posible encontrar un $\delta \geq 0$ tal que la submatriz principal de orden $l \leq n$ de

$$B + \lambda I + \delta e_l e_l^t$$

es singular. Además, es posible determinar $u \in \mathcal{R}^n$ tal que

$$(B + \lambda I + \delta e_l e_l^t)u = 0$$

con $u_l = 1$ y $u_i = 0$ para $i > l$. Entonces

$$\lambda_S = \max\left(\lambda_S, \lambda + \frac{\delta}{\|u\|^2}\right) \tag{4.19}$$

es una cota inferior de $-\lambda_1$.

Como $\|R\hat{z}\|^2$ es usualmente cercana a $\lambda + \lambda_1$, actualizar λ_S vía (4.18) tiende a evitar intentos de λ para los cuales $B + \lambda I$ no sea positiva definida y de aquí, reduce el número de iteraciones necesarias para la convergencia.

Las reglas para actualizar λ_L y λ_U son las siguientes:

Actualización de λ_L, λ_U y λ_S

1. Si $\lambda \in (-\lambda_1, \infty)$ y $\Phi(\lambda) < 0$ entonces
   $\lambda_U = \min(\lambda_U, \lambda)$
   si no
   $\lambda_L = \max(\lambda_L, \lambda)$
2. Actualizar λ_S usando (4.18) y (4.19)
3. Sea $\lambda_L = \max(\lambda_L, \lambda_S)$

Los valores iniciales para estos parámetros son

$$\lambda_S = \max\{-\beta_{ii}\}$$

con β_{ii} el i–ésimo elemento de la diagonal de B y

$$\lambda_L = \max\left(0, \lambda_S, \frac{\|g\|}{\Delta} - \|B\|_1\right)$$

$$\lambda_U = \frac{\|g\|}{\Delta} + \|B\|_1$$

Estas elecciones de λ_L y λ_U están basadas en la observación (4.3), la cual implica que

$$\frac{\|g\|}{|\lambda + \lambda_n|} \leq \|p(\lambda)\| \leq \frac{\|g\|}{|\lambda + \lambda_1|}, \qquad -\lambda_1 < \lambda$$

donde λ_1 y λ_n son, respectivamente, los eigenvalores más pequeño y mas grande de B. No son las únicas opciones para valores iniciales, pero estas son simples y efectivas para grandes valores de $\|g\|/\Delta$.

En resumen el siguiente algoritmo define una iteración típica para resolver nuestro problema

Algoritmo (4.20)

1. Proteger λ
2. Si $B + \lambda I$ es positiva definida factorizar $B + \lambda I = R^t R$, si no, ir a 5
3. Resolver $R^t R p = -g$
4. Si $\|p\| < \Delta$ calcular τ y $\hat{z}$
5. Actualizar λ_L, λ_U, λ_S
6. Verificar criterios de convergencia
7. Si $B + \lambda I$ es positiva definida y $g \neq 0$ actualizar λ vía paso 5 del algoritmo (4.17); si no, actualizar λ vía $\lambda = \lambda_S$.

El último paso del algoritmo se propone por lo siguiente: Si $B + \lambda I$ es positiva definida y $g \neq 0$ entonces el algoritmo de Newton en (4.17) tiende a una cota inferior de $-\lambda_1$ para $\|g\|$ suficientemente pequeña y entonces actualizar λ vía λ_S es una elección razonable cuando $g = 0$. También nótese que asignar λ_S a λ obliga a proteger λ en la siguiente iteración y ésta es la estrategia buscada cuando la iteración de Newton no puede operar.

Ahora veamos que el algoritmo termina en un número finito de pasos produciendo un $\lambda \in (-\lambda_1, \infty)$ con $\Phi(\lambda) \geq 0$ o un intervalo de incertidumbre arbitrariamente pequeño. Si suponemos que la longitud del intervalo de incertidumbre permanece acotado lejos de cero, entonces el tercer paso de la protección de λ garantiza que $\lambda \leq \lambda_S$ solo puede ocurrir un número finito de veces. Ahora, si $\lambda \leq -\lambda_1$ entonces $\lambda \leq \lambda_S$ para la siguiente iteración. Finalmente si $\lambda \in (-\lambda_1, \infty)$ y $\Phi(\lambda) < 0$ entonces recordemos que la siguiente iteración de Newton, λ_+, va a ser tal que o $\lambda_+ \leq -\lambda_1$ o $\Phi(\lambda_+) \geq 0$. Entonces los argumentos anteriores muestran que $\lambda_+ \leq -\lambda_1$ solo puede ocurrir un número finito de veces, entonces eventualmente $\lambda_+ \in (-\lambda_1, \infty)$ y $\Phi(\lambda_+) \geq 0$.

La importancia de lo anterior es porque dado $\lambda \in (-\lambda_1, \infty)$ con $\Phi(\lambda) \geq 0$ el algoritmo (4.20) eventualmente satisfacerá (4.13) mientras que si el intervalo de incertidumbre es pequeño entonces R es cercana a ser singular y es entonces posible que satisfaga (4.14), con esto el algoritmo terminará con una solución calculada s que satisface (4.15).

4.7.5 Experimentos Numéricos

Se realizó una implementación del algoritmo anterior en Matlab para probar su efectividad. Se realizaron experimentos numéricos y se compararon con los resultados obtenidos por Moré y Sorensen en [32] donde se describe una implementación del algoritmo en Fortran.

Para definir el problema, se genera la matriz $B = QDQ^t$ para alguna matriz ortogonal Q y D una matriz diagonal, se considera $g = Q\hat{g}$ para algún vector $\hat{g}$. Esto hace posible generar un caso duro haciendo cero la componente de $\hat{g}$ correspondiente al elemento mas pequeño de D. La estructura de B se obtiene escogiendo la matriz ortogonal Q como $Q = Q_1Q_2Q_3$, donde

$$Q_j = I - 2\frac{w_j w_j^t}{\|w_j\|^2}, \qquad j = 1, 2, 3$$

y las componentes de w_j números aleatorios distribuidos uniformemente en $(-1, 1)$. El problema estará determinado una vez que se especifiquen Δ, $\hat{g}$ y D.

Escogemos $\hat{g}$ y D con números aleatorios uniformemente distribuidos en $(-1, 1)$.

Para la elección del punto inicial consideramos

$$\lambda_0 = \frac{\|g\|}{\Delta} \tag{4.21}$$

si no se tiene mas información del problema (4.21) es una buena elección para iniciar la iteración.

Solo queda determinar Δ, un parámetro importante; si Δ se escoge en $(0, 1)$ (4.21) es usualmente un buen estimador inicial y el algoritmo terminará rápidamente. Un buen valor para Δ en este caso es considerarla dentro del intervalo $(0, 100)$ para que el algoritmo pueda operar, entonces se escoge Δ dentro de este intervalo también de forma aleatoria.

Los criterios de paro están determinados por $\sigma_1 = 0.1$ y $\sigma_2 = 0$ en (4.13) y (4.14). Se escoge $\sigma_2 = 0$ ya que σ_2 se necesita para manejar el caso $g = 0$ y B positiva semidefinida y singular, no estamos considerando B con esas características .

Presentamos ahora los resultados de los experimentos para problemas de dimensión 10, 20, 40, 60, 80 y 100. Para cada dimensión se generaron 5 problemas y se reportan el número promedio y el máximo de iteraciones necesitadas para la convergencia. Los resultados obtenidos se presentan en la siguiente tabla.

Tabla 1. Caso general

	Iteraciones	
Dimensión	Promedio	Máximo
10	2.2	3
20	3.2	5
40	3.4	4
60	2.6	3
80	4.2	5
100	3.8	5

Para probar el algoritmo en el caso duro se genera un problema de este tipo haciendo cero la componente de $\hat{g}$ correspondiente al elemento mas pequeño de D. Los resultados obtenidos en este caso se reportan en la siguiente tabla.

Tabla 2. Caso duro

	Iteraciones	
Dimensión	Promedio	Máximo
10	2.6	3
20	2.2	3
40	2.8	3
60	3.4	4
80	4	4
100	3.6	5

Los resultados son comparables con los reportados en [32] el número de iteraciones se reporta muy similar en ambos casos. Como se mencionó, el algoritmo acelera la convergencia para Δ mas pequeña, por ejemplo en el caso general si Δ se escoge en $(0, 1)$, el número máximo de iteraciones es 2, con lo cual el algoritmo converge rápidamente. De la tabla 2, podemos mencionar que el caso duro es tratado eficientemente y puede manejarse con el mismo esfuerzo computacional que el caso general.

Referencias

J. Dennis, R. Schnabel, (1983) *Numerical Methods for Unconstrained Optimization and nonlinear equations* Prentice–Hall.

J. Moré, D. Sorensen, (1983). Computing a Trust Region Step, *SIAM J. Sci. Stat. Comput.* vol. 4, no. 3, pp. 553–572.

J. Moré, D. Sorensen (1984). Newton's Method, *Studies in Numerical Analysis*, Ed. G.H. Golub, pp. 29–82.

T.Steihaug (1983). The conjugate gradient method and trust regions in large scale optimization, *SIAM J.Numer. Anal.*, vol. 20, pp.626–637.

Capítulo 5

Métodos de Región de Confianza

En el método de Búsqueda Lineal discutido en el capítulo 2, dada una dirección de búsqueda se enfocaba en encontrar un tamaño de paso adecuado que garantizara un decrecimiento en la función objetivo, esto se lograba haciendo uso de un modelo unidimensional de la función. Esta estrategia es exitosa, pero tiene la desventaja de no hacer uso de la información que puede obtenerse aproximando localmente a f con un modelo n–dimensional. El método de Región de Confianza que describiremos en este capítulo se basa en la aproximación local por medio de un modelo cuadrático n–dimensional, para determinar la dirección en la cual avanzar. Este método puede verse como que opera en sentido contrario a la búsqueda lineal, aquí primero se determina el tamaño de paso y en base a este tamaño de paso se encuentra la dirección sobre la que hay que avanzar. El algoritmo resultante es muy confiable y tiene propiedades de convergencia global. En este capítulo veremos las ideas principales de este método y sus propiedades básica de convergencia.

Si $f : \mathbb{R}^n \to \mathbb{R}$ es una función dos veces continuamente diferenciable, entonces localmente en una vecindad con centro en x_k y radio Δ_k se satisface

$$f(x_k + w) \approx f(x_k) + \Psi_k(w), \qquad \|w\| \leq \Delta_k$$

donde

$$\Psi_k(w) = \nabla f(x_k)^t w + \frac{1}{2} w^t B_k w$$

es un modelo cuadrático alrededor de x_k, con B_k una aproximación a la matriz Hessiana en el punto x_k.

Si para cada x_k podemos determinar Δ_k, el radio de la región donde confiamos que el modelo cuadrático aproxima a la función f, entonces se sugiere calcular el paso

s_k tal que resuelva el problema

$$s_k = \min_{\|w\| \leq \Delta_k} \Psi_k(w)$$

de tal forma que

$$x_{k+1} = x_k + s_k$$

sea nuestro siguiente elemento en la iteración.

Entonces en el método de Región de Confianza primero localmente se hace una aproximación cuadrática n–dimensional de la función objetivo, se define una región alrededor de la iteración actual en donde se confía que el modelo representa adecuadamente a la función, esto es, una región de confianza, y se escoge la dirección a seguir como el mínimo del modelo sobre la región de confianza. Al hacerlo estamos escogiendo la dirección y el tamaño de paso simultáneamente.

Si el paso no es aceptable, es decir, que $x_k + s_k$ no produzca una reducción suficiente en f, entonces se reduce el tamaño de la región de confianza y se encuentra un nuevo s_k. En general, al cambiar el tamaño de la región, Δ_k, cambia también la dirección, s_k.

El tamaño de la región de confianza, Δ_k, es crucial para el buen funcionamiento del método. Si la región es muy pequeña puede ser que el modelo sea válido en una región mas grande y que se puedan tomar pasos mas grandes que nos acerquen más rápido a la solución buscada. En cambio, si la región es muy grande puede ser que el modelo sólo sea válido en una vecindad muy pequeña de x_k y que el paso encontrado no sea aceptable, en cuyo caso se tiene que reducir el tamaño de la región y calcular un nuevo s_k.

En algoritmos prácticos el tamaño de la región se escoge de acuerdo a la reducción obtenida en f en iteraciones anteriores. Si la reducción es muy significativa entonces el modelo es confiable y podemos aumentar el tamaño de la región, permitiendo asi pasos mas grandes, si el paso no es aceptable, el modelo representa inadecuadamente a la función en la región de confianza actual, entonces se debe reducir el tamaño de la región de confianza y generar s_k nuevamente.

Entonces en el método de Región de Confianza en cada iteración tiene que resolverse el problema

$$\min_{\|w\|_2 \leq \Delta} \Psi(w), \qquad \Psi(w) = \nabla f_k^t w + \frac{1}{2} w^t B_k w \tag{5.1}$$

Consideraremos la norma euclideana, de tal forma que necesitamos el mínimo de $\Psi(w)$ en la bola de radio Δ_k. La existencia de la solución de (5.1) así como algoritmos

prácticos para el cálculo de soluciones aproximadas y un algoritmo para aproximar la solución exacta, se enunciaron en el capítulo 4. Haremos aquí uso de estos métodos y sus propiedades para demostrar convergencia global mas adelante. Veremos ahora la descripción del algoritmo en el entendido de que sabemos cómo calcular una solución aproximada a (5.1).

5.1 Algoritmo

Para definir el algoritmo de Región de Confianza, debemos primero definir la estrategia para la elección de Δ_k, el radio de la región de confianza en cada iteración. Obteniendo un paso s_k, se calcula el cociente

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{\Psi(0) - \Psi(s_k)}$$

el numerador se conoce como la reducción real y el denominador como la reducción pronosticada.

Nótese que

$$-\Psi(s_k) = \Psi(0) - \Psi(s_k)$$

éste siempre será un valor no negativo ya que s_k es el mínimo sobre $\{s : \|s\| \leq \Delta\}$ y este conjunto contiene $s = 0$, entonces $\Psi(s_k) \leq \Psi(0)$.

Entonces si

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{-\Psi(s_k)} \tag{5.2}$$

es negativo, entonces $f(x_k + s_k) > f(x_k)$ y el paso debe ser rechazado, se debe disminuir la región de confianza y calcular nuevamente s_k.

Si ρ_k es positivo pero cercano a cero, quiere decir que la reducción en f no fue significativa, hay que volver a calcular s_k con un Δ_k mas pequeño.

En cambio, si ρ_k es cercano a uno, entonces el modelo aproxima muy bien a f en esta región, se acepta el paso y se aumenta la región de confianza.

El algoritmo que describe el proceso se enuncia enseguida.

Algoritmo: Región de Confianza
Sean $0 < \mu < \eta < 1$ y $0 < \gamma_1 < 1 < \gamma_2$ constantes dadas
Dados $x_0 \in \mathbb{R}^n$ y $\Delta_0 > 0$
for k=0,1,2,...
  Obtener s_k resolviendo (5.1)
  Evaluar ρ_k de (5.2)
  Si $\rho_k \leq \mu$ entonces $\Delta_{k+1} \in (0, \gamma_1 \Delta_k]$
  else
    $x_{k+1} = x_k + s_k$
    Si $\rho_k \leq \eta$ entonces $\Delta_{k+1} \in [\gamma_1 \Delta_k, \Delta_k]$
    else
      $\Delta_{k+1} \in [\Delta_k, \gamma_2 \Delta_k]$
    end
  end
end

Los valores usuales de las constantes son

$$\mu = \frac{1}{4}, \qquad \eta = \frac{3}{4}, \qquad \gamma_1 = \frac{1}{2}, \qquad \gamma_2 = 2$$

éstos varían de acuerdo a las necesidades del problema.

En la figura 5.1 ilustramos la actualización de Δ de acuerdo al valor de ρ_k.

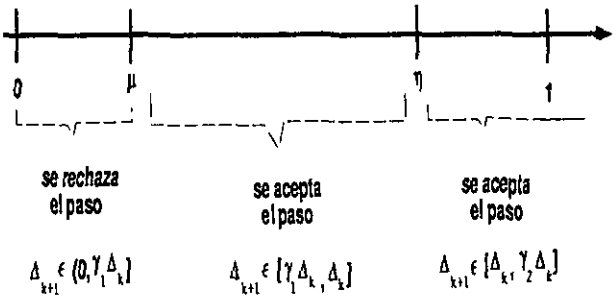


Figura 5.1: Actualización de Δ de acuerdo al valor de ρ_k

Esta es la forma básica del algoritmo de Región de Confianza.

Tenemos entonces cómo actualizar Δ_k el radio de la región de confianza, veremos ahora que condiciones debe satisfacer s_k en cada paso de manera que se logre una reducción significativa de la función y tal que el algoritmo genere una sucesión $\{x_k\}$ que converja a la solución buscada.

Al igual que en el problema de búsqueda lineal no estamos interesados en resolver el problema con gran exactitud, sino en encontrar condiciones flexibles para aceptar s_k que sean suficientes para garantizar convergencia de la sucesión generada por el algoritmo de Región de Confianza. s_k debe ser tal que esté dentro de la región de confianza y donde se obtenga una reducción significativa del modelo cuadrático.

La primera condición para aceptar el paso es que dada $\delta > 0$, s_k debe ser tal que

$$\|s_k\| \leq \delta \Delta_k$$

La reducción que se necesita en el modelo para garantizar convergencia es la que está dada por el estimador

$$-\Psi_k(s_k) \geq c_1 \|g_k\| \min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right) \tag{5.3}$$

para alguna constante $c_1 \in (0, 1]$. Notemos que cuando Δ_k es el valor mínimo en (5.3), la condición se parece a la primera condición de Wolfe–Powell: la reducción deseada en el modelo es proporcional al gradiente y el tamaño de paso.

Veremos ahora que tanto el algoritmo de la *"pata de perro"* como el de Steihaug producen soluciones aproximadas que satisfacen (5.3), para esto nos apoyaremos en la reducción alcanzada en el modelo por el punto de Cauchy. Recordemos la definición del punto de Cauchy.

Definición. El punto de Cauchy para el problema de región de confianza (5.1) denotado por p_k^C es el mínimo de Ψ_k a lo largo de la dirección de descenso más rápido, $-g_k$, sujeto a la restricción de la región de confianza $\|p\| \leq \Delta_k$. Se tiene que

$$p_k^C = -\tau g_k$$

con

$$\tau = \begin{cases} \Delta_k / \|g_k\| & \text{si } g_k^t B_k g_k \leq 0 \\ \min\{\|g_k\|^2/(g_k^t B_k g_k), \Delta_k/\|g_k\|\} & \text{en otro caso} \end{cases}$$

El punto de Cauchy, como veremos enseguida, nos ayudará a demostrar convergencia global del algortimo de región de confianza, éste es utilizado ya que es muy económico de encontrar, ya que no requiere factorizaciones matriciales. Tenemos que el punto de Cauchy satisface (5.3) como se enuncia en el siguiente resultado.

Lema. *El punto de Cauchy p_k^C satisface (5.3) con $c_1 = \frac{1}{2}$, esto es,*

$$-\Psi_k(p_k^C) \geq \frac{1}{2}\|g_k\| \min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right) \tag{5.4}$$

Demostración.

Consideraremos tres casos: $g_k^t B_k g_k \leq 0$ donde el punto de Cauchy está en la frontera de la región y el caso $g_k^t B_k g_k > 0$ donde hay dos opciones, p_k^C dentro o en la frontera de la región.

Si $g_k^t B_k g_k \leq 0$, entonces tenemos que $p_k^C = -\frac{\Delta_k}{\|g_k\|} g_k$,

$$\begin{aligned}
\Psi_k(p_k^C) &= g_k^t \left(-\frac{\Delta_k}{\|g_k\|} g_k\right) + \frac{1}{2}\left(-\frac{\Delta_k}{\|g_k\|} g_k\right)^t B_k \left(-\frac{\Delta_k}{\|g_k\|} g_k\right) \\
&= -\frac{\Delta_k g_k^t g_k}{\|g_k\|} + \frac{1}{2}\frac{\Delta_k^2}{\|g_k\|^2} g_k^t B_k g_k \\
&\leq -\Delta_k \|g_k\| \\
&\leq -\|g_k\| \min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right)
\end{aligned}$$

Entonces

$$\Psi_k(0) - \Psi_k(p_k^C) \geq \|g_k\| \min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right)$$

y se satisface el Lema en este caso.

Supongamos ahora que $g_k^t B_k g_k > 0$ y que el punto de Cauchy está dentro de la región, esto es, $p_k^C = -\frac{\|g_k\|^2}{g_k^t B_k g_k} g_k$ entonces

$$\begin{aligned}
\Psi_k(p_k^C) &= -g_k^t g_k \frac{\|g_k\|^2}{g_k^t B_k g_k} + \frac{1}{2}\left(\frac{\|g_k\|^4}{(g_k^t B_k g_k)^2}(g_k^t B_k g_k)\right) \\
&= -\frac{1}{2}\frac{\|g_k\|^4}{g_k^t B_k g_k} \\
&\leq -\frac{1}{2}\frac{\|g_k\|^4}{\|B_k\|\|g_k\|^2} \\
&= -\frac{1}{2}\frac{\|g_k\|^2}{\|B_k\|} \\
&\leq -\frac{1}{2}\|g_k\| \min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right)
\end{aligned}$$

entonces se tiene que

$$-\Psi_k(p_k^C) \geq \frac{1}{2}\|g_k\| \min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right)$$

y se satisface el Lema también en este caso.

Por último consideremos el caso $g_k^t B_k g_k > 0$ y $p_k^C = -\frac{\Delta_k}{\|g_k\|} g_k$, esto es

$$\frac{\|g_k\|^2}{g_k^t B_k g_k} \geq \frac{\Delta_k}{\|g_k\|} \qquad \Rightarrow g_k^t B_k g_k \leq \frac{\|g_k\|^3}{\Delta_k}$$

entonces

$$\begin{aligned}
\Psi_k(p_k^C) &= -\frac{\Delta_k}{\|g_k\|}\|g_k\|^2 + \frac{1}{2}\frac{\Delta_k^2}{\|g_k\|^2} g_k^t B_k g_k \\
&\leq -\frac{\Delta_k}{\|g_k\|}\|g_k\|^2 + \frac{1}{2}\frac{\Delta_k^2}{\|g_k\|^2}\frac{\|g_k\|^3}{\Delta_k} \\
&= -\Delta_k\|g_k\| + \frac{1}{2}\Delta_k\|g_k\| \\
&= -\frac{1}{2}\Delta_k\|g_k\| \\
&\leq -\frac{1}{2}\min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right)
\end{aligned}$$

Por lo tanto el Lema se satisface en todos los casos, y esto termina la demostración. ■

Para satisfacer (5.3) la solución aproximada s_k solo tiene que alcanzar una reducción de al menos una fracción fija c_2 de la reducción alcanzada por el punto de Cauchy. Establecemos esto formalmente en el siguiente teorema.

Teorema. *Sea s_k cualquier vector tal que $\|s_k\| \leq \Delta_k$ y*

$$-\Psi_k(s_k) \geq c_2(-\Psi_k(p_k^C))$$

Entonces s_k satisface (5.3) con $c_1 = \frac{1}{2}c_2$. En particular si s_k es la solución exacta s_k^* de (5.1) entonces satisface (5.3) con $c_2 = 1$.

Demostración.

Como $\|s_k\| \leq \Delta_k$, entonces

$$\begin{aligned}
-\Psi_k(s_k) &\geq c_2(-\Psi_k(p_k^C)) \\
&\geq c_2\left(\frac{1}{2}\|g_k\|\min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right)\right) \\
&\geq c_1\|g_k\|\min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right)
\end{aligned}$$

con $c_1 = \frac{1}{2}c_2$. ■

Note que los algoritmos de *"pata de perro"*, y de Steihaug satisfacen (5.3) con $c_1 = \frac{1}{2}c_2$, ya que todos producen soluciones aproximadas s_k para las cuales $\Psi_k(s_k) \leq \Psi_k(p_k^C)$.

5.2 Convergencia Global

En esta sección, probamos el resultado de convergencia global para al Algoritmo de Región de Confianza. Supondremos que las aproximaciones a las Hessianas B_k están uniformemente acotadas en norma.

Teorema. *Supongamos que $f : \mathbb{R}^n \to \mathbb{R}$ continuamente diferenciable y acotada inferiormente, que se genera $\{x_k\}$ con el algoritmo de región de confianza. Supongamos que $\|B_k\| \leq \beta$, con β una constante positiva y que todas las soluciones aproximadas de (5.1) satisfacen la desigualdad (5.3) para alguna $c_1 > 0$, entonces tenemos que*

$$\lim_{k\to\infty} \|g_k\| = 0$$

Demostración.

Procederemos por contradicción. Supongamos que existe un $\epsilon > 0$ tal que $\|g_k\| > \epsilon$ para toda k suficientemente grande.

Mostraremos que

$$\sum_{k=1}^{\infty} \Delta_k < \infty \tag{5.5}$$

Si existe un número finito de iteraciones donde se acepta el paso, esto es, se tiene que $\rho_k > \mu$ un número finito de veces, entonces tenemos que

$$\rho_k \leq \mu$$

para toda k suficientemente grande, pero por la forma de actualizar Δ_k se tiene que

$$\Delta_{k+1} \leq \gamma_1 \Delta_k$$

para toda k suficientemente grande, como $\gamma_1 < 1$, entonces se tiene que en este caso se satisface (5.5).

Si en cambio, existe una sucesión infinita de iteraciones $\{k_i\}$ donde se acepta el paso, entonces se tiene que $\rho_{k_i} > \mu$, esto es,

$$\frac{f(x_{k_i}) - f(x_{k_i} + s_{k_i})}{-\Psi(s_{k_i})} > \mu$$

Ahora, como se satisface (5.3), es decir,

$$-\Psi(s_{k_i}) \geq c_1 \|g_k\| \min\left(\Delta_{k_i}, \frac{\|g_{k_i}\|}{\|B_{k_i}\|}\right)$$

y como $\|B_k\| < \beta$, entonces se tiene que

$$f(x_{k_i}) - f(x_{k_i+1}) \geq \mu c_1 \|g_k\| \min\left(\Delta_{k_i}, \frac{\|g_{k_i}\|}{\beta}\right)$$

como f está acotada inferiormente se tiene que

$$\sum_{i=1}^{\infty} \Delta_{k_i} < \infty$$

Veremos ahora que

$$\sum_{k=1}^{\infty} \Delta_k \leq \sum_{i=1}^{\infty} \Delta_{k_i}$$

Consideremos que el paso k_i fue aceptado y que para $\Delta_{k_i+1}, \Delta_{k_i+2}, \ldots, \Delta_{k_i+r}$ el paso fue rechazado hasta llegar a Δ_{k_i+1} donde se aceptó el paso, de tal forma que

$$\Delta_{k_i+r+1} = \Delta_{k_i+1}$$

Como para Δ_{k_i+1} el paso se rechaza, se tiene que de acuerdo al algoritmo

$$\Delta_{k_i+1} \leq \gamma_1 \Delta_{k_i} < \Delta_{k_i}$$

ya que $0 < \gamma_1 < 1$, de la misma manera

$$\Delta_{k_i+2} \leq \gamma_1 \Delta_{k_i+1} \leq \gamma_1 \Delta_{k_i}$$

y en general, se tiene que

$$\Delta_{k_i+j} \leq \gamma_1^{j-1} \Delta_{k_i}$$

para $j \leq r$. Entonces se tiene que

$$\begin{aligned}\sum_{j=1}^{r} \Delta_{k_i+j} &\leq (1+\gamma_1+\gamma_1^2+\cdots+\gamma_1^{j-1})\Delta_{k_i} \\ &\leq \frac{1}{1-\gamma_1}\Delta_{k_i}\end{aligned}$$

lo cual implica que

$$\begin{aligned}\sum_{k=1}^{\infty} \Delta_k &= \sum_{i=1}^{\infty} \Delta_{k_i} + \sum_{k\neq k_i}^{\infty} \Delta_k \\ &\leq \sum_{i=1}^{\infty} \Delta_{k_i} + \sum_{i=1}^{\infty} \left(\frac{1}{1-\gamma_1}\right) \Delta_{k_i} \\ &= \left(1+\frac{1}{1-\gamma_1}\right) \sum_{i=1}^{\infty} \Delta_{k_i} \\ &< \infty\end{aligned}$$

por lo tanto (5.5) también se satisface en este caso.

Ahora mostraremos que (5.5) implica que $\{|\rho_k - 1|\}$ converge a cero. Como primer paso notemos que

$$\|x_{k+1} - x_k\| \leq \|s_k\| \leq \delta\Delta_k$$

y por (5.5) se tiene que $\{x_k\}$ converge.

Por otro lado,

$$|\rho_k - 1| = \left|\frac{\Psi_k(s_k) - (f(x_k+s_k) - f(x_k))}{-\Psi_k(s_k)}\right|$$

y se tiene que

$$\begin{aligned}|\Psi_k(s_k) - \nabla f(x_k)^t s_k| &\leq \left|\frac{1}{2} s_k^t B_k s_k\right| \\ &\leq \frac{1}{2}\|s_k\|^2\beta \\ &\leq \frac{1}{2}\delta^2\Delta_k^2\beta\end{aligned}$$

Usando esta desigualdad, tenemos que

$$\begin{aligned} |\Psi_k(s_k) - (f(x_k + s_k) - f(x_k))| &\leq \left|\Psi_k(s_k) - \nabla f(x_k)^t s_k - \frac{1}{2} s_k^t B_k s_k\right| \\ &\leq \frac{1}{2}\|s_k\|^2\beta + \frac{1}{2}\|s_k\|^2\beta \\ &= \|s_k\|^2\beta \\ &\leq \beta\delta\Delta_k^2 \end{aligned} \tag{5.6}$$

Ahora, por (5.3) y por la hipótesis de que $\|g_k\| > \epsilon$ se tiene que

$$\begin{aligned} -\Psi_k(s_k) &\geq c_1\|g_k\|\Delta_k \\ &\geq c_1\epsilon\Delta_k \end{aligned}$$

De esta desigualdad, de (5.6) y de (5.5) se concluye que

$$\{|\rho_k - 1|\} \to 0$$

Pero las reglas de actualización de Δ_k muestran que Δ_k no disminuye si $\rho_k \geq \eta$, entonces $\{\Delta_k\}$ no puede converger a cero. Esto contradice (5.5) y demuestra el teorema. ■

Tenemos entonces que siempre se alcanza convergencia global con el método de Región de Confianza. Veremos ahora como caso especial la convergencia global para el método de Newton con región de confianza, donde la solución de (5.1) se calcula con el algoritmo que aproxima la solución exacta.

5.3 Convergencia Global para métodos que aproximan la solución exacta

Con el algoritmo de Newton unidimensional para aproximar la solución exacta y sus correspondientes protecciones descritos en 4.7, tenemos que la solución aproximada s satisface las condiciones

$$-\Psi_k(s) \geq c_1(-\Psi_k(s^*)) \tag{5.7}$$

$$\|s\| \leq \gamma\Delta \tag{5.8}$$

donde s^* es la solución exacta, $c_1 \in (0, 1]$ es una constante y $\gamma > 0$. La condición (5.7) asegura que la solución aproximada alcanza una fracción significativa del máximo

decrecimiento posible en el modelo de la función Ψ_k. Una de las principales diferencias entre (5.7) y el criterio anterior (5.3) es que (5.7) hace mejor uso de la parte de segundo orden de Ψ, esto es, el término p^tBp. Esta diferencia se aprecia mejor en el caso en que $g = 0$ y B con eigenvalores negativos, lo cual indica que la iteración actual es un punto silla. Aquí el lado derecho de (5.3) es cero, y con este criterio el algoritmo termina en este punto, mientras que el lado derecho de (5.7) es positivo, lo cual indica que todavía es posible un decrecimiento en el modelo de la función, y obliga al algoritmo a moverse del punto x_k.

La atención que pone el algoritmo que aproxima la solución exacta al término del segundo orden solo está garantizada si el término refleja muy precisamente el comportamiento actual de la función f. La literatura ha considerado solo el caso $B = \nabla^2 f(x)$, esto es, el método de Newton con Región de confianza, y los resultados mas conocidos de convergencia global tratan con esta elección particular de B. El uso de la segunda derivada de la función en estos casos permite decir algo más de los puntos límite del algoritmo que el hecho de que son puntos estacionarios. De hecho, las condiciones de segundo orden se satisfacen en los puntos límite del algoritmo. Antes de establecer el resultado que demuestra esto, veremos algunos resultados sobre la aproximación a la solución exacta vista en el capítulo 4, para el caso especial en que $B_k = \nabla^2 f(x_k)$.

Tenemos que si $s_k \in \mathbb{R}^n$ es una solución al problema (5.1) entonces se tiene que existe un $\lambda_k \geq 0$ tal que

$$(\nabla^2 f(x_k) + \lambda_k I)s_k = -\nabla f(x_k), \quad \lambda_k(\Delta_k - \|s_k\|) = 0$$

Ahora, sea R^tR la factorización de Cholesky de $\nabla^2 f(x_k) + \lambda_k I$. Entonces tenemos que

$$R^tRs_k = -\nabla f(x_k)$$

de donde se tiene que

$$s_k^tR^tRs_k = -s_k^t\nabla f(x_k)$$

lo cual implica que

$$\begin{aligned} \|Rs_k\|^2 &= \nabla f(x_k)^t(\nabla^2 f(x_k) + \lambda_k I)^{-1}\nabla f(x_k) \\ &\geq \|\nabla f(x_k)\|^2\|\nabla^2 f(x_k) + \lambda_k I\|^{-1} \\ &\geq \frac{\|\nabla f(x_k)\|^2}{\|\nabla^2 f(x_k) + \lambda_k I\|} \\ &\geq \frac{\|\nabla f(x_k)\|^2}{\|\nabla^2 f(x_k)\| + \lambda_k} \end{aligned} \tag{5.9}$$

Por otro lado de (4.11) se tiene que

$$|\Psi(s^*)| = \frac{1}{2}(\|R_k s_k\|^2 + \lambda_k \Delta_k^2)$$

Entonces como se satisfacen (5.7) y (5.8) resulta que

$$-\Psi_k(s_k) \geq \frac{1}{2} c_1 (\|R_k s_k\|^2 + \lambda_k \Delta_k^2) \tag{5.10}$$

Y entonces las iteraciones $\{x_k\}$, generadas por el algoritmo de Región de Confianza, satisfacen

$$f(x_k) - f(x_{k+1}) > -\Psi_k(s_k)\mu$$

lo cual implica

$$f(x_k) - f(x_{k+1}) \geq \frac{1}{2}\mu c_1 (\|R_k s_k\|^2 + \lambda_k \Delta_k^2) \tag{5.11}$$

Estas últimas desigualdades son esenciales para probar nuestro siguiente resultado.

Teorema. *Sea $f : \mathbb{R}^n \to \mathbb{R}$ dos veces continuamente diferenciables en un conjunto abierto D, y suponga que el punto inicial x_0 es tal que el conjunto de nivel*

$$\Omega = \{x \in D : f(x) \leq f(x_0)\}$$

es compacto. Si la sucesión $\{x_k\}$ se genera con el algoritmo de región de confianza donde s_k satisaface (5.7) y (5.8) entonces o el algoritmo termina en un $x_k \in \Omega$ donde

$$\nabla f(x_k) = 0, \qquad \nabla^2 f(x_k) \text{ positiva semidefinida}$$

o $\{x_k\}$ tiene un punto límite $x^* \in \Omega$ donde $\nabla f(x^*) = 0$, $\nabla^2 f(x^*)$ es positiva semidefinida.

Demostración.

Si $\nabla f(x_k) = 0$ y $\nabla^2 f(x_k)$ es positiva semidefinida para algún iterando $x_k \in \Omega$, entonces el algoritmo termina, de otra forma (5.7) y (5.8) implican que $\Psi_k(s_k) < 0$ para $k \geq 0$ y entonces $\{x_k\}$ está bien definida y permanece en Ω.

Demostraremos el resultado bajo la hipótesis de que $\{\lambda_k\}$ tiende a cero. Si alguna subsucesión de $\{\lambda_k\}$ converge a cero, entonces como Ω es compacto, podemos suponer sin pérdida de generalidad que la misma subsucesión de

$\{x_k\}$ converge a algún x^* en el conjunto de nivel Ω. Como $\nabla^2 f(x_k)+\lambda_k I$ es positiva semidefinida, entonces $\nabla^2 f(x^*)$ también es positiva semidefinida.

El hecho de que $\nabla f(x^*) = 0$, se sigue de (5.9), tenemos que

$$\|R_k s_k\|^2 \geq \frac{\|\nabla f(x_k)\|^2}{\|\nabla^2 f(x_k)\| + \lambda_k}$$

y de (5.11) se tiene que $\{\|R_k p_k\|\}$ converge a cero.

El resultado se satisface si $\{\lambda_k\} \to 0$, mostraremos ahora que esto se cumple. Procederemos por contradicción. Si $\lambda_k \geq \epsilon > 0$, entonces, (5.8) y (5.10) implican que

$$-\Psi_k(s_k) \geq \frac{1}{2} c_1 \lambda_k \Delta_k^2 \geq \frac{1}{2}\left(\frac{c_1}{\gamma^2}\right)\epsilon \|s_k\|^2$$

Ahora, un estimador estándar es que

$$|f(x_k + s_k) - f(x_k) - \Psi_k(s_k)| \leq \frac{1}{2}\|s_k\|^2 \max_{0\leq\varepsilon\leq 1} \|\nabla^2 f(x_k + \varepsilon s_k) - \nabla^2 f(x_k)\|$$

y entonces las últimas dos desigualdades muestran que

$$|\rho_k - 1| \leq \left(\frac{\gamma^2}{c_1 \epsilon}\right) \max_{0\leq\varepsilon\leq 1} \|\nabla^2 f(x_k + \varepsilon s_k) - \nabla^2 f(x_k)\| \qquad (5.12)$$

La ecuación (5.11) implica que $\{\Delta_k\}$ converge a cero y entonces $\{\|s_k\|\}$ también converge a cero. Entonces la continuidad uniforme de $\nabla^2 f$ sobre Ω junto con (5.12) implican que $\rho_k > \eta$ para toda k suficientemente grande y entonces las reglas de actualización para Δ_k implican que $\{\Delta_k\}$ no converge a cero, lo cual es una contradicción.

Con esto queda demostrado el teorema. ■

Referencias

J. Dennis, R. Schnabel, (1983) *Numerical Methods for Unconstrained Optimization and nonlinear equations* Prentice-Hall.

R. Fletcher (1987). *Practical Methods of Optimization* John Wiley & Sons, 2nd.Edition.

J. Nocedal and J. Wright (1999). *Numerical Optimization*, Springer Series in Operations Research.

D.Sorensen, (1982). Newton's Method with a Model Trust Region Modification, *SIAM Numer. Anal.* vol. 19, no. 2, pp. 409–426.

Capítulo 6

Métodos Quasi–Newton

Muchos de los métodos de optimización están basados en modelos cuadráticos locales de la función objetivo, la forma mas conocida de este tipo de métodos es el método de Newton. El modelo cuadrático se obtiene de truncar la expasión en serie de Taylor de $f(x)$ alrededor del punto x_k, esto es,

$$f(x_k + s) \approx Q_k(s) = f(x_k) + s^t \nabla f(x_k) + \frac{1}{2} s^t \nabla^2 f(x_k) s$$

donde $s = x - x_k$ y $Q_k(s)$ es la aproximación cuadrática en la iteración k. Entonces la siguiente iteración en el método de Newton está dada por

$$x_{k+1} = x_k + s_k$$

donde la corrección s_k minimiza $Q_k(s)$. El método requiere disponer de la primera y la segunda derivada de la función en cualquier punto, de manera que $Q_k(s)$ esté bien definida, también necesita que $\nabla^2 f(x_k)$ sea positiva definida para que la cuadrática $Q_k(s)$ tenga un único mínimo.

Si todas las condiciones se cumplen, la k–ésima iteración del método de Newton está dada por

i) Resolver $\nabla^2 f(x_k) s_k = -\nabla f(x_k)$

ii) $x_{k+1} = x_k + s_k$

El orden de convergencia local para el método de Newton es cuadrático, es por esto que es de los métodos mas utilizados en caso de disponer de segundas derivadas de la función. El método admite estrategias para alcanzar convergencia global como búsqueda lineal o región de confianza.

La principal desventaja del método de Newton es el conocimiento de la segunda derivada de la función en cualquier punto, en muchas aplicaciones lo mas que puede tenerse es la disposición de la primera derivada. Es por esto que se generaron métodos relacionados con el método de Newton donde solo se hace uso de la información de la primera derivada de la función. La primera idea es aproximar la matriz hessiana de f por medio de diferencias de los vectores gradientes, pero esto tiene el inconveniente de que la matriz resultante puede no ser positiva definida, además se requieren n evaluaciones del gradiente para cada iteración, el proceso puede resultar muy costoso.

Para superar estas desventajas Davidon en 1958, introdujo una nueva clase de métodos conocidos como Quasi–Newton, éstos son métodos tipo Newton en donde la inversa de la matriz hessiana es aproximada por una matriz simétrica positiva definida, la cual es corregida o actualizada en cada iteración. La estructura básica de estos métodos en la iteración k, partiendo que se tiene H_k aproximación a la inversa de la matriz hessiana en x_k, es la siguiente:

i) $s_k = -H_k \nabla f(x_k)$

ii) $x_{k+1} = x_k + s_k$

iii) Actualizar H_k para obtener H_{k+1}.

La matriz inicial H_0 puede ser cualquier matriz positiva definida, si no se tiene información del problema puede escogerse $H_0 = I$. Este método presenta algunas ventajas respecto al método de Newton:

a) Solo requiere información de la primera derivada de la función.

b) H_k positiva definida implica que la dirección encontrada es siempre una dirección de descenso.

c) Reduce el número de flops por iteración.

La parte c) puede lograrse aproximando la inversa de la matriz hessiana en cada iteración en lugar de aproximar directamente la matriz hessiana, con lo que no se requiere resolver el sistema lineal en cada iteración. Debido a estas propiedades es que los métodos Quasi–Newton son bastante utilizados en la práctica. Su desempeño es muy satisfactorio, el orden de convergencia ya no es cuadrático como en el caso de Newton, se reduce a superlineal.

Este tipo de métodos también son conocidos como métodos de métrica variable o métodos tipo secante, nos referiremos a ellos como métodos Quasi–Newton. En este capítulo veremos en qué consisten estos métodos y las ideas detrás de ellos.

6.1 Idea de Davidon

Iniciaremos con la primera idea generada en esta dirección, la cual fue introducida por Davidon en 1958. Davidon trabajó el problema cuadrático

$$\min_x \frac{1}{2}x^tAx + b^tx + \gamma, \quad A \text{ positiva definida, } A = A^t$$

Sabemos que que la solución viene dada por $x^* = -A^{-1}b$. Davidon trabajó con direcciones conjugadas, A–ortogonales. Un conjunto $\{d_1, d_2, \ldots, d_n\}$ se dice A–ortogonal si

$$d_i^tAd_j = 0, \quad d_i^tAd_i \neq 0$$

Una de las propiedades importantes de las direcciones conjugadas es que es fácil calcular A^{-1}, ésta está dada por

$$A^{-1} = \sum_{i=1}^{n} \frac{d_id_i^t}{d_i^tAd_i}$$

La demostración es muy sencilla, tenemos que

$$A^{-1}Ad_k = \frac{d_kd_k^tAd_k}{d_k^tAd_k} = d_k, \quad k = 1, \ldots, n$$

La idea de Davidon es que dada una x_0 inicial, una B_0 arbitraria que sea una aproximación inicial a A^{-1} y el gradiente de la función en x_0, $\nabla f(x_0)$, encontrar

$$x_1 = x_0 + \alpha_0 d_0$$

donde $d_0 = -B_0\nabla f(x_0)$ es una dirección de descenso y con α_0 que resuelve el problema de búsqueda lineal en este punto, esto es,

$$\alpha_0 = \arg \min_\alpha f(x_0 + \alpha d_0)$$

Como f es una cuadrática, tenemos que la búsqueda lineal puede determinar α_0 de forma exacta

$$\alpha_0 = -\frac{d_0^t\nabla f(x_0)}{d_0^tAd_0}$$

Con estos elementos se quiere construir $B_1 > 0$ tal que aproxime mejor A^{-1}.

Una forma es que B_1 satisfaga

$$(AB_1)d_0 = d_0$$

esto considerando que $A^{-1} = \sum_{i=1}^{n} \frac{d_i d_i^t}{d_i^t A d_i}$, queremos que B_1 sea una corrección de B_0 y teniendo en cuenta el primer sumando en la expresión de A^{-1}, queremos que

$$B_1 = B_0 + \frac{d_0 d_0^t}{d_0^t A d_0} + R_0 > 0$$

de donde

$$B_1 A d_0 = B_0 A d_0 + \frac{d_0 d_0^t A d_0}{d_0^t A d_0} + R_0 A d_0 = d_0$$

entonces

$$\begin{aligned} B_0 A d_0 + R_0 A d_0 &= 0 \\ (B_0 + R_0) A d_0 &= 0 \end{aligned}$$

lo que implica

$$R_0 A d_0 = -B_0 A d_0$$

Entonces lo que conocemos sobre R_0 es que

$$R_0 u_0 = v_0, \quad R_0 = R_0^t$$

con $u_0 = A d_0$ y $v_0 = -B_0 A d_0$. La forma mas simple para R_0 es

$$R_0 = \theta w_0 w_0^t$$

en este caso tenemos

$$\theta w_0 (w_0^t u_0) = v_0 \quad \Rightarrow \quad w_0 = \lambda v_0$$

de donde

$$\theta \lambda v_0 (\lambda v_0^t u_0) = v_0$$

lo que implica

$$\theta \lambda^2 (v_0^t u_0) v_0 = v_0 \quad \Rightarrow \quad \theta \lambda^2 = \frac{1}{v_0^t u_0}$$

Entonces

$$\begin{aligned} R_0 &= \theta w_0 w_0^t \\ &= \theta (\lambda v_0)(\lambda v_0^t) \\ &= \theta \lambda^2 v_0 v_0^t \\ &= \frac{v_0 v_0^t}{v_0^t u_0} \end{aligned}$$

Que aplicado a nuestro caso, tenemos

$$R_0 = \frac{(-B_0 A d_0)(-B_0 A d_0)^t}{(-B_0 A d_0)^t (A d_0)}$$

de donde

$$R_0 = -\frac{B_0 A d_0 d_0^t A B_0}{d_0^t A B_0 A d_0} \tag{6.1}$$

Entonces tenemos que $x_1 = x_0 + \alpha_0 d_0$, de donde

$$s_0 = x_1 - x_0 = \alpha_0 d_0, \quad \Rightarrow \quad d_0 = \frac{s_0}{\alpha_0}$$

Por otro lado,

$$A(x_1 - x_0) = A(\alpha_0 d_0)$$

Entonces $\nabla f(x_1) - \nabla f(x_0) = \alpha_0 A d_0$. Sea $y_0 = \nabla f(x_1) - \nabla f(x_0)$, entonces

$$A d_0 = \frac{y_0}{\alpha_0}$$

Utilizando esto en (6.1), resulta

$$\begin{aligned} R_0 &= -\frac{B_0 \left(\frac{y_0}{\alpha_0}\right) \left(\frac{y_0}{\alpha_0}\right)^t B_0}{\left(\frac{y_0}{\alpha_0}\right)^t B_0 \left(\frac{y_0}{\alpha_0}\right)} \\ &= -\frac{B_0 y_0 y_0^t B_0}{y_0^t b_0 y_0} \end{aligned}$$

De donde

$$B_1 = B_0 + \frac{d_0 d_0^t}{d_0^t A d_0} - \frac{B_0 y_0 y_0^t B_0}{y_0^t B_0 y_0}$$

Pero no tenemos A explícitamente, pero si tenemos (Ad_0), entonces

$$\frac{d_0 d_0^t}{d_0^t A d_0} = \frac{\left(\frac{s_0}{\alpha_0}\right) \left(\frac{s_0}{\alpha_0}\right)^t}{\left(\frac{s_0}{\alpha_0}\right)^t \left(\frac{y_0}{\alpha_0}\right)} = \frac{s_0 s_0^t}{s_0^t y_0}$$

por lo tanto

$$B_1 = B_0 + \frac{s_0 s_0^t}{s_0^t y_0} - \frac{B_0 y_0 y_0^t B_0}{y_0^t B_0 y_0}$$

Esta es la fórmula de actualización de Davidon. Como B_0 es positiva definida, para que B_1 sea positiva definida, lo que necesito es que $s_0^t y_0 > 0$, como se establece en el siguiente teorema.

Teorema. *Si B es positiva definida y $s^t y > 0$, entonces la actualización de Davidon, B_1, es positiva definida.*

Demostración.

La actualización de Davidon viene dada por

$$B_1 = B - \frac{Byy^tB}{y^tBy} + \frac{ss^t}{s^ty}$$

Mostraremos que $z^t B_1 z > 0$ para toda $z \neq 0$. Usaremos el hecho que la suma de matrices positivas definidas es positiva definida. Como B es positiva definida, entonces podemos encontrar su factorización de Cholesky, y definimos

$$B = LL^t, \quad a = L^t z, \quad b = L^t y$$

entonces

$$\begin{aligned} z^t \left(B - \frac{Byy^tB}{y^tBy} \right) z &= z^t LL^t z - \frac{z^t LL^t yy^t LL^t z}{y^t LL^t y} \\ &= a^t a - \frac{(a^t b)(b^t a)}{b^t b} \\ &= a^t a - \frac{(a^t b)^2}{b^t b} \end{aligned}$$

Ahora, sabemos que $|a^t b| \leq \|a\|\|b\|$, entonces

$$(a^t b)^2 \leq (a^t a)(b^t b) \quad \Rightarrow \quad \frac{(a^t b)^2}{b^t b} \leq a^t a$$

entonces

$$a^t a - \frac{(a^t b)^2}{b^t b} \geq 0 \quad \Rightarrow \quad z^t \left(B - \frac{Byy^tB}{y^tBy} \right) z \geq 0$$

Como $z \neq 0$ la igualdad se satisface solo si a es proporcional a b, esto es z proporcional a y, pero como $s^t y > 0$, entonces

$$z^t \left(\frac{ss^t}{s^t y} \right) z \geq 0$$

con la igualdad estricta si z es proporcional a y. Por lo tanto

$$z^t B_1 z > 0$$

lo cual implica que B_1 es positiva definida. ■

Esta propiedad de preservar la positividad definida de las matrices será utilizada mas adelante.

En general, en la k–ésima iteración para el método de Davidon, tenemos que

$$B_{k+1} = B_k + \frac{s_k s_k^t}{s_k^t y_k} - \frac{B_k y_k y_k^t B_k}{y_k^t B_k y_k}$$

con $s_k = x_{k+1} - x_k$, $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.

Esta es la idea de Davidon que introdujo para funciones cuadráticas, que después se extendió para funciones en general como lo veremos mas adelante. Esta actualización, descubierta por Davidon, estudiada, implementada y popularizada por Fletcher y Powell en 1962, se conoce como la actualización DFP (Davidon–Fletcher–Powell), es además un miembro de la llamada familia de Broyden, que se estudia en las siguientes secciones.

Esta misma aproximación puede encontrarse por otro camino, pero antes de verlo con detalle, veremos el método de Broyden el cual es de la familia de los Quasi–Newton pero aplicado a sistemas de ecuaciones no lineales.

6.2 Método de Broyden

En 1965 Broyden propuso un método para aproximar matrices jacobianas, siguiendo una idea diferente a la de Davidon pero que produce resultados similares.

En esta sección consideraremos el problema de encontrar una solución al sistema de n ecuaciones con n incógnitas dado por

$$F(x) = 0$$

donde $F : \mathbb{R}^n \to \mathbb{R}^n$. Partiremos de A_k aproximación a $F'(x_k)$ tal que A_{k+1} puede obtenerse de A_k y de la evaluación de F en x_k y en x_{k+1}.

Supondremos $F : \mathbb{R}^n \to \mathbb{R}^n$ continuamente diferenciable en un conjunto abierto y convexo D y que para un $x \in D$ dada y $s \neq 0$ el vector $x_+ = x + s \in D$, entonces dada A queremos una buena aproximación A_+ de $F'(x_+)$.

Es fácil establecer que

$$F'(x_+)(x - x_+) \approx F(x) - F(x_+)$$

si x es cercana a x_+. Entonces si A_+ denota una aproximación a $F'(x_+)$ parece razonable pedir que A_+ satisfaga la ecuación

$$F(x) = F(x_+) + A_+(x - x_+)$$

Esta en general se escribe como

$$A_+ s = y$$

donde $s = x_+ - x$, $y = F(x_+) - F(x)$, ésta es concida como la ecuación de la secante.

Lo que se requiere es que la nueva aproximación sea una corrección de la aproximación anterior, esto es,

$$A_+ = A + C$$

y que no difiera mucho de la aproximación actual A, se pide

$$A_+ \approx A$$

Sabemos que $\mathbb{R}^n = [s] + [s]^\perp$, entonces la condición que se pide es que

$$A_+ s = y, \quad A_+ v = Av, \quad \forall v \perp s$$

Entonces,

$$A_+ s = As + Cs = y, \qquad \Rightarrow \qquad Cs = y - As$$

por otro lado

$$A_+ v = Av + Cv \qquad \forall v \perp s$$

lo que implica

$$Cv = 0, \qquad \text{si } v \perp s$$

Entonces tenemos dos condiciones sobre C

$$\begin{cases} Cs = y - As \\ Cv = 0, \text{ si } v \perp s \end{cases}$$

esto implica que C es de rango 1 y tiene la forma

$$C = uw^t$$

de donde

$$Cs = u(w^t s) = y - As$$

$$\Rightarrow \; u = \lambda(y - As), \text{ con } \lambda = \frac{1}{w^t s}$$

entonces

$$C = \lambda(y - As)w^t = \frac{(y - As)w^t}{w^t s}$$

Ahora falta determinar w, tenemos que

$$Cv = 0, \text{ si } v \perp s$$

entonces

$$Cv = \frac{(y - As)w^t v}{w^t s} = 0$$

lo cual implica

$$w^t v = 0, \text{ esto es, } w = \gamma s$$

de manera que

$$C = \frac{(y - As)(\gamma s)^t}{(\gamma s)^t s} = \frac{(y - As)s^t}{s^t s}$$

y se tiene que

$$A_+ = A + \frac{(y - As)s^t}{s^t s}$$

ésta es la aproximación que Broyden propuso en 1965 como una aproximación a la matriz Jacobiana en la siguiente iteración.

De manera iterativa la forma más simple del método de Broyden es la siguiente:

$$x_{k+1} = x_k - A_k^{-1} F(x_k), \quad k = 0, 1, \ldots \tag{6.2}$$

donde las matrices $A_k \in \mathbb{R}^{n \times n}$ son generadas por

$$A_{k+1} = A_k + \frac{(y - A_k s_k)s_k^t}{s_k^t s_k}, \quad k = 0, 1, \ldots \tag{6.3}$$

con

$$y_k = F(x_{k+1}) - F(x_k), \quad s_k = s_{k+1} - x_k$$

Se tiene también la versión del método de Broyden donde se aproxima la inversa de la matriz Jacobiana, evitándose la resolución del sistema lineal, ésta está basada en el siguiente resultado de Sherman–Morrison.

Lema. Sherman–Morrison *Sea $u, v \in \mathbb{R}^n$ y suponga que $A \in \mathbb{R}^{n\times n}$ es no singular. Entonces $A + uv^t$ es no singular si y sólo si $\sigma = 1/(v^t(A^{-1}u)) \neq 0$. Si $\sigma \neq 0$ entonces*

$$(A + uv^t)^{-1} = A^{-1} - \frac{1}{\sigma}A^{-1}uv^tA^{-1}$$

Del lema anterior se desprende que si $B_k = A_k^{-1}$, entonces $B_{k+1} = A_{k+1}^{-1}$ está dada por

$$B_{k+1} = B_k + \frac{(s_k - B_k y_k)s_k^t B_k}{s_k^t(B_k y_k)} \tag{6.4}$$

siempre que $s_k^t(B_k y_k) \neq 0$. Entonces el método de Broyden también puede ser implementado como

$$x_{k+1} = x_k - B_k F(x_k)$$

donde $\{B_k\}$ es generada por (6.4).

Otra forma de construir una aproximación a la inversa de la matrz Jacobiana es cambiar la condición de la secante. Si B es la aproximación a la inversa de la matriz Jacobiana en la iteración actual, entonces para construir la siguiente aproximación pedimos que

$$B_+ y = s$$

de donde resulta la actualización

$$B_+ = B + \frac{(s - By)y^t}{y^t y}$$

la cual será la nueva aproximación a la inversa de la matriz Jacobiana. Esta formulación también fue introducida por Broyden.

Al igual que con el método de Davidon, esta es una forma de derivar el método de Broyden, mas adelante veremos que éstos forman parte de la familia de actualizaciones tipo secante de cambio mínimo y se describirá otra forma de su derivación que conduce a la misma actualización.

Enseguida veremos que para problemas de optimización se puede encontrar toda una familia de actualizaciones, la familia de Broyden.

6.3 La familia de Broyden

Después de trabajar con sistemas no lineales, Broyden empezó a trabajar el problema de optimización, en donde necesitamos encontrar x^* tal que $\nabla f(x^*) = 0$. Entonces se trabaja con

$$\nabla f(x_0) \approx \nabla f(x_1) + \nabla^2 f(x_1)(x_0 - x_1)$$

trabajaremos tratando de aproximar la inversa de la matriz hessiana, esto es, partiendo de $B_0 \approx \nabla^2 f(x_0)^{-1}$ encontrar $B_+ \approx \nabla^2 f(x_+)^{-1}$, a diferencia de cuando resolvimos sistemas no lineales, aquí las matrices B tienen características especiales, son simétricas y positivas definidas. En este caso la ecuación de la secante, se escribe como

$$B_+ y_0 = s_0$$

donde $s_0 = x_+ - x_0$ y $y_0 = \nabla f(x_+) - \nabla f(x_0)$.

En adelante suprimiremos los subíndices, partiendo de $s = s_0$, $y = y_0$ y $B = B_0$, trataremos de encontrar B_+.

En 1967 Broyden encuentra toda una familia de soluciones para B_+, lo que él propuso fue actualizar en cada iteración con una matriz de corrección de rango dos construida con los vectores s y By; en general estas actualizaciones tienen la forma

$$B_1 = B + \alpha ss^t + \beta(Bys^t + sy^tB) + \gamma Byy^tB$$

con α, β y $\gamma \in \mathbb{R}$.

Esta actualización debe cumplir la condición

$$B_+ y = s$$

entonces

$$s = By + \alpha(ss^t)y + \beta Bys^ty + \beta sy^tBy + \gamma Byy^tBy$$

lo cual implica

$$s = By + \alpha(s^ty)s + \beta(s^ty)By + \beta(y^tBy)s + \gamma(y^tBy)By$$

Igualando coeficientes en s y By resultan condiciones para α, β y γ, éstas satisfacen

$$\begin{cases} 1 &= \alpha(s^ty) + \beta(y^tBy) \\ 0 &= 1 + \beta(s^ty) + \gamma(y^tBy) \end{cases}$$

de donde tenemos dos ecuaciones con tres incógnitas. Broyden parametrizó esta solución considerando

$$\beta = -\frac{\phi}{s^ty}, \quad \phi \in \mathbb{R}$$

de donde de la segunda ecuación, tenemos

$$0 = 1 - \phi + \gamma(y^t B y) \quad \Rightarrow \quad \gamma = \frac{\phi - 1}{y^t B y}$$

y entonces

$$\begin{aligned} \alpha &= \frac{1 - \beta(y^t B y)}{s^t y} \\ &= \left(1 + \phi \frac{y^t B y}{s^t y}\right) \frac{1}{s^t y} \end{aligned}$$

Con estos valores tenemos que

$$B_+ = B + \left(1 + \phi \frac{y^t B y}{s^t y}\right) \frac{1}{s^t y} s s^t - \frac{\phi}{s^t y}(B y s^t + s y^t B) + \frac{\phi - 1}{y^t B y} B y y^t B \tag{6.5}$$

esto implica

$$B_+ = B + \frac{s s^t}{s^t y} - \frac{B y y^t B}{y^t B y} + \phi \left(\frac{y^t B y}{(s^t y)^2} s s^t - \frac{B y s^t + s y^t B}{s^t y} + \frac{B y y^t B}{y^t B y} \right)$$

de donde

$$\begin{aligned} B_+ &= B + \frac{s s^t}{s^t y} - \frac{B y y^t B}{y^t B y} \\ &\quad + \phi(y^t B y) \left(\frac{s}{s^t y} \left(\frac{s}{s^t y} \right)^t - \frac{B y}{y^t B y} \left(\frac{s}{s^t y} \right)^t - \frac{s}{s^t y} \left(\frac{B y}{y^t B y} \right)^t + \frac{B y}{y^t B y} \left(\frac{B y}{y^t B y} \right)^t \right) \end{aligned}$$

entonces

$$B_+ = B + \frac{s s^t}{s^t y} - \frac{B y y^t B}{y^t B y} + \phi(y^t B y) \left(\frac{s}{s^t y} - \frac{B y}{y^t B y} \right) \left(\frac{s}{s^t y} - \frac{B y}{y^t B y} \right)^t$$

que se puede reescribir como

$$B_+ = B + \frac{s s^t}{s^t y} - \frac{B y y^t B}{y^t B y} + \phi(y^t B y)(w w^t) \tag{6.6}$$

con $w = \dfrac{s}{s^t y} - \dfrac{B y}{y^t B y}$. Esta es la familia de Broyden introducida en 1967. Tiene la propiedad de que si B es positiva definida y $\phi \geq 0$ entonces B_+ es positiva definida.

Se ha sugerido una formulación más general que la que da lugar a la familia de Broyden, Huang en 1970 da una familia de tres parámetros en los cuales permite que B no sea simétrica y donde la condición de la secante la escribe como

$$B_k y = \rho_k s \tag{6.7}$$

Un caso especial aquí es la familia simétrica de Huang de dos parámetros en la cual B es simétrica, pero cumple la condición de la secante (6.7), sin embargo la familia de Broyden, $\rho_k = 1$ para toda k es la más importante.

La familia de Broyden contiene dos elementos especiales para casos particulares de ϕ. Si $\phi = 0$,

$$B_+ = B + \frac{ss^t}{s^t y} - \frac{Byy^t B}{y^t B y} \qquad \text{(DFP)}$$

Esta es la actualización de Davidon–Fletcher–Powell donde se hace aproximación a la inversa de la matriz hessiana.

Otro elemento especial de la familia se obtiene al considerar $\phi = 1$, en este caso de (6.5) tenemos

$$B_+ = B + \left(1 + \frac{y^t B y}{s^t y}\right)\frac{ss^t}{s^t y} - \frac{Bys^t + sy^t B}{s^t y} \qquad \text{(BFGS)} \tag{6.8}$$

que es conocida como la actualización BFGS con aproximación a la inversa de la matriz hessiana, ésta fue descubierta en 1970 independientemene por Broyden–Fletcher–Goldfarb–Shanno.

Estas dos actualizaciones son muy importantes y más adelante hablaremos mas sobre ellas.

Existen resultados muy interesantes cuando se trabajan iteraciones de la forma

$$x_{k+1} = x_k + \alpha_k p_k$$

donde $p_k = -B_k \nabla f(x_k)$ con B_k una matriz de la familia de Broyden y α_k obtenida por búsqueda lineal, uno de los mas importantes es que para funciones cuadráticas se alcanza convergencia en n pasos, enunciamos esto en el siguiente resultado.

Teorema. *Un método de Broyden con búsqueda lineal exacta termina después de $m \leq n$ iteraciones en una función cuadrática y se satisface para toda $i = 1, 2, \ldots, m$*

$$B_{i+1} y_j = s_j, \qquad j = 1, 2, \ldots, i \tag{6.9}$$

$$p_i^t A p_j = 0, \qquad j = 1, 2, \ldots, i-1 \tag{6.10}$$

Si $m = n$, entonces $B_{n+1} = A^{-1}$, donde A representa la matriz Hesiana de la función cuadrática.

Demostración.
Procederemos por inducción para (6.9) y (6.10). Para $i = 1$, tenemos que

$$\begin{aligned} B_2 y_1 &= \left(B_1 + \frac{s_1 s_1^t}{s_1^t y_1} - \frac{B_1 y_1 y_1^t B_1}{y_1^t B_1 y_1} + \phi(y_1^t B_1 y_1) w_1 w_1^t \right) y_1 \\ &= B_1 y_1 + s_1 - B_1 y_1 + 0 \\ &= s_1 \end{aligned}$$

Por lo tanto se cumple (6.9) para $i = 1$. (6.10) se satisface por vacuidad en este caso.

Entonces supondremos que se satisface para i con $1 \leq i < m$ y veremos que se satisface también para $i+1$. Demostraremos primero (6.10). Como $\nabla f_{i+1} \neq 0$ la iteración está bien definida con $p_{i+1} \neq 0$ y $\alpha_{i+1} \neq 0$.

Entonces para cualquier $j \leq i$ tenemos que

$$\begin{aligned} Ap_j &= A\left(\frac{x_{j+1} - x_j}{\alpha_j} \right) \\ &= \frac{\nabla f_{i+1} - \nabla f_i}{\alpha_j} \\ &= \frac{y_j}{\alpha_j} \end{aligned}$$

entonces

$$p_{i+1}^t A p_j = \frac{-\nabla f_{i+1}^t B_{i+1} y_j}{\alpha_j}$$

y por (6.9) se tiene que

$$p_{i+1}^t A p_j = \frac{-\nabla f_{i+1}^t s_j}{\alpha_j}$$

pero como estamos utilizando búsqueda lineal exacta y la función es cuadrática se tiene que

$$\nabla f_{i+1}^t p_j = 0$$

por lo tanto

$$p_{i+1}^t A p_j = 0$$

entonces (6.10) se satisface para $i + 1$, por lo tanto se cumple para toda $1 \leq i < m$.

Demostraremos ahora (6.9) para $i+1$. De la ecuación (6.6) tenemos que

$$B_{i+2}y_{i+1} = s_{i+1}$$

Solo falta demostrarlo para $j \leq i$, en este caso tenemos que $B_{i+2}y_j = B_{i+1}y_j$ más términos multiplicados por los escalares

$$y_{i+1}^t B_{i+1} y_j \qquad \text{o} \qquad s_{i+1}^t y_j$$

veremos que estos dos escalares son cero y con esto tendremos completa la demostación de (6.9).

Usando (6.10) con $i+1$ y recordando que $p_{i+1} = \dfrac{s_{i+1}}{\alpha_{i+1}}$, tenemos que

$$p_{i+1}^t A p_j = \frac{s_{i+1}^t}{\alpha_{i+1}} \frac{y_j}{\alpha_j} = 0$$

$$\Rightarrow s_{i+1}^t y_j = 0$$

y por la hipótesis de inducción, tenemos que

$$B_{i+1}y_j = s_j, \qquad j = 1, 2, \ldots, i$$

$$\Rightarrow y_{i+1}^t B_{i+1} y_j = 0$$

Como ambos escalares son cero, entonces tenemos que

$$B_{i+1}y_j = B_{i+1}y_j = s_j$$

para $j \leq i$ y ya habíamos demostrado que se satisface en $i+1$, por lo tanto se satisface para toda i, y con esto termina la demostración de (6.9).

Demostraremos ahora que $B_{n+1} = A^{-1}$. Los vectores p_i, $i = 1, 2, \ldots, m$ por (6.10) son A–conjugados y por lo tanto linealmente independientes de aquí que $m \leq n$. Ahora, sabemos que $A^{-1}y_j = s_j$ para toda j, si $m = n$, entonces por (6.10) y como $\alpha_i \neq 0$, tenemos que los s_i, $i = 1, 2, \ldots, n$ son linealmente independientes y por (6.9) tenemos que

$$B_{n+1}y_j = s_j, \qquad j = 1, 2, \ldots, n$$

lo cual implica que $B_{n+1} = A^{-1}$ que es lo que queríamos demostrar. ■

Mas resultados de convergencia de la familia de Broyden se enuncian mas adelante, después de establecer propiedades sobre los métodos BFGS y DFP. Ahora veremos que estas actualizaciones de Broyden y de la familia de Broyden, forman parte de un conjunto de actualizaciones mas general: las llamadas actualizaciones tipo secante de cambio mínimo.

6.4 Actualizaciones tipo secante de cambio mínimo

El método de Davidon, el método de Broyden y la familia de Broyden son casos particulares de la familia de actualizaciones tipo secante de cambio mínimo. Estas en general, son actualizaciones a la matriz Jacobiana $J(x)$ del problema

$$\text{dada } F : \mathbb{R}^n \to \mathbb{R}^n, \text{ encontrar } x^* \in \mathbb{R}^n \text{ tal que } F(x^*) = 0$$

que como ya mencionamos en el caso en que $F(x) = \nabla f(x)$ estamos resolviendo un problema de optimización. En esta sección trabajaremos actualizaciones directamente a la matriz Jacobiana o la matriz Hessiana en caso de optimización, entonces dada A una aproximación a $J(x)$ queremos actualizarla con A_+ una aproximación a $J(x_+)$. Las aproximaciones usualmente se escogen tales que satisfagan la ecuación de la secante

$$A_+(x_+ - x) = F(x_+) - F(x) \tag{6.11}$$

la cual se satisface exactamente si F es lineal y de aquí que se les llame actualizaciones tipo secante.

Si $n \geq 2$ y $x_+ - x \neq 0$, muchas matrices satisfacen (6.11), si el Jacobiano o la matriz Hesiana, tienen una estructura especial, tal como simetría, entonces cada A_+ se restringe al subconjunto $\mathcal{A}$ de matrices que tienen la propiedad deseada. Se ha encontrado que las mejores actualizaciones son las que escogen A_+ tales que, en alguna norma apropiada, resuelven

$$\min_{A_+ \in \mathcal{A}} \|A_+ - A\| \quad \text{sujeto a (6.11)} \tag{6.12}$$

La estrategia (6.12) es buena ya que ayuda a preservar la información de iteraciones anteriores. Las actualizaciones resultantes son llamadas Actualizaciones tipo secante de cambio mínimo.

Necesitamos escoger una norma adecuada de tal forma que tengamos unicidad en la solución. Se vió que la norma de Frobenius es la que satisface estos requerimientos, las bolas en esta norma son estrictamente convexas y suaves, si $M \in \mathbb{R}^{n\times n}$, $M = (m_{ij})$, entonces la norma de Frobenius se define como

$$\|M\|_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} m_{ij}^2 \right)^{1/2}$$

otra norma que nos será muy útil es la norma ponderada de Frobenius

$$\|W_1 M W_2\|_F, \quad W_1, W_2 \in \mathbb{R}^{n\times n} \text{ no singulares}$$

Entonces estamos interesados en matrices de actualización donde el cambio a la aproximación actual sea tan pequeño como sea posible y donde se satisfaga la ecuación de la secante y quizá algunas otras propiedades. En general escribiremos la ecuación de la secante para A_+ como

$$A_+ s = y, \quad s, y \in \mathbb{R}^n, \quad s \neq 0$$

y definimos el conjunto

$$Q(y, s) = \{M \in \mathbb{R}^{n \times n} : Ms = y\}$$

esto es, $Q(y, s)$ es el conjunto de todas las matrices que satisfacen la ecuación de la secante, donde

$$s = x_+ - x, \qquad y = F(x_+) - F(x)$$

La actualización tipo secante de cambio mínimo mas simple es la que resuelve

$$\min_{A_+ \in Q(y,s)} \|A_+ - A\|_F$$

esto es, solo requerimos que A_+ satisfaga la ecuación de la secante. La solución a este problema es la actualización de Broyden para sistemas de ecuaciones no lineales, como se establece en el siguiente resultado.

Teorema. *Sea $A \in \mathbb{R}^{n \times n}$, $s, y \in \mathbb{R}^n$, $s \neq 0$. La única solución a*

$$\min_{A_+ \in Q(y,s)} \|A_+ - A\|_F \tag{6.13}$$

es

$$A_+ = A + \frac{(y - As)s^t}{s^t s} \tag{6.14}$$

Demostración.
Sea $A_+, C \in Q(y, s)$, entonces

$$\begin{aligned} \|A_+ - A\| &= \left\| \frac{(y - As)s^t}{s^t s} \right\|_F \\ &= \left\| \frac{(Cs - As)s^t}{s^t s} \right\|_F \end{aligned}$$

$$\begin{aligned} &= \left\| \frac{(C-A)ss^t}{s^ts} \right\|_F \\ &= \|C-A\|_F \left\| \frac{s^t}{s^ts} \right\|_F \\ &= \|C-A\|_F \end{aligned}$$

A_+ es la única solución ya que el mapeo $h : \mathbb{R}^{n\times n} \to \mathbb{R}$ definido por $h(A) = \|A_+ - A\|_F$ es estrictamente convexo en $\mathbb{R}^{n\times n}$, ya que es la norma–2 de una matriz escrita como un n^2 vector, y también por el hecho de que $Q(y,s)$ es un subconjunto convexo de $\mathbb{R}^{n\times n}$. ■

La actualización (6.14) fue introducida por Broyden y ha sido la actualización mas exitosa para aproximar el Jacobiano cuando no hay características especiales de $J(x)$ que tengan que ser reflejadas en A.

También puede derivarse la actualización de cambio mínimo para aproximar la inversa de la matriz Jacobiana. En este caso se resuelve

$$\min_{B_+ \in Q(s,y)} \|B_+ - B\|_F$$

la solución a este problema viene dada por

$$B_+ = B + \frac{(s - By)y^t}{y^ty}$$

con la cual se aproxima a la inversa de la matriz Jacobiana.

Note que son las mismas actualizaciones que se obtuvieron en la sección del método de Broyden solo que por otro camino.

Veremos ahora las ideas cuando requerimos características especiales en A_+.

Primeramente necesitamos que las matrices pertenezcan a S_1, el subespacio de las matrices simétricas, esto es,

$$S_1 = \{M \in \mathbb{R}^{n\times n} : M = M^t\}$$

entonces requerimos que las matrices cumplan la ecuación de la secante y al mismo tiempo que sean simétricas, esto es, queremos que

$$A_+ \in Q(y,s) \cap S_1$$

El problema que queremos resolver es

$$\min_{A_+ \in Q(y,s) \cap S_1} \|A_+ - A\|_F$$

Powell fue el primero en trabajar en esta dirección, lo que él hizo fue aplicar directamente la actualización de Broyden a A, esto es, obtener

$$A_1 = A - \frac{(y - As)s^t}{s^t s}$$

y se tiene que $A_1 \in Q(y,s)$, pero A_1 no es necesariamente simétrica, entonces lo que se hace en proyectar A_1 sobre S_1 y obtener $A_2 \in S_1$, dada por

$$A_2 = \frac{A_1 + A_1^t}{2}$$

pero ésta puede no pertencer a $Q(y,s)$, entonces se proyecta sobre este subespacio calculando A_3 la actualización de Broyden de A_2, y luego se hace simétrica a A_3 obtieniendo A_4, y asi sucesivamente, para generar la sucesión

$$A_{2k+1} = A_{2k} - \frac{(y - A_{2k}s)s^t}{s^t s}$$

$$A_{2k+2} = \frac{1}{2}\left(A_{2k+1} + A_{2k+1}^t\right)$$

Powell en 1970 demostró que esta sucesión converge a la matriz

$$A_+ = A + \frac{(y - As)s^t + s(y - As)^t}{s^t s} - \frac{(y - As)^t s(ss^t)}{(s^t s)^2} \tag{6.15}$$

la cual es conocida como la actualización de Powell simétrica tipo Broyden, *(PSB Powell-Symmetric-Broyden)*. Geométricamente, nos apoyamos en la figura (6.1) para ilustrar esta sucesión.

Esto es, (6.15) es la actualización tipo secante de cambio mínimo simétrica, como lo establecemos formalmente en el siguiente resultado.

Teorema. *La única solución a*

$$\min_{A_+ \in Q(y,s) \cap S_1} \|A_+ - A\|_F$$

es

$$A_+ = A + \frac{(y - As)s^t + s(y - As)^t}{s^t s} - \frac{s^t(y - As)ss^t}{(s^t s)^2} \tag{6.16}$$

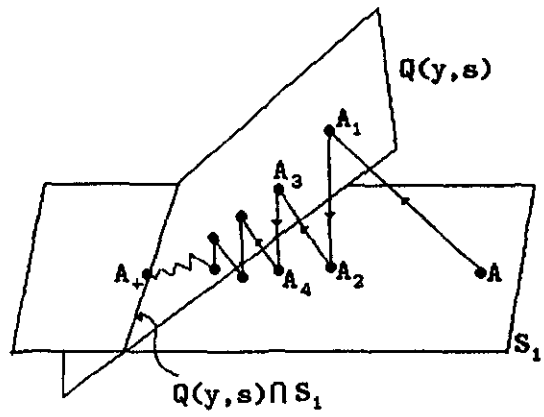


Figura 6.1: Proceso de Powell

Demostración.
Sea C una matriz simétrica que cumpla que $Cs = y$, entonces

$$A_+ - A = \frac{(y - As)s^t + s(y - As)^t}{s^t s} - \frac{s^t(y - As)ss^t}{(s^t s)^2}$$

Pero como $y - As = Cs - As = (C - A)s$, entonces

$$\bar{E} = A_+ - A = \frac{(C - A)ss^t + ss^t(C - A)^t}{s^t s} - \frac{s^t(C - A)s}{(s^t s)^2} ss^t$$

Si definimos $E = C - A$, tenemos

$$\bar{E} = \frac{Ess^t + ss^t E^t}{s^t s} - \frac{s^t Es}{(s^t s)^2} ss^t$$

Ahora

$$\|\bar{E}s\| = \|A_+ s - As\| = \|y - As\| = \|(C - A)s\| = \|Es\|$$

Si v es ortogonal a s tenemos que

$$\|\bar{E}v\| \leq \|Ev\|$$

lo cual implica que

$$\|\bar{E}\|_F \leq \|E\|_F$$

La unicidad de A_+ se sigue de la convexidad del mapeo en la norma de Frobenius en el conjunto de matrices simétricas que pertenecen a $Q(y,s)$.

■

La actualización PSB no ha sido muy exitosa ya que no preserva la propiedad de positiva definida de las matrices. Fue entonces que se pensó en una actualización de cambio mínimo que preservara esta propiedad para poder aplicarlas a problemas de optimización. Esto se logra haciendo la minimización en la norma ponderada de Frobenius, donde la matriz de ponderación debe escogerse adecuadamente para lograr la propiedad deseada.

Antes de ver la matriz de ponderación adecuada para este problema, veremos el resultado que caracteriza las actualizaciones tipo secante de cambio mínimo en la norma ponderada de Frobenius.

Teorema. *Sea $W \in \mathbb{R}^{n\times n}$ simétrica y no-singular. Entonces la única solución a*

$$\min_{A_+ \in Q(y,s)\cap S_1} \|W(A_+ - A)W\|_F \tag{6.17}$$

es

$$A_+ = A + \frac{(y - As)v^t + v(y - As)^t}{v^t s} - \frac{s^t(y - As)vv^t}{(v^t s)^2} \tag{6.18}$$

con $v = W^{-2}s$.

Demostración.
Definamos $C = WAW$, $C_+ = WA_+W$, como W, A, A_+ son simétricas, se tiene que $C, C_+ \in S_1$. Entonces (6.17) es equivalente a

$$\min_{C_+ \in Q(Wy, W^{-1}s)\cap S_1} \|C_+ - C\|_F$$

y por el teorema anterior, la única solución a este problema viene dada por

$$\begin{aligned} C_+ &= C + \frac{(Wy - CW^{-1}s)(W^{-1}s)^t + (W^{-1}s)(Wy - CW^{-1}s)^t}{(W^{-1}s)^t(W^{-1}s)} \\ &\quad - \frac{(W^{-1}s)^t(Wy - CW^{-1}s)W^{-1}s(W^{-1}s)^t}{((W^{-1}s)^t(W^{-1}s))^2} \end{aligned}$$

Ahora sustituyendo $C_+ = WA_+W$, $C = WAW$ y $v = W^{-2}s$, tenemos que

$$WA_+W = WAW + \frac{Wys^tW^{-1} - WAss^tW^{-1} + W^{-1}sy^tW - W^{-1}ss^tAW}{v^ts} - \frac{(s^ty - s^tAs)(W^{-1}ss^tW^{-1})}{(v^ts)^2}$$

De donde pre y postmultiplicando por W^{-1}, tenemos

$$A_+ = A + \frac{ys^tW^{-2} - Ass^tW^{-2} + W^{-2}sy^t - W^{-2}ss^tA}{v^ts} - \frac{s^t(y - As)(W^{-2}ss^tW^{-2})}{(v^ts)^2}$$

lo cual implica que

$$A_+ = A + \frac{(y - As)v^t + v(y - As)^t}{v^ts} - \frac{s^t(y - As)vv^t}{(v^ts)^2}$$

que es lo que queríamos demostrar. ■

Este resultado nos ayudará a encontrar actualizaciones ponderadas, veremos ahora cuál es la ponderación que nos conviene.

Los algoritmos de minimización utilizan aproximaciones cuadráticas locales a $f(x)$ y entonces es deseable tener transformaciones del problema que hagan que estas aproximaciones cuadráticas tengan un buen comportamiento. En particular, si la matriz Hessiana en la solución $\nabla^2 f(x^*)$ es positiva definida, la transformación

$$\hat{x} = \nabla^2 f(x^*)^{1/2}x$$

la cual hace que el espacio de variables para el cual las curvas de nivel de la aproximación cuadrática alrededor de x^* son círculos, es ideal. Esta transformación que para matrices Hessianas corresponde a una ponderación de

$$\nabla^2 f(x^*)^{-1/2}$$

en cada lado, puede considerarse un escalamiento natural del problema de optimización y entonces es deseable usar esta ponderación al medir el cambio de las aproximaciones Hessianas.

Sin embargo, resolver para la actualización tipo secante simétrica que minimiza

$$\|\nabla^2 f(x^*)^{-1/2}(A_+ - A)\nabla^2 f(x^*)^{-1/2}\|_F$$

es imposible ya que no conocemos x^*. Lo mejor que podemos hacer es reemplazar $\nabla^2 f(x^*)$ por alguna matriz $\bar{A}$ del conjunto factible de actualizaciones $Q(y,s) \cap S_1$, ya que este espacio contiene la mejor aproximación a $\nabla^2 f(x^*)$ en este momento. Esto significa utilizar una matriz de ponderación

$$W = \bar{A}^{-1/2}$$

en (6.18). Esto es posible solo si $\bar{A}$ es positiva definida, enseguida veremos que esta es una suposición válida. Entonces usando el último teorema vemos que la solución a

$$\begin{array}{l} \min \\ A_+, \bar{A} \in Q(y,s) \cap S_1 \\ \bar{A} \text{ positiva definida} \end{array} \quad \|\bar{A}^{-1/2}(A_+ - A)\bar{A}^{-1/2}\|_F$$

es

$$A_+ = A + \frac{(y - As)y^t + y(y - As)^t}{y^t s} - \frac{s^t(y - As)yy^t}{(y^t s)^2} \tag{6.19}$$

Esta es la actualización DFP, Davidon–Fletcher–Powell con aproximación directa a la matriz Hessiana.

La actualización DFP fue la actualización mas efectiva para problemas de minimización por varios años. Una de las ventajas de esta actualización comparada con PSB es que si $y^t s > 0$ entonces A_+ es positiva definida siempre que A lo sea.

Aunque la actualización DFP trabaja razonablemente bien, desde 1970 se ha visto que una actualización aparentemente superior para minimización sin restricciones, resulta de escoger la actualización tipo secante simétrica A_+ que minimiza $(A_+^{-1} - A^{-1})$ en la norma de Frobenius poderada apropiada.

Como la ponderación ideal de la aproximación de la Hessiana es $\nabla^2 f(x^*)^{-1/2}$, la ponderación ideal de la aproximación a la inversa de la Hessiana es $\nabla^2 f(x^*)^{1/2}$ y haciendo un proceso similar al anterior, resolvemos

$$\begin{array}{l} \min \\ A_+, \bar{A} \in Q(y,s) \cap S_1 \\ \bar{A} \text{ positiva definida} \end{array} \quad \|\bar{A}^{1/2}(A_+^{-1} - A^{-1})\bar{A}^{1/2}\|_F$$

donde A se supone no singular. La solución a este problema viene dada por

$$A_+^{-1} = A^{-1} + \frac{(s - A^{-1}y)s^t + s(s - A^{-1}y)^t}{s^t y} - \frac{y^t(s - A^{-1}y)ss^t}{(s^t y)^2} \tag{6.20}$$

Esta es la actualización BFGS. También tiene la propiedad de que A_+ es positiva definida si A es positiva definida y $y^t s > 0$.

Entonces al hacer aproximaciones a la matriz Hesiana por una actualización tipo secante, la mejor medida de cambio parece ser $W(A_1^{-1} - A^{-1})W$, donde W es una matriz simétrica que transforma el espacio de variables x en un nuevo espacio $\hat{x} = Wx$ para el cual la aproximación cuadrática local del problema cerca de la solución se comporta bastante bien. Este consejo ha sido corroborado en la práctica, ya que BFGS es, hasta este momento, la mejor elección al hacer la actualización.

Solo queda un detalle por definir en un método Quasi–Newton, Cómo debe escogerse la primera aproximación B_0. Desafortunadamente no hay fórmula que trabaje bien en todos los casos. Se puede utilizar información específica del problema, por ejemplo, la aproximación a la matriz Hesiana calculada por diferencias finitas en x_0, o simplemente un múltiplo de la matriz identidad que refleje un escalamiento de las variables, si no se conoce nada del problema en la práctica $B_0 = I$ ha dado buenos resultados.

6.5 BFGS y DFP

En la sección anterior derivamos las actualizaciones BFGS y DFP como aproximaciones a la inversa de la matriz Hesiana y a la matriz Hesiana, respectivamente. Es interesante hacer notar que estas actualizaciones son duales una de la otra en el sentido de que una puede obtenerse de la otra intercambiando $s \leftrightarrow y$, $A \leftrightarrow A^{-1}$. Esto da lugar a las actualizaciones directas que aproximan directamente a la matriz Hesiana y a las actualizaciones inversas que tratan de llegar a la inversa de la matriz Hesiana. Tanto BFGS como DFP tienen estas dos versiones.

En la derivación anterior obtuvimos DFP resolviendo

$$\min_{\substack{A_+, \bar{A} \in Q(y,s) \cap S_1 \\ \bar{A} \text{ positiva definida}}} \|\bar{A}^{-1/2}(A_+ - A)\bar{A}^{-1/2}\|_F$$

y obtuvimos

$$A_+ = A + \frac{(y - As)y^t + y(y - As)^t}{y^t s} - \frac{s^t(y - As)yy^t}{(y^t s)^2} \tag{6.21}$$

que aproxima directamente a la matriz Hesiana. Si en la ecuación anterior utilizamos la fórmula de Sherman–Morrison, obtenemos la actualización DFP que aproxima a la

inversa de la matriz hesiana, ésta viene dada por

$$B_+ = B + \frac{ss^t}{s^t y} - \frac{Byy^t B}{y^t By}$$

De forma análoga en la sección anterior resolviendo

$$\begin{array}{ll} \min & \|\bar{A}^{1/2}(A_+^{-1} - A^{-1})\bar{A}^{1/2}\|_F \\ A_+, \bar{A} \in Q(y,s) \cap S_1 & \\ \bar{A} \text{ positiva definida} & \end{array}$$

y haciendo $B = A^{-1}$, $B_+ = A_+^{-1}$, obtuvimos

$$B_+ = B + \frac{(s - By)s^t + s(s - By)^t}{s^t y} - \frac{y^t(s - By)ss^t}{(s^t y)^2} \tag{6.22}$$

que es la actualización BFGS con aproximación a la inversa de la matriz Hesiana, pero se puede obtener la correspondiente aproximación directa, utilizando nuevamente Sherman–Morrison, obtenemos

$$A_+ = A + \frac{yy^t}{y^t s} - \frac{Ass^t A}{s^t As}$$

6.6 Convergencia

Enunciaremos los resultados de convergencia mas importantes de los métodos Quasi–Newton.

Iniciamos con el resultado de convergencia local tanto de DFP como de BFGS. Para esto definimos como método BFGS (DFP) un método de la forma

$$x_{k+1} = x_k + \alpha_k p_k, \qquad k = 0, 1, \ldots \tag{6.23}$$

donde $p_k = -B_k \nabla f(x_k)$ y B_k son las matrices de actualización BFGS (DFP) con aproximación a la inversa de la matriz Hesiana. Con esta definición, tenemos el siguiente resultado.

Teorema. (Broyden, Dennis, Moré, 1973) *Sea $f : \mathbb{R}^n \to \mathbb{R}$, $f \in C^2$ en un conjunto abierto y convexo D, suponga que $\nabla f(x^*) = 0$ y $\nabla^2 f(x^*)$ es positiva definida para alguna $x^* \in D$. Suponga también que*

$$\|\nabla^2 f(x) - \nabla^2 f(x^*)\| \leq \kappa \|x - x^*\|, \quad x \in D$$

y considere los métodos BFGS y DFP definidos por (6.23) con $\alpha_k = 1$. *Entonces los métodos BFGS y DFP son localmente y linealmente convergentes a* x^*.

De hecho condiciones mas estrictas con la búsqueda lineal demuestran convergencia superlineal en ambos métodos. La demostración de este teorema se puede consultar en [16].

Los resultados de convergencia de BFGS y DFP nos interesan porque pueden de alguna manera generalizarse a la familia de Broyden completa, como veremos enseguida. Recordemos que la familia de Broyden está dada por

$$B_+ = B + \frac{ss^t}{s^t y} - \frac{Byy^t B}{y^t By} + \phi(y^t By)(ww^t)$$

con $w = \dfrac{s}{s^t y} - \dfrac{By}{y^t By}$ y que DFP y BFGS son elementos particulares de esta familia.

Si trabajamos un método de descenso de la forma

$$x_{k+1} = x_k + \alpha_k p_k$$

donde $p_k = -B_k \nabla f(x_k)$ con B_k una matriz de la familia de Broyden y α_k obtenida por búsqueda lineal exacta, entonces tenemos que

$$p_{k+1} = -B_{k+1} \nabla f(x_{k+1})$$

como la búsqueda lineal es exacta entonces se puede demostrar [19]

$$p_{k+1} = w \left(\frac{y^t s}{y^t By} - \phi w^t \nabla f(x_k) \right) \tag{6.24}$$

Esta ecuación tiene una implicación fundamental. Muestra que si se utiliza búsqueda lineal exacta, el efecto de diferentes elecciones de ϕ_k es sólo un cambio de longitud de p_{k+1} y no de su dirección. Entonces podemos esperar que un método que utiliza alguna actualización de la familia de Broyden va a ser independiente de ϕ_k, este resultado fue descubierto por Dixon en 1972, lo enunciamos en la forma de Powell.

Teorema. (Dixon 1972) *Cuando se aplica a cualquier función* C^1 *un método de Broyden con búsqueda lineal exacta, se tiene la propiedad de que para toda* $k \geq 1$,

$$x_{k+1} \text{ son independientes de } \phi_1, \phi_2, \ldots, \phi_{k-1}$$

suponiendo que los mínimos locales en la búsqueda lineal se resuelven consistentemente, que se evitan valores degenerados de ϕ *y que el algoritmo está bien definido.*

La demostración está basada en la observación (6.24), esto es, con búsqueda lineal exacta las direcciones generadas por los métodos de la familia de Broyden difieren solo en sus longitudes. La búsqueda lineal identifica el mismo mínimo sobre la dirección de búsqueda escogida, por lo que los valores del parámetro de la búsqueda lineal pueden diferir debido al escalamiento, pero la sucesión $\{x_k\}$ generada es la misma para cualquier elemento de la familia de Broyden.

Este resultado es muy importante ya que permite extender resultados de convergencia y orden de convergencia de un elemento de la familia a toda la familia completa. Uno de tales resultados debido a Powell [41], [42] en 1971 y 1972 establece que

Teorema. *Si $f(x)$ es convexa, entonces el método DFP con búsqueda lineal exacta converge globalmente con convergencia superlineal si $\nabla^2 f(x^*)$ es positiva definida.*

Entonces por el teorema de Dixon, éste resultado se extiende a toda la familia de Broyden.

Sin embargo, en la práctica las iteraciones generadas con elementos distintos de la familia de Broyden no generaban las mismas sucesiones y esto se debe a la inexactitud de la búsqueda lineal. En la práctica no se trabaja con búsquedas lineales exactas, entonces es importante considerar los resultados que sean comunes a todos los métodos de Broyden cuando se utilice búsqueda lineal inexacta, en esta dirección, se tienen resultados de convergencia de la familia de Broyden completa sin DFP.

Las condiciones sobre la búsqueda lineal para garantizar convergencia son las condiciones de Wolfe–Powell, enunciamos enseguida el resultado al respecto.

Teorema. (Powell) *Sea $f : \mathbb{R}^n \to \mathbb{R}$ dos veces diferenciable y convexa en $\mathbb{R}^n$ y suponga que dada un $x_0 \in \mathbb{R}^n$ el conjunto de nivel $\{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ está acotado. Suponga que $\{x_k\}$ es generada por (6.23) y que B_k se escoge como cualquier elemento de la familia de Broyden excepto DFP y α_k satisface las condiciones de Wolfe–Powell, entonces para cualquier matriz simétrica positiva definida B_0 y un $\epsilon > 0$ existe un índice k tal que $\|\nabla f(x_k)\| < \epsilon$.*

La primera versión de este teorema trabajaba solo el caso BFGS y fue demostrado por Powell [43]. Byrd, Nocedal y Yuan [12] generalizan el resultado para toda la familia de Broyden excepto DFP.

Referencias

J. Dennis, J. Moré, (1977). Quasi–Newton Methods, Motivation and Theory, *SIAM review*, vol. 19, No.1, pp.46–89.

J. Dennis, R. Schnabel, (1979). Least Change Secant Updates for Quasi–Newton Methods, *SIAM Review* vol. 21, no. 4, pp. 443–459.

J. Nocedal and J. Wright (1999). *Numerical Optimization*, Springer Series in Operations Research.

Capítulo 7

Métodos para problemas de gran escala

7.1 Gradiente Conjugado no Lineal

Cuando se tienen problemas con un gran número de variables se buscan métodos de optimización económicos que operen bien y con pocos recursos. Los métodos de Gradiente Conjugado fueron los primeros métodos que se utilizaron para resolver problemas con un gran número de variables, iniciando con el Gradiente Conjugado Lineal, introducido a principio de los 50's, seguido del Gradiente Gonjugado no Lineal en los 60's. El método de gradiente conjugado lineal ya se ha descrito con detalle en el capítulo 3, este método es muy económico ya que su implementación solo requiere el almacenamiento de pocos vectores, por lo que se ha convertido en un método bastante atractivo para problemas cuadráticos de gran escala por los pocos recursos en memoria necesitados para operar.

Una nota interesante respecto a estos métodos es que el gradiente conjugado lineal fue desarrollado en los 50's como una alternativa a métodos de factorización para resolver soluciones exactas de sistemas lineales con matrices simétricas positivas definidas. No fue sino hasta varios años después, en uno de los mas importantes desarrollos del álgebra lineal sparse, que el método se vio como un método iterativo que puede dar buenas aproximaciones a la solución del sistema en mucho menos que n pasos. El primer método de gradiente conjugado no lineal fue propuesto cuando el gradiente conjugado lineal perdía popularidad y varios años antes de que fuera redescubierto como un método iterativo. Veremos en esta sección las principales ideas del método de Gradiente Conjugado no lineal.

7.1.1 Método de Fletcher–Reeves

Hemos visto que el método de Gradiente Conjugado lineal es utilizado para minimizar funciones cuadráticas, es entonces natural preguntarnos si se puede adaptar para la minimización de funciones en general $f(x)$. En los años 60's Fletcher y Reeves [22] encontraron que esta extensión es posible, dando lugar a métodos de gradiente conjugado no lineal. El algoritmo para el caso no lineal de Fletcher–Reeves cambia dos aspectos del caso lineal: α_k se escoge por algún método de búsqueda lineal y el cálculo de β_k involucra el gradiente de la función, g_k, que no es el residual como en el caso lineal. Como la función ya no es cuadrática y la búsqueda lineal no es exacta, no se garantiza la convergencia en n pasos, entonces se propuso realizar recomienzos cada n pasos con la dirección de descenso mas rápido para alcanzar convergencia.

Esto da lugar al algoritmo de Gradiente Conjugado de Fletcher–Reeves para optimización no lineal.

Algoritmo de Gradiente Conjugado de Fletcher–Reeves
Escoger un punto inicial x_0, evaluar f_0 y g_0, hacer $p_0 = -g_0$ y $k = 0$.
while $g_k \neq 0$
calcular $\alpha_k = \arg\ \min_{\alpha \geq 0} f(x_k + \alpha p_k)$
$x_{k+1} = x_k + \alpha_k p_k$
Evaluar g_{k+1}
If $k \equiv 0 \bmod n$ entonces
$\beta_{k+1}^{FR} = 0$
else
$$\beta_{k+1}^{FR} = \frac{g_{k+1}^t g_{k+1}}{g_k^t g_k}$$
end
$p_{k+1} = -g_{k+1} + \beta_{k+1}^{FR} p_k$
$k = k + 1$
end while

La búsqueda lineal requiere un buen valor inicial para α_k, el cual es crucial si se quiere obtener un algoritmo eficiente. Fletcher experimentalmente obtuvo que en el gradiente conjugado el decrecimiento de la función de x_{k-1} a x_k era del mismo orden de magnitud que el de x_k a x_{k+1}, suponiendo que $f(x)$ es cuadrática, entonces tenemos que el tamaño de paso en la iteración anterior α_{k-1} y la actual α_k están relacionadas por la expresión

$$\alpha_k = \alpha_{k-1} \frac{p_{k-1}^t g_{k-1}}{p_k^t g_k}$$

entonces puede considerarse éste como valor inicial para iniciar la búsqueda lineal para el caso de gradiente conjugado.

Si f es una cuadrática y si el tamaño de paso α_k es el minimizador exacto de la búsqueda lineal, entonces el método de Fletcher–Reeves es idéntico al de gradiente conjugado lineal. Nótese que no necesitamos ninguna operación matricial y que solo necesitamos almacenar unos cuantos vectores, es por esto que el método es muy utilizado para resolver problemas grandes de optimización no lineal.

7.1.2 Método de Polak–Ribiére

Existen muchas variantes del método de Fletcher–Reeves que difieren principalmente en la elección del parámetro β_k. Una de las mas importantes fue propuesta por Polak y Ribiére [40], en donde se considera que como la función no es cuadrática no tiene porqué satisfacerse la condición de ortogonalidad de los gradientes y proponen tomar β_{k+1} como

$$\beta_{k+1}^{PR} = \frac{g_{k+1}^t(g_{k+1} - g_k)}{\|g_k\|^2}$$

Esta β coincide con el parámetro de Fletcher–Reeves cuando f es una función cuadrática y convexa y con búsqueda lineal exacta, ya que los gradientes son mutuamente ortogonales en este caso. Sin embargo, al aplicarlo a funciones no lineales el parámetro β_k^{PR} produce iteraciones diferentes al algoritmo de Fletcher–Reeves. La experiencia numérica favorece al método de Polak–Ribiére el cual es mas robusto y eficiente.

Se ha observado en la práctica que si el método de Polak–Ribiére genera una mala dirección y un paso muy pequeño en una iteración, por la forma de β_k^{PR}, el algoritmo se recupera en la siguiente iteración, mientras que para Fletcher–Reeves, el algoritmo no se recupera y genera otra nueva dirección y tamaño paso también malos, entrando en un ciclo del que no puede recuperarse al menos que se realice un recomienzo. Mas adeltante justificamos estos comentarios.

Existen otras elecciones de β_{k+1} que coinciden con la de Fletcher–Reeves β_{k+1}^{FR} en el caso en que la función objetivo es cuadrática y la búsqueda lineal exacta, por ejemplo Hestenes–Stiefel sugieren

$$\beta_{k+1}^{HS} = \frac{g_{k+1}^t(g_{k+1} - g_k)}{(g_{k+1} - g_k)^t p_k}$$

esta formulación es similar al método de Polak–Ribiére, tanto en convergencia como en desarrollo en la práctica.

Se han propuesto otras variantes de β_k pero no se ha logrado superar la eficiencia de la fórmula de Polak–Ribiére.

7.1.3 Recomienzos

La parte mas costosa del método de gradiente conjugado es la búsqueda lineal, para hacerlo mas económico es que se utiliza la búsqueda lineal inexacta, sin embargo, esto puede ocasionar que la dirección p_{k+1} no sea de descenso, ya que de acuerdo al algoritmo

$$g_k^t p_k = -\|g_k\|^2 + \beta_k^{FR} g_k^t p_{k-1} \tag{7.1}$$

si la búsqueda lineal es exacta, entonces $g_k^t p_{k-1} = 0$ y p_k será una dirección de descenso. Pero para búsquedas lineales inexactas (7.1) puede ser positivo si $g_k^t p_{k-1}$ es positivo y no suficientemente pequeño.

Esto puede resolverse usando periódicamente un recomienzo, es decir, $\beta_{k+1} = 0$, lo que equivale a hacer $p_{k+1} = -g_{k+1}$, esto es, reiniciar con la dirección de máximo descenso, aunque frecuentes recomienzos pueden afectar la eficiencia del método por lo que a pesar de que la búsqueda lineal se inexacta, debe ser una buena aproximación al mínimo real.

Criterios de recomienzo

Como ya mencionamos, el método de gradiente conjugado mejora mucho su efectividad si se recomienza cada n iteraciones, pero cuando queremos resolver problemas de grandes dimensiones queremos evitar el realizar n pasos para reiniciar el proceso, de hecho queremos encontrar la solución en mucho menos de n pasos. Entonces los métodos de gradiente conjugado se implementan frecuentemente sin recomienzos o se incluyen otras estrategias para recomenzar distinta al número de iteraciones. Uno de los criterios de recomienzo mas utilizados en la práctica está basado en la propiedad de ortogonalidad de los gradientes para el caso lineal, en el caso no lineal es necesario prevenir que los gradientes consecutivos no se alejen mucho de la ortogonalidad, una forma de hacerlo recomendada por Powell [44] es recomenzar si

$$\frac{|g_{k+1}^t g_k|}{\|g_k\|^2} > \gamma$$

donde γ es un número positivo menor que uno. El valor de $\gamma = 0.2$ es ampliamente recomendado.

Otro criterio para hacer un recomienzo, también recomendado por Powell, es cuando la dirección p_{k+1} no es de suficiente descenso, esto es, si

$$\frac{g_{k+1}^t p_{k+1}}{\|g_{k+1}\|^2} \notin [-1.2, -0.8]$$

entonces es conveniente recomenzar.

7.1.4 Búsqueda Lineal para Gradiente Conjugado

La única parte del algoritmo que necesita especificarse es el cálculo del tamaño de paso, α_k. La búsqueda lineal debe proveer una reducción suficiente en la función objetivo, pero para esto primero debemos asegurar que las direcciones de búsqueda sean direcciones de descenso para f. Para esto la búsqueda lineal debe proveer un balance correcto entre los dos términos de (7.1). Mostraremos que esto se logra pidiendo que α_k satisfaga las condiciones fuertes de Wolfe–Powell

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \rho \alpha_k g_k^t p_k$$

$$|g(x_k + \alpha_k p_k)^t p_k \leq \sigma |g_k^t p_k|$$

con $0 < \rho < \sigma < 1/2$. La restricción $\sigma < 1/2$ es importante, de otra forma no se garantiza el descenso en el método de Fletcher–Reeves. Entonces la búsqueda lineal requiere mas precisión que para otros métodos de optimización. Establecemos esto en nuestro siguiente resultado.

Lema. *Supongamos que el método de Fletcher–Reeves está implementado con un tamaño de paso α_k que satisface las condiciones fuertes de Wolfe–Powell con $0 < \sigma < 1/2$. Entonces el método genera direcciones de descenso p_k que satisfacen*

$$-\frac{1}{1-\sigma} \leq \frac{g_k^t p_k}{\|g_k\|^2} \leq \frac{2\sigma - 1}{1-\sigma}, \qquad k = 1, 2, \ldots \tag{7.2}$$

Demostración.
Nótese que la función $t(\epsilon) = (2\epsilon - 1)/(1 - \epsilon)$ es monótona creciente en el intervalo $[0, 1/2]$, con $t(0) = -1$ y $t(1/2) = 0$. Como $\sigma \in [0, 1/2)$ tenemos que

$$-1 \leq \frac{2\sigma - 1}{1-\sigma} < 0 \tag{7.3}$$

Entonces p_k será una dirección de descenso si se satisface (7.2).

Procederemos por inducción. Para $k = 0$ tenemos que

$$\frac{g_0^t(-g_0)}{\|g_0\|^2} = -1$$

por lo tanto de (7.3) tenemos que el resultado se satisface en este caso.

Ahora, supongamos que (7.2) se satisface para alguna $k \geq 1$. Por el algoritmo de Gradiente Conjugado no lineal tenemos que

$$p_{k+1} = -g_{k+1} + \beta_{k+1} p_k$$

de donde

$$\begin{aligned} \frac{g_{k+1}^t p_{k+1}}{\|g_{k+1}\|^2} &= \frac{-g_{k+1}^t g_{k+1}}{\|g_{k+1}\|^2} + \beta_{k+1} \frac{g_{k+1}^t p_k}{\|g_{k+1}\|^2} \\ &= -1 + \frac{g_{k+1}^t g_{k+1}}{g_k^t g_k} \frac{g_{k+1}^t p_k}{\|g_{k+1}\|^2} \\ &= -1 + \frac{g_{k+1}^t p_k}{\|g_k\|^2} \end{aligned} \tag{7.4}$$

Por la segunda condición de Wolfe–Powell, tenemos que

$$|g_{k+1}^t p_k| \leq -\sigma g_k^t p_k$$

lo cual implica que

$$\sigma g_k^t p_k \leq g_{k+1}^t p_k \leq -\sigma g_k^t p_k$$

y combinando esto con (7.4) resulta

$$-1 + \sigma \frac{g_k^t p_k}{\|g_k\|^2} \leq \frac{g_{k+1}^t p_{k+1}}{\|g_{k+1}\|^2} \leq -1 - \sigma \frac{g_k^t p_k}{\|g_k\|^2}$$

pero por hipótesis de inducción

$$-\frac{1}{1-\sigma} \leq \frac{g_k^t p_k}{\|g_k\|^2} \leq \frac{2\sigma - 1}{1-\sigma}$$

usando el lado izquierdo de esta ecuación resulta que

$$-1 - \frac{\sigma}{1-\sigma} \leq \frac{g_{k+1}^t p_{k+1}}{\|g_{k+1}\|^2} \leq -1 + \frac{\sigma}{1-\sigma}$$

lo cual implica que

$$-\frac{1}{1-\sigma} \leq \frac{g_{k+1}^t p_{k+1}}{\|g_{k+1}\|^2} \leq \frac{2\sigma - 1}{1-\sigma}$$

entonces el resultado se satisface también para $k+1$. Por lo tanto, se satisface para toda k, lo cual completa la demostración. ■

Este resultado muestra que es suficiente con pedir la condición de curvatura en la búsqueda lineal para garantizar que el método FR genere direcciones de descenso. La primera condición de Wolfe–Powell la necesitaremos para establecer la convergencia global. Usaremos este resultado para explicar la ineficiencia del método de FR. Argumentaremos que si el método genera una mala dirección y un paso muy pequeño, la siguiente dirección y tamaño de paso serán malos también. Supongamos que en la iteración k se generó una dirección no buena, de tal forma que $\cos\theta_k \approx 0$ donde θ_k es el ángulo entre la dirección de búsqueda y la de descenso más rápido, esto eso, p_k es casi ortogonal a $-g_k$. Tenemos que $\cos\theta_k = -g_k^t p_k / \|g_k\|\|p_k\|$, de (7.2) y de la definición de $\cos\theta_k$, tenemos que para toda k,

$$c_1 \frac{\|g_k\|}{\|p_k\|} \leq \cos\theta_k \leq c_2 \frac{\|g_k\|}{\|p_k\|} \tag{7.5}$$

para algunas constantes c_1 y c_2, de donde tenemos que $\cos\theta_k \approx 0$ solo si

$$\|g_k\| \ll \|p_k\|$$

Supongamos entonces que como las direcciones de búsqueda son casi ortogonales al gradiente, el paso de x_k a x_{k+1} es pequeño, esto es, $x_{k+1} \approx x_k$, entonces $\|g_{k+1}\| \approx \|g_k\|$, y

$$\beta_{k+1}^{FR} \approx 1 \tag{7.6}$$

Entonces tenemos que

$$\|g_{k+1}\| \approx \|g_k\| \ll \|p_k\|$$

Usando esta relación y (7.6) en la definición de p_{k+1} en el algoritmo de GC no lineal, tenemos que

$$\|p_{k+1}\| \approx \|p_k\| \gg \|g_{k+1}\|$$

lo cual por (7.5) implica que $\cos\theta_{k+1} \approx 0$. Aplicando de nuevo este mismo argumento se muestra la ineficiencia de las iteraciones. Un recomienzo en estas circunstancias ayuda a salir de este ciclo de malas direcciones y tamaños de paso muy pequeños.

El método de Polak–Ribiére se comporta muy diferente en estas circunstancias. Supongamos de nuevo que se ha generado una mala dirección de búsqueda en x_k, tal que $\cos\theta_k \approx 0$ y que se ha tomado un tamaño de paso muy pequeño, esto es, $x_{k+1} \approx x_k$. Entonces $\|g_{k+1}\| \approx \|g_k\|$ y se tiene que $\beta_{k+1}^{PR} \approx 0$. Esto significa que la nueva dirección de búsqueda se aproxima a la dirección de descenso más rápido, de tal forma que $\cos\theta_{k+1} \gg \cos\theta_k$. Entonces el método de Polak–Ribiére se recupera de esta situación desfavorable. El mismo argumento es aplicable al método de Hestenes–Stiefel.

7.1.5 Análisis de Convergencia

Presentamos en esta sección los principales resultados conocidos de convergencia para los métodos de Fletcher–Reeves y Polak–Ribiére utilizando búsquedas lineales inexactas.

Iniciamos considerando convergencia global. Si el método de gradiente conjugado reinicia cada n pasos considerando β_k igual a cero, esto es, tomando la dirección de descenso más rápido, entonces se obtiene convergencia global. Esto se sigue del análisis dado en el capítulo de búsqueda lineal: si se toma un número infinito de direcciones de descenso más rápido, y con las condiciones de Wolfe–Powell, entonces tendremos que

$$\liminf_{k\to\infty}\|g_k\| = 0$$

Esto significa que si se reinicia periodicamente, entonces todos los métodos de gradiente conjugado vistos en esta sección son globalmente convergentes a un punto estacionario.

Nos interesa estudiar la convergencia global de métodos de gradiente conjugado sin reinicios, dado que como estos métodos son útiles para resolver problemas grandes, es importante considerar su comportamiento cuando $n \to \infty$. Cuando n es grande, esperamos resolver el problema en menos de n iteraciones y no utilizar reinicios. Para establecer resultados de convergencia, haremos la siguiente suposición sobre la función objetivo.

Suposiciones I.

i) *El conjunto de nivel* $\mathcal{L} = \{x : f(x) \leq f(x_0)\}$ *está acotado.*

ii) *En alguna vecindad* $\mathcal{N}$ *de* $\mathcal{L}$ *la función objetivo* f *es continuamente diferenciable, y si el gradiente es Lipschitz continuo, esto es, existe una constante* $L > 0$ *tal que*

$$\|g(x) - g(\tilde{x})\| \leq L\|x - \tilde{x}\|$$

para toda $x, \tilde{x} \in \mathcal{N}$.

Estas suposiciones implican que existe una constante $\bar{\gamma}$ tal que

$$\|g(x)\| \leq \bar{\gamma}, \quad \text{para toda} \quad x \in \mathcal{L} \tag{7.7}$$

Como en la práctica la búsqueda lineal es inexacta, supondremos también que ésta satisface las condiciones de Wolfe–Powell. Para propósitos del análisis consideraremos también la siguiente búsqueda lineal ideal: un tamaño de paso $\alpha_k > 0$ es aceptable si

$$f(x_k + \alpha_k p_k) \leq f(x_k + \hat{\alpha}_k p_k) \tag{7.8}$$

donde $\hat{\alpha}_k$ es el punto estacionario positivo mas pequeño de la función $\varphi(\alpha) = f(x_k + \alpha p_k)$. Entonces la selección ideal del tamaño de paso requiere que el decrecimiento en la función objetivo sea al menos tan bueno que el que se obtiene en el primer punto estacionario unidimensional de la función. Nótese que tanto el primero mínimo local como el minimizador global de f a lo largo de la dirección de búsqueda satisfacen esta condición.

Esto nos ayudará a establecer el siguiente resultado.

Teorema. *(Zoutendijk) Supongamos que se satisfacen las Suposiciones I. Considere cualquier iteración de búsqueda lineal de la forma*

$$x_{k+1} = x_k + \alpha_k p_k$$

donde p_k es una dirección de descenso y α_k satisface las condiciones de Wolfe–Powell o la condición de la búsqueda lineal ideal (7.8), entonces

$$\sum_{k=1}^{\infty} \cos^2 \theta_k \|g_k\|^2 < \infty \tag{7.9}$$

Este teorema fue demostrado en el capítulo de búsqueda lineal para las condiciones de Wolfe–Powell y es fácil su extensión al caso de búsqueda lineal ideal. También será este resultado la base para el análisis de los métodos de gradiente conjugado no lineal.

Haremos el análisis bajo el supuesto de que

$$\cos \theta_k \geq c \frac{\|g_k\|}{\|p_k\|}, \qquad k = 1, 2, \ldots \tag{7.10}$$

para alguna constante positiva c. Nótese que esto se satisface para el método de Fletcher–Reeves por el resultado (7.2). Sustituyendo esto en la condición de Zoutendijk, obtenemos

$$\sum \frac{\|g_k\|^4}{\|p_k\|^2} < \infty \tag{7.11}$$

Si mostramos que $\{\|g_k\|/\|p_k\|\}$ no tiende a cero, esto es,

$$\frac{\|g_k\|}{\|p_k\|} \geq c' > 0, \qquad k = 1, 2, \ldots \tag{7.12}$$

entonces podemos deducir de (7.11) que

$$\lim_{k \to \infty} \|g_k\| = 0 \tag{7.13}$$

La cota (7.12) se puede establecer para el método de Polak–Ribiére [40] bajo las suposiciones de búsqueda lineal ideal y f *fuertemente convexa* esto es, $(g(x)-g(\tilde{x})^t(x-\tilde{x}) \geq c\|x-\tilde{x}\|^2$ para alguna constante c positiva. Para funciones en general no es fácil establecer la cota (7.12), asi que probaremos un resultado mas débil que (7.13), demostraremos que

$$\liminf_{k\to\infty}\|g_k\| = 0 \tag{7.14}$$

Procederemos por contradicción. Supongamos que no se satisface, lo cual significa que el gradiente no tiende a cero. Entonces existe un escalar $\gamma > 0$ tal que

$$\|g_k\| \geq \gamma \tag{7.15}$$

para toda $k \geq 1$. Sustituyendo en (7.11) tenemos que

$$\sum_{k\geq 1}\frac{1}{\|p_k\|^2} < \infty \tag{7.16}$$

Concluimos que si la iteración falla en el sentido de que se satisface (7.15), entonces $\|p_k\| \to \infty$ suficientemente rápido para que la suma en (7.16) sea finita. Entonces podemos enunciar que: *Un método de gradiente conjugado no lineal puede no tener convergencia global si las direcciones de búsqueda son no acotadas en longitud.*

Usaremos este razonamiento para probar convergencia global para el método de Fletcher–Reeves.

Teorema. *Suponga que se satisfacen las suposiciones I y que se cuenta con el método de Fletcher–Reeves con una búsqueda lineal que satisface las condiciones fuertes de Wolfe–Powell con* $0 < \rho < \sigma < 1/2$. *Entonces*

$$\liminf_{k\to\infty}\|g_k\| = 0$$

Demostración.
Procederemos por contradicción logrando acotar $\|p_k\|$. De la segunda condición de Wolfe–Powell y de (7.2) tenemos

$$|g_k^t p_{k-1}| \leq -\sigma g_{k-1}^t p_{k-1} \leq \frac{\sigma}{1-\sigma}\|g_{k-1}\|^2 \tag{7.17}$$

de esta ecuación, de la forma de obtener p_k en el algoritmo de Fletcher-Reeves y de la definición de β_k^{FR} tenemos

$$\begin{aligned}\|p_k\|^2 &\leq \|g_k\|^2 + 2\beta_k^{FR}|g_k^t p_{k-1}| + (\beta_k^{FR})^2\|p_{k-1}\|^2 \\ &\leq \|g_k\|^2 + \frac{2\sigma}{1-\sigma}\beta_k^{FR}\|g_{k-1}\|^2 + (\beta_k^{FR})^2\|p_{k-1}\|^2 \\ &\leq \left(\frac{1+\sigma}{1-\sigma}\right)\|g_k\|^2 + (\beta_k^{FR})^2\|p_{k-1}\|^2\end{aligned}$$

Aplicando de nuevo esta desigualdad y definiendo $\hat{\sigma} = (1+\sigma)/(1-\sigma) \geq 1$ obtenemos

$$\begin{aligned}\|p_k\|^2 &\leq \hat{\sigma}\|g_k\|^2 + \frac{\|g_k\|^4}{\|g_{k-1}\|^4}[\hat{\sigma}\|g_{k-1}\|^2 + (\beta_{k-1}^{FR})^2\|p_{k-2}\|^2] \\ &\leq \hat{\sigma}\|g_k\|^2 + \hat{\sigma}\frac{\|g_k\|^4}{\|g_{k-1}\|^2} + \frac{\|g_k\|^4}{\|g_{k-2}\|^4}[\hat{\sigma}\|g_{k-2}\|^2 + (\beta_{k-2}^{FR})^2\|p_{k-3}\|] \\ &= \hat{\sigma}\|g_k\|^2 + \hat{\sigma}\frac{\|g_k\|^4}{\|g_{k-1}\|^2} + \hat{\sigma}\frac{\|g_k\|^4}{\|g_{k-2}\|^2} + \frac{\|g_k\|^4}{\|g_{k-3}\|^4}\|p_{k-3}\| \\ &\leq \hat{\sigma}\|g_k\|^4\sum_{j=1}^{k}\|g_j\|^{-2} \qquad (7.18)\end{aligned}$$

Ahora procedemos por contradicción suponiendo que $\|g_k\| \geq \gamma > 0$ para toda k. De la desigualdad (7.18) y de (7.7) tenemos que

$$\|p_k\|^2 \leq \frac{\hat{\sigma}\tilde{\gamma}^4}{\gamma^2}k \qquad (7.19)$$

Ahora utilizamos el proceso descrito antes de enunciar el teorema. Del resultado de Zoutendijk obtenemos

$$\sum_{k\geq 1}\frac{\|g_k\|^4}{\|p_k\|^2} < \infty$$

Si los gradientes no tienden a cero, entonces

$$\sum_{k\geq 1}\frac{1}{\|p_k\|^2} < \infty$$

pero esto contradice (7.19), con lo cual la hipótesis es falsa y tenemos que

$$\liminf_{k\to\infty}\|g_k\| = 0$$

que es lo que queríamos demostrar. ■

Este es un resultado de convergencia global muy importante ya que se aplica a la implementación práctica del método de Fletcher–Reeves para funciones objetivo no lineales en general.

Dado que el método de Polak–Ribiére trabaja mejor en la práctica que el de Fletcher–Reeves, se esperaría tener un resultado de convergencia global para este método similar al teorema anterior. Esto no es posible ya que se enuncia enseguida un resultado que muestra que el método de Polak–Ribiére con una búsqueda lineal ideal puede ciclarse infinitamente, sin aproximarse al punto solución.

Teorema. *Considere un método de gradiente conjugado con β_k como el método de Polak–Ribiére y con búsqueda lineal ideal tal que siempre se escoja el primer punto estacionario positivo de $\varphi(\alpha) = f(x_k + \alpha p_k)$. Existe una función objetivo f dos veces continuamente diferenciable de tres variables y un punto inicial x_0 tal que la sucesión de gradientes $\{\|g_k\|\}$ no tiende a cero.*

La demostración de este resultado se debe a Powell [45], se requiere que algunas direcciones de búsqueda consecutivas sean casi contrarias. Esto solo puede alcanzarse, para búsquedas lineales ideales, cuando $\beta_k < 0$, entonces el análisis sugiere modificar el método de Polak–Ribiére considerando

$$\beta_k^+ = \max\{\beta_k^{PR}, 0\}$$

que da lugar el método PR$^+$. En este caso las condiciones de Wolfe–Powell garantizan que todas las direcciones de búsqueda generadas son direcciones de descenso. El argumento es el siguiente: primero realizamos la búsqueda lineal a lo largo de la dirección de descenso p_{k-1} y obtenemos un punto x_k que satisface las condiciones de Wolfe–Powell con $0 < \rho < \sigma < 1$. Si $g_k^t p_{k-1} \leq 0$, la no negatividad de β_k^+ y (7.1) implican que la nueva dirección p_k satisface $g_k^t p_k < 0$ y de aquí es una dirección de descenso. Por otro lado, si $g_k^t p_{k-1} > 0$, continuamos la minimización unidimensional hasta que el término $|g_k^t p_{k-1}|$ se reduzca lo suficiente, de tal forma que (7.1) garantice la condición de descenso. Usando esto Gilbert y Nocedal [23] demuestran que PR$^+$ es globalmente convergente, bajo las hipótesis del teorema de convergencia global de Fletcher–Reeves. Ellos proponen un método de gradiente conjugado donde β_k está

restringido por el parámetro de Fletcher–Reeves y donde permiten valores negativos para β_k, la propuesta es considerar

$$\beta_k = \begin{cases} -\beta_k^{FR} & \text{si } \beta_k^{PR} < -\beta_k^{FR} \\ \beta_k^{PR} & \text{si } |\beta_k^{PR}| < \beta_k^{FR} \\ \beta_k^{FR} & \text{si } \beta_k^{PR} > \beta_k^{FR} \end{cases} \qquad \text{para } k \geq 0$$

con esta elección el método resultante es globalmente convergente.

Veremos ahora algunos resultados que conectan los métodos de gradiente conjugado con métodos Quasi–Newton dando lugar a algoritmos prácticos mas robustos.

7.2 Método de Shanno

Una de las mayores dificultades que se presenta en el gradiente conjugado es que la búsqueda lineal inexacta ocasiona que las direcciones generadas pueden no ser de descenso, lo que no sucede con los Quasi–Newton, ya que si $p_{k+1} = -H_{k+1}g_{k+1}$ entonces $p_{k+1}^t g_{k+1} < 0$ si H_{k+1} es positiva definida. Entonces es deseable poder combinar la poca cantidad de recursos que necesita gradiente conjugado para operar con las propiedades de los métodos Quasi–Newton para tener un algoritmo eficiente y económico. Shanno [47] muestra que es posible, como lo enunciamos enseguida.

7.2.1 Enfoque Quasi–Newton del método de Gradiente Conjugado

Utilizaremos en esta sección la notación introducida en el capítulo 6 para los métodos Quasi–Newton:

$$y_k = g_{k+1} - g_k, \qquad s_k = x_{k+1} - x_k = \alpha_k p_k$$

Veremos que es posible ver la dirección de gradiente conjugado como una dirección Quasi–Newton.

Perry en 1977 notó que la dirección de Gradiente Conjugado

$$p_{k+1} = -g_{k+1} + \beta_{k+1} p_k$$

con

$$\beta_{k+1} = \frac{g_{k+1}^t y_k}{p_k^t y_k}$$

puede escribirse como

$$p_{k+1} = -\left(I - \frac{s_k y_k^y}{s_k^t y_k}\right) g_{k+1} = -R g_{k+1}$$

donde

$$R = I - \frac{s_k y_k^t}{s_k^t y_k}$$

Esta dirección se parece a las direcciones de Quasi–Newton, lamentablemente, esta matriz no es positiva definida. Sin embargo se modificó por corrección usando la propiedad de que bajo la hipótesis de búsqueda lineal exacta la nueva matriz y R produzcan la misma dirección. El método resultante es más eficiente que los métodos clásicos con búsqueda lineal inexacta.

La matriz R de Perry no es simétrica, por lo que resulta natural simetrizarla considerando

$$R_s = R - \frac{y_k s_k^t}{s_k^t y_k} = I - \frac{s_k y_k^t + y_k s_k^t}{s_k^t y_k}$$

Si la búsqueda lineal es exacta, entonces $s_k^t g_{k+1} = 0$, lo que implica que

$$-R_s g_{k+1} = -R g_{k+1} = p_{k+1}$$

Ahora bien, para obtener una iteración similar a un método Quasi–Newton, necesitamos que

$$p_{k+1} = -R^* g_{k+1}$$

donde R^* debe satisfacer la condición

$$R^* y_k = s_k \tag{7.20}$$

para lo cual se busca una matriz de corrección C tal que

$$R^* = R_s + C$$

satisfaga (7.20), por lo tanto

$$R^* y_k = s_k$$

de donde

$$(R_s + C) y_k = s_k$$

lo que implica que

$$\begin{aligned} Cy_k &= s_k + \frac{y_k^t y_k}{s_k^t y_k} s_k \\ &= \left(1 + \frac{y_k y_k^t}{s_k^t y_k}\right) s_k \\ &= \lambda s_k \end{aligned}$$

Como C debe ser simétrica, esto se puede obtener con una matriz de rango 1, esto es,

$$C = ww^t$$

sustituyendo

$$Cy_k = w(w^t y_k) = \left(1 + \frac{y_k^t y_k}{s_k^t y_k} s_k\right)$$

lo que implica que

$$w = \sigma s_k$$

de donde

$$w^t y_k = \sigma s_k^t y_k \qquad \Rightarrow \qquad \sigma^2 s_k (s_k^t y_k) = \left(1 + \frac{y_k^t y_k}{s_k^t y_k}\right) s_k$$

entonces

$$\sigma^2 = \left(1 + \frac{y_k^t y_k}{s_k^t y_k}\right) \frac{1}{s_k^t y_k}$$

lo cual tiene sentido solo si $s_k^t y_k > 0$. De esto, obtenemos que

$$C = ww^t = \left(1 + \frac{y_k^t y_k}{s_k^t y_k}\right) \frac{1}{s_k^t y_k} s_k s_k^t$$

de donde como $R^* = R_s + C$ tenemos

$$R^* = I - \frac{s_k y_k^t + y_k s_k^t}{s_k^t y_k} + \left(1 + \frac{y_k^t y_k}{s_k^t y_k}\right) \frac{s_k s_k^t}{s_k^t y_k}$$

La matriz R^* tiene las siguientes propiedades

$p_{k+1} = -R^* g_{k+1}$ si la búsqueda lineal es exacta.

R^* es la actualización tipo BFGS de la matriz identidad usando los vectores s_k y y_k como en (6.8).

Lo que podemos escribir

$$R^* = U_{BFGS}(I, s_k, y_k)$$

Entonces es posible ver el método de Gradiente Conjugado como un método Quasi–Newton de BFGS en el que en cada iteración se aproxima al inverso de la matriz Hessiana por $R^* = U_{BFGS}(I, s_k, y_k)$. Esto introduce un cambio significativo en la formulación del método de Gradiente Conjugado ya que aunque la búsqueda lineal sea inexacta, lo que único que requerimos es garantizar que $s_k^t y_k > 0$ para que la dirección encontrada sea de descenso, y esto se obtiene con facilidad si el algoritmo de búsqueda lineal cumple las condiciones de Wolfe–Powell.

Una de las principales ventajas de ver al Gradiente Conjugado como un Quasi–Newton es que la búsqueda lineal puede iniciar siempre la búsqueda *con* $\alpha = 1$ y conforme nos acerquemos al mínimo este valor de prueba será suficiente para avanzar en el algoritmo.

7.2.2 Extensión del método de Gradiente Conjugado de Beale usando el enfoque Quasi–Newton

Las ideas de la sección anterior pueden extenderse al método de Gradiente Conjugado de Beale, visto en 3.4, el cual utiliza recomienzos cada t iteraciones, y donde se define

$$p_{k+1} = -g_{k+1} + \beta_{k+1} p_k + \gamma_k p_t$$

con

$$\beta_{k+1} = \frac{g_{k+1}^t y_k}{p_k^t y_k}, \qquad \gamma_k = \frac{g_{k+1}^t y_k}{p_t^t y_t}$$

de donde tenemos que

$$\begin{aligned} \gamma_k p_t &= \frac{g_{k+1}^t y_k}{p_t^t y_t} p_t \\ &= \frac{p_t y_t^t g_{k+1}}{p_t^t y_t} \\ &= \frac{s_t y_t^t}{s_t^t y_t} g_{k+1} \end{aligned}$$

lo cual implica que

$$\begin{aligned} p_{k+1} &= -g_{k+1} + \beta_{k+1} p_k + \frac{s_t t_t^t}{s_t^t y_t} g_{k+1} \\ &= -\left(I - \frac{s_t y_t^t}{s_t^t y_t} \right) g_{k+1} + \beta_{k+1} p_k \\ &= -H_t g_{k+1} + \beta_{k+1} p_k \end{aligned}$$

La idea es entonces ver a H_t como un precondicionador. Si tratamos de ver el proceso relacionado con un método Quasi–Newton, aplicando el proceso de la sección anterior obtenemos que

$$H_t^* = U_{BFGS}(I, s_t, y_t)$$

con esta forma de H_t^*, tenemos que

$$g_{k+1}^t H_t^* y_k = g_{k+1}^t \left(\left(I - \frac{s_t y_t^t + y_t s_t^t}{s_t^t y_t} \right) + \left(\frac{y_t^t y_t}{s_t^t y_t} \right) \frac{s_t s_t^t}{s_t^t y_t} \right) y_k$$

Como $y_k = g_{k+1} - g_k$ y para funciones cuadráticas con búsqueda lineal exacta se satisface que el gradiente es ortogonal a todas las direcciones anteriores y los gradientes son mutuamente ortogonales, entonces tenemos que $s_t^t y_k = 0$ y $g_t^t y_k = 0$, lo cual implica que

$$g_{k+1}^t H_t^* y_k = g_{k+1}^t y_k$$

entonces

$$\beta_{k+1} = \frac{g_{k+1}^t y_k}{p_k^t y_k} = \frac{g_{k+1}^t H_t^* y_k}{p_k^t y_k}$$

con esta β_{k+1} y recordando que $p_{k+1} = -H_t^* g_{k+1} + \beta_{k+1} p_k$ este método se puede ver como un Gradiente Conjugado Precondicionado.

Por lo tanto, podemos escribir

$$p_{k+1} = -\left(H_t^* - \frac{s_k y_k^t H_t^*}{s_k^t y_k} \right) g_{k+1}$$

repitiendo una vez mas el proceso anterior, tenemos

$$H_{k+1} = U_{BFGS}(H_t^*, s_k, y_k)$$

de esta manera se puede enfocar el método de Beale como un Quasi–Newton donde

$$p_{t+1} = -H_t^* g_{t+1}, \qquad H_t^* = U_{BFGS}(I, s_t, y_t)$$

$$p_{k+1} = -H_{k+1}^* g_{k+1}, \qquad H_{k+1}^* = U_{BFGS}(H_t, s_k, y_k) \text{ para } k > t$$

Con estos resultados Shanno [48] realiza una implementación de la extensión del método de Beale, utiliza el criterio de Powell para el recomienzo, esto es, recomienza si

$$|g_{k+1}^t g_k| \geq 0.2 \|g_k\|^2$$

Una experimentación computacional intensiva indica que en cada recomienzo el método pierde un poco su efectividad, por lo que prueba con otros miembros de la familia

de Broyden (6.6) que en el paso inicial proporcionen una mayor efectividad. Shanno notó que en la primera iteración de recomienzo era mejor usar la actualización

$$\hat{H}_t = U_{BFGS}(\gamma_t I, s_t, y_t)$$

donde

$$\gamma_t = \frac{s_t^t y_t}{y_t^t y_t}$$

La conveniencia de este escalamiento está todavía abierta a discusión y se usa en base a la experiencia computacional. El algoritmo de Shanno es el siguiente.

Algoritmo de Gradiente Conjugado de Shanno

1. Escoger $x_0 \in \mathbb{R}^n$, $\epsilon > 0$
2. $\alpha_0 = \dfrac{1}{\|g_0\|}$, $p_0 = -g_0$, $k = 0$
3. Si $\|g_k\| < \epsilon$ terminar, si no hacer $k = k + 1$, ir al paso 4.
4. Búsqueda lineal, encontrar
$$\alpha_k = \arg\min_{\alpha \geq 0} f(x_k + \alpha p_k)$$
5. Hacer $x_{k+1} = x_k + \alpha_k p_k$
6. Calcular la dirección p_{k+1}
   Si $k = 0 \bmod n$ o $|g_{k+1}^t g_k| \geq 0.2\|g_k\|^2$ hacer
   $k \rightarrow t$; $p_{t+1} = -\hat{H} g_{t+1}$; $\alpha_{t+1} = 1$
   con $\hat{H} = U_{BFGS}(\gamma_t I, s_t, y_t)$; $k = k + 1$
   ir al paso 3
   Si no
   $p_{k+1} = -U_{BFGS}(\hat{H}, s_k, y_k) g_{k+1} - H_{k+1}^* g_{k+1}$
   $\alpha_{k+1} = \alpha_k \dfrac{p_k^t g_k}{p_{k+1}^t g_{k+1}}$; $k = k + 1$
   ir al paso 3
7. Fin

Veamos que no es necesario el almacenamiento de ninguna de las dos matrices anteriormente indicadas.

La iteración para la dirección es

$$p_{k+1} = -H^*_{k+1} g_{k+1} = -U_{BFGS}(\hat{H}_t, s_k, y_k) g_{k+1}$$

donde

$$H^*_{k+1} = \hat{H}_t - \frac{s_k y_k^t \hat{H}_t + \hat{H}_t y_k s_k^t}{s_k^t y_k} + \left(1 + \frac{y_k^t \hat{H}_t y_k}{s_k^t y_k}\right) \frac{s_k s_k^t}{s_k^t y_k}$$

y

$$\hat{H}_t = \gamma_t \left(I - \frac{y_t s_t^t + s_t^t y_t}{s_t^t y_t} + \frac{y_t^t y_t}{s_t^t y_t} \frac{s_t s_t^t}{s_t^t y_t}\right) \frac{s_t s_t^t}{s_t^t y_t}$$

con

$$\gamma_t = \frac{s_t^t y_t}{y_t^t y_t}$$

Cuando se está en una iteración de recomienzo, ésta se obtiene como

$$p_{t+1} = -\hat{H}_t g_{t+1}$$

para lo cual solo necesito tener almacenados a $y_k, y_t, s_t, s_k, g_{k+1}$.

Entonces no se necesita del almacenamiento de ninguna matriz solo de los siete vectores: $y_k, s_k, y_t, s_t, g_{k+1}, x_{k+1}$ y x_k.

7.2.3 Método de Gradiente Conjugado con almacenamiento variable

Cuando se aplica un método de Gradiente Conjugado con precondicionador $H_0 > 0$ a una función cuadrática, $\phi(x) = (1/2)x^t Q x$ con $Q > 0$, se tienen las siguientes propiedades:

i) El método converge en n pasos a lo sumo.

ii) Los vectores de búsqueda p_{k+1} son Q–conjugados.

iii) $g_i^t H_0 g_j = 0$ si $i \neq j$.

iv) Las direcciones p_k^{GC} (generadas por Gradiente Conjugado precondicionado con H_0 fijo) y p^ϵ (dirección generada por el método Quasi–Newton usando un miembro de la familia de Broyden con parámetro ϵ) son linealmente dependientes.

v) A menos que el método alcance la solución antes de n pasos, se tiene que

$$H_{n+1}^{\varepsilon} = Q^{-1}$$

En el caso que el elemento de la familia de Broyden sea la actualización tipo BFGS se tiene que no solo la dirección generada coincide con la del Gradiente Conjugado Precondicionado, sino que también coinciden en norma, esto es, son iguales, como lo establece el siguiente resultado debido a Nazareth [37].

Lema. *Cuando los algoritmos de Gradiente Conjugado Precondicionado y BFGS se aplican a una función cuadrática $\phi(x) = x^t Q x$, $Q > 0$, usando el mismo punto inicial y la misma H_0, entonces*

$$p_k^{GC} = p_k^{BFGS}$$

Para el caso de funciones no cuadráticas, Powell [46] demostró el siguiente resultado.

Teorema. *Sea el método Quasi–Newton que usa las matrices de la familia de Broyden aplicado a una función diferenciable $f(x)$ y supongamos que las búsquedas lineales son exactas, seleccionándose los α_k por el mismo método. Sea $x_1, x_2, \ldots, x_k$ la sucesión de iteraciones y $H_1^{\varepsilon}, \ldots, H_{k+1}^{\varepsilon}$ la sucesión de matrices desarrolladas antes de la k–ésima iteración, y supongamos que ningún vector de búsqueda p_k^{ε} se anula. Entonces si en la iteración k se usa la elección de ε correspondiente al método de BFGS, la matriz H_k^{BFGS} obtenida es independiente de los parámetros ε usados durante las iteraciones anteriores.*

Usando este teorema y poniendo $\varepsilon = 1$ en la fórmula para generar las matrices de Broyden y con búsquedas lineales exactas, se tiene:

$$\begin{aligned} p_{k+1}^{BFGS} &= -H_{k+1}^{BFGS} g_{k+1} \\ &= -H_k^{BFGS} g_{k+1} - \left(1 + \frac{y_k^t H_k^{BFGS} y_k}{s_k^t y_k}\right) \frac{s_k s_k^t}{s_k^t y_k} g_{k+1} + \\ &\quad \frac{1}{s_k^t y_k} \left(s_k y_k^t H_k^{BFGS} g_{k+1} + H_k^{BFGS} y_k s_k^t g_{k+1}\right) \end{aligned}$$

como la búsqueda lineal es exacta, se tiene que $s_k^t g_{k+1} = 0$ y tenemos que

$$p_{k+1}^{BFGS} = -H_k^{BFGS} g_{k+1} + \frac{s_k y_k^t H_k^{BFGS}}{s_k^t y_k} g_{k+1}$$

$$\Longrightarrow p_{k+1}^{BFGS} = -H_k^{BFGS} g_{k+1} + \frac{y_k^t H_k^{BFGS} g_{k+1}}{s_k^t y_k} s_k$$

$$\Longrightarrow p_{k+1}^{BFGS} = -H_k^{BFGS} g_{k+1} + \frac{y_k^t H_k^{BFGS} g_{k+1}}{y_k^t p_k^{BFGS}} p_k^{BFGS}$$

Si ahora suponemos que en los pasos anteriores al $k+1$ se había aplicado la matriz H_i^ε, $i = 0, \ldots, k$ con $\varepsilon \neq 1$, haciendo uso del teorema de Powell, se puede escribir la iteración como

$$p_0^\varepsilon = -H_0 g_0; \qquad k = 0$$

$$x_{k+1} = x_k + \alpha_k p_k^\varepsilon$$

$$p_{k+1} = -H_k^\varepsilon g_{k+1} + \frac{y_k^t H_k^\varepsilon g_{k+1}}{y_k^t p_k^\varepsilon} p_k^\varepsilon$$

para $k+2$ hasta la convergencia

$$p_{k+2}^{BFGS} = -H_{k+1}^{BFGS} g_{k+1} + \frac{y_k^t H_{k+1}^{BFGS} g_{k+2}}{y_k^t p_{k+1}^{BFGS}} p_{k+1}^\varepsilon$$

Es decir, estamos viendo al algoritmo de BFGS como un algoritmo de gradiente conjugado precondicionado, para el que la métrica (dada por la matriz que sirve de precondicionador) en lugar de estar fija se actualiza en cada paso con un miembro cualquiera de la familia de Broyden.

Nazareth [37] propuso interpretar el resultado anterior como un esquema para algoritmos de gradiente conjugado para problemas de gran escala. Los algoritmos de gradiente conjugado como ya hemos mencionado, requieren poca cantidad de memoria para operar, mientras que los de Quasi–Newton necesitan gran cantidad debido a que hay que almacenar la matriz hessiana. En la práctica el método de Gradiente Conjugado necesita mas iteraciones para converger comparado con los Quasi–Newton. Esto ha sido explicado por el hecho de que en el caso de gradiente conjugado, se tiene menos información acerca del comportamiento de la función (no se dispone de la matriz hessiana). Otra forma de explicarlo, es que en cada paso de gradiente conjugado se actualiza la matriz identidad, por lo que no guarda información de pasos anteriores.

Dado un problema de dimensión n muy grande, es posible que no se cuente con las $n(n+1)/2$ localizaciones de memoria, necesarias para almacenar la aproximación a la matriz hessiana que se necesitan para un método Quasi–Newton, pero posiblemente haya algo mas que $4n$ localizaciones que requiere el método de gradiente conjugado. De lo que se trata, entonces, es de aprovechar la memoria disponible, usando un método que, mientras tenga espacio, usa las direcciones de BFGS, y en

caso contrario, continúe con un método de gradiente conjugado precondicionado. Por supuesto, la matriz no se almacenará sino que se guardan los vectores que hacen falta para formarla, y aunque esto requiere mas trabajo de cálculo, ahorra la memoria a usar, que es el objetivo de este caso.

El método se ve desde el inicio por uno de Gradiente Conjugado Precondicionado, con precondicionador variable. El algoritmo propuesto por Nazareth es el siguiente:

Algoritmo.

Paso 1 Escoger $x_0 \in \mathbb{R}^n$, H_0 simétrica y positiva definida.

Paso 2 $p_0 = -H_0 g_0$, $k = 0$

Paso 3 $x_{k+1} = x_k + \alpha_k p_k$, con $\alpha_k = \arg\min_{\alpha \geq 0} f(x_k + \alpha p_k)$

Paso 4 Analizar criterio de parada.

Paso 5 Si ya se agotó la memoria disponible usar una de las siguientes opciones

Paso 5.1 poner $H_{k+1} \leftarrow H_0$ e ir al paso 5.2, o poner $H_{k+1} \leftarrow H_k$ fija e ir al paso 6.

Paso 5.2 Si aún queda memoria disponible, entonces actualizar la matriz y elegir cual miembro de la familia de Broyden se va a usar y cuan frecuentemente se va a hacer la actualización; si actualizar siempre que sea posible o cada j iteraciones donde j es una fracción de n determinada por la cantidad de memoria disponible. Note que solo se almacenan los vectores y escalares para encontrar la matriz.

Paso 6 Si no se satisfacen los criterios de recomienzo, ir al paso 7.
En caso contrario, úsese la opción de recomienzo deseada (este recomienzo está estrechamente ligado a la elección para cuando se agote la memoria disponible).

Paso 7 $p_{k+1} = -H_k g_{k+1} + \dfrac{y_k^t H_k g_{k+1}}{y_k^t p_k} p_k$ ir al paso 3

Se tienen algunas observaciones de este algoritmo, primero, en el paso 5.1 se podría calcular la diagonal de H_k y volvera poner en uno el contador de vectores de memoria utilizados (el que ocupará la diagonal) y entonces continuar en el paso 5.2, teniendo otra vez que generar las matrices por la fórmula de Broyden, ocupando los lugares de los vectores anteriores.

Segunda observación: el número de vectores posibles a tener en memoria es variable. Si la cantidad de memoria es la mínima, entonces el paso 5.2 nunca se ejecuta y el método es el estándar de gradiente conjugado precondicionado.

Hay varias posibilidades para la actualización de las matrices cuando se ha agotado la memoria, en particular en la siguiente sección trabajamos una de ellas.

7.3 Métodos Quasi–Newton para problemas de gran escala

Los métodos Quasi–Newton descritos en el capítulo 6 no se aplican directamente a problemas de optimización grandes, ya que los requerimientos de almacenamiento y cómputo son excesivos. Las aproximaciones a la matriz hessiana o a su inversa son usualmente densas y aún si las matrices se almacenan en forma descompuesta (por ejemplo, almacenando los vectores s_k y y_k para todas las iteraciones $k = 0, 1, \ldots$) los requerimientos de memoria crecen en cada iteración y eventualmente sobrepasarán los recursos disponibles.

Sin embargo, hay varias formas de extender los métodos Quasi–Newton para problemas grandes. La primera aproximación, los métodos Quasi–Newton de memoria limitada, definen una pequeña modificación a las técnicas descritas en el capítulo 6 para obtener aproximaciones a la matriz hessiana que se pueden almacenar de manera compacta en unos cuantos vectores de longitud n, donde n es el número de incógnitas del problema. Estos métodos son bastante robustos, baratos y de fácil implementación, pero su convergencia no es muy rápida. Otra aproximación consiste en definir fórmulas Quasi–Newton para actualizar aproximaciones a la hessiana que preserve patrones sparse, aunque estas aproximaciones no se ha demostrado sean tan efectivas como las anteriores. Nos efocaremos en los métodos Quasi–Newton de memoria limitada.

7.3.1 Memoria Limitada BFGS

Los métodos Quasi–Newton de Memoria Limitada son útiles para resolver problemas grandes donde la matriz hessiana no puede ser calculada a un costo razonable, o son densas. Estos métodos mantienen aproximaciones simples y compactas de las matrices hessianas: en lugar de almacenar las aproximaciones completas de $n \times n$ se guardan solo una cuantos vectores de longitud n que representan las aproximaciones implícitamente. Aún con los modestos requerimientos de almacenamiento estos métodos frecuentemente tienen una aceptable razón de convergencia, casi siempre

lineal. Se han propuesto varios métodos de memoria limitada, nos enfocaremos principalmente en el algoritmo conocido como L-BFGS, introducido por Nocedal [38], y que está basado en la fórmula de actualización BFGS. La idea básica del método es usar información solo de las mas recientes iteraciones para construir la aproximación a la hessiana. La información de la curvatura de iteraciones no recientes, las cuales no aportan información relevante del comportamiento de la hessiana en la iteración actual, son eliminadas para ahorrar espacio en memoria.

Para empezar la descripción del método L-BFGS, recordaremos el método BFGS estándar, el cual fue descrito con detalle en el capítulo 6. Cada paso de este método tiene la forma

$$x_{k+1} = x_k - \alpha_k H_k g_k, \qquad k = 0, 1, 2, \ldots$$

donde α_k es la longitud del paso y H_k se actualiza en cada iteración por la fórmula

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)s_k^t + s_k(s_k - H_k y_k)^t}{s_k^t y_k} - \frac{y_k^t(s_k - H_k y_k)s_k s_k^t}{(s_k^t y_k)^2}$$

Esta expresión puede verse de forma compacta si escribimos

$$\begin{aligned} H_{k+1} &= H_k + \frac{(s_k - H_k y_k)s_k^t + s_k(s_k - H_k y_k)^t}{s_k^t y_k} + \frac{y_k^t H_k y_k s_k s_k^t - (y_k^t s_k)s_k s_k^t}{(s_k^t y_k)^2} \\ &= H_k + \frac{(s_k - H_k y_k)s_k^t - s_k y_k^t H_k}{s_k^t y_k} + \frac{s_k(y_k^t H_k y_k)s_k^t}{(s_k^t y_k)^2} \end{aligned}$$

Definiendo

$$\rho_k = \frac{1}{s_k^t y_k}$$

entonces

$$\begin{aligned} H_{k+1} &= H_k - \rho_k H_k y_k s_k^t - \rho_k s_k y_k^t H_k + \rho_k^2 s_k(y_k^t H_k y_k)s_k^t + \rho_k s_k s_k^t \\ &= (H_k - \rho_k s_k y_k^t H_k)(I - \rho_k y_k s_k^t) + \rho_k s_k s_k^t \\ &= (I - \rho_k y_k s_k^t)^t H_k (I - \rho_k y_k s_k^t) + \rho_k s_k s_k^t \\ &= V_k^t H_k V_k + \rho_k s_k s_k^t \end{aligned}$$

donde

$$V_k = I - \rho_k y_k s_k^t$$

entonces tenemos una representación mas compacta de las matrices BFGS

$$H_{k+1} = V_k^t H_k V_k + \rho_k s_k s_k^t \tag{7.21}$$

donde

$$\rho_k = \frac{1}{y_k^t s_k}, \qquad V_k = I - \rho_k y_k s_k^t \tag{7.22}$$

y

$$s_k = x_{k+1} - x_k, \qquad y_k = g_{k+1} - g_k \tag{7.23}$$

decimos que la matriz H_{k+1} se obtiene actualizando H_k usando el par $\{s_k, y_k\}$.

La aproximación H_k generalmente va a ser densa, tal que el costo de almacenamiento y manipulación es bastante alto cuando el número de variables es grande, usualmente H_{k+1} se reescribe sobre H_k y por ser matrices simétricas necesitamos $n(n+1)/2$ lugares en memoria para almacenarla. Una forma de superar este problema es no formar y almacenar la matriz H_k de $n \times n$ explícitamente, en su lugar almacenamos una versión modificada de H_k implícitamente, almacenando un cierto número, digamos m de los pares de vectores $\{s_i, y_i\}$ que se utilizan en la formulación (7.21)–(7.23). Veamos como sería el proceso. Supongamos que tenemos espacio para guardar $m = 2$ pares de vectores $\{s_i, y_i\}$ y sea H_0 una matriz positiva definida dada, entonces si

$$\rho_i = \frac{1}{y_i^t s_i}, \qquad V_i = (I - \rho_i y_i s_i^t)$$

se tiene que

$$H_1 = V_0^t H_0 V_0 + \rho_0 s_0 s_0^t$$

de donde

$$\begin{aligned} H_2 &= V_1^t H_1 V_1 + \rho_1 s_1 s_1^t \\ &= V_1 (V_0^t H_0 V_0 + \rho_0 s_0 s_0^t) V_1 + \rho_1 s_1 s_1^t \\ &= V_1^t V_0^t H_0 V_0 V_1 + \rho_0 V_1^t s_0 s_0^t V_1 + \rho_1 s_1 s_1^t \end{aligned}$$

Ahora, para calcular H_3 necesitamos $\{s_2, y_2\}$, pero no podemos almacenar tres pares de vectores, ya que $m = 2$, entonces se elimina la información mas antigua, manteniendo asi la información mas reciente, esto es, calculamos $\{s_2, y_2\}$ y se elimina de memoria $\{s_0, y_0\}$, con estos datos tenemos la aproximación

$$H_2 = V_1^t H_0 V_1 + \rho_1 s_1 s_1^t$$

y obtenemos

$$H_3 = V_2^t V_1^t H_0 V_1 V_2 + \rho_1 V_2^t s_1 s_1^t V_2 + \rho_2 s_2 s_2^t$$

siguiendo el mismo proceso, tenemos

$$\begin{aligned} H_4 &= V_3^t V_2^t H_0 V_2 V_3 + \rho_2 V_3^t s_2 s_2^t V_3 + \rho_3 s_3 s_3^t \\ &\vdots \end{aligned}$$

En general, para $k \leq m$ tenemos la aproximación estándar BFGS

$$\begin{aligned} H_k &= (V_{k-1}^t \cdots V_0^t) H_0 (V_0 \cdots V_{k-1}) \\ &\quad + \rho_0 (V_{k-1}^t \cdots V_1^t) s_0 s_0^t (V_1 \cdots V_{k-1}) \\ &\quad + \rho_1 (V_{k-1}^t \cdots V_2^t) s_1 s_1^t (V_2 \cdots V_{k-1}) \\ &\quad + \cdots \\ &\quad + \rho_{k-1} s_{k-1} s_{k-1}^t \end{aligned} \tag{7.24}$$

Y para $k > m$ tenemos la actualización especial BFGS

$$\begin{aligned} H_k &= (V_{k-1}^t \cdots V_{k-m}^t) H_0 (V_{k-m} \cdots V_{k-1}) \\ &\quad + \rho_{k-m} (V_{k-1}^t \cdots V_{k-m+1}^t) s_{k-m} s_{k-m}^t (V_{k-m+1} \cdots V_{k-1}) \\ &\quad + \rho_{k-m+1} (V_{k-1}^t \cdots V_{k-m+2}^t) s_{k-m+1} s_{k-m+1}^t (V_{k-m+2} \cdots V_{k-1}) \\ &\quad + \cdots \\ &\quad + \rho_{k-1} s_{k-1} s_{k-1}^t \end{aligned} \tag{7.25}$$

Las matrices (7.24) y (7.25) Nocedal [38] las denomina matrices especiales BFGS. Esta aproximación es adecuada para problemas grandes ya que la experiencia práctica ha mostrado que valores de m entre 3 y 20 producen resultados satisfactorios.

Una de las propiedades importantes de estas matrices es que si H_0 es positiva definida y $s_i^t y_i > 0$ para toda i, entonces las matrices (7.24) y (7.25) son positivas definidas.

Una forma de garantizar que $y_i^t s_i > 0$ para toda i es que la búsqueda lineal satisfaga las condiciones de Wolfe–Powell, con lo cual se garantiza que la dirección encontrada es siempre una dirección de descenso utilizando estas matrices especiales.

Con esta definición de H_k el algoritmo L–BFGS puede establecerse como sigue.

Algoritmo L-BFGS

Escoger un punto inicial x_0 y un entero $m > 0$
for $k = 0, 1, \ldots$
    Escoger $H_0^{(k)}$
    Calcular $p_k = -H_k g_k$
    Calcular $x_{k+1} = x_k + \alpha_k p_k$, donde α_k se escoge tal que satisface
      las condiciones de Wolfe–Powell
    Si $k > m$
      Eliminar el par de vectores $\{s_{k-m}, y_{k-m}\}$ de memoria
    end
    Calcular $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$
    $k = k + 1$
end

Durante las primeras $m - 1$ iteraciones el algoritmo L-BFGS es equivalente al algoritmo BFGS del capítulo 6 si la matriz inicial $H_0^{(k)}$ es la misma en cada iteración, nótese que a diferencia de BFGS estándar aqui se permite que la aproximación a la matriz inicial, H_0, varíe de iteración a iteración. El algoritmo L-BFGS no almacena H_k explícitamente y de aqui que utilice un procediemiento numérico distinto para calcular p_k, sin embargo, las diferencias de las primeras m iteraciones generadas por los dos algoritmos pueden deberse a errores de redondeo. De hecho, se puede reimplementar el método BFGS estándar asignando a m un valor grande en el algoritmo L-BFGS (mayor que el número de iteraciones necesarias para encontar la solución), pero cuando m se aproxime a n (específicamente $m > n/2$) esta aproximación va a ser mas costosa en términos de tiempo de cómputo y almacenamiento que la aproximación del algoritmo BFGS estándar.

El algoritmo L–BFGS no almacena H_k explícitamente, solo se guardan los pares de vectores $\{s_i, y_i\}$, $i = k - m + 1, \ldots, k$, con esta información es posible obtener un algoritmo eficiente para calcular el producto $H_k g_k$ y obtener la dirección de búsqueda p_k. Nocedal [38] propone un algoritmo de dos ciclos para el cálculo de p_k.

Recursión de dos ciclos para L-BFGS

$q = g_k$
for $i = k-1, k-2, \ldots, k-m$
    $\alpha_i = \rho_i s_i^t q$
    $q = q - \alpha_i y_i$
end
$r = H_0^{(k)} q$
for $i = k-m, k-m+1, \ldots, k-1$
    $\beta = \rho_i y_i^t r$
    $r = r + s_i(\alpha_i - \beta)$
end

De aqui resulta que $r = H_k g_k$.

Aparte de la multiplicación $H_0^{(k)} q$, el esquema de recusión de dos ciclos requiere $4mn$ multiplicaciones, si $H_0^{(k)}$ es diagonal, entonces se necesitan solo n multiplicaciones mas. Además de ser barato, esta recursión tiene la ventaja de que la multiplicación por la matriz inicial $H_0^{(k)}$ se aisla del resto de los cálculos, permitiendo que esta matriz pueda escogerse libremente y que pueda variar entre iteraciones. Entonces podemos hacer una elección implícita de $H_0^{(k)}$ definiendo alguna aproximación inicial $B_0^{(k)}$ a la matriz hessiana, no a su inversa, y obtener r resolviendo el sistema $B_0^{(k)} r = q$.

Un método para escoger $H_0^{(k)}$ que ha probado su efectividad en la práctica es considerar $H_0^{(k)} = \gamma_k I$, donde

$$\gamma_k = \frac{s_{k-1}^t y_{k-1}}{y_{k-1}^t y_{k-1}} \tag{7.26}$$

γ_k es el factor de escalamiento que intenta estimar el tamaño de la matriz hessiana real sobre la dirección de búsqueda mas reciente. Esta elección parece que permite asegurar que la dirección de búsqueda p_k esté bien escalada, de manera que existen parámetros aceptables de la búsqueda lineal cercanos a uno, y tal que el número de evaluaciones de la función necesarias en la búsqueda lineal no sea muy grande en general.

Hasta ahora hemos supuesto que el número m de correcciones almacenadas es constante, pero el proceso de recursión de puede adaptarse para permitir que m varíe de una iteración a la siguiente.

La estrategia de mantener los m mas recientes pares $\{s_i, y_i\}$ trabaja bien en la práctica, y hasta el momento no se ha probado que otra estrategia sea mejor. Otro criterio que se puede considerar es por ejemplo, aquél en donde se mantenga el conjunto de correcciones para las cuales la matriz formada por las s_i tenga el mejor condicionamiento, esto tiende a evitar conjuntos de vectores en los cuales algunas s_i's

son colineales.

En investigaciones recientes, Byrd, Nocedal y Schnabel, [11], proponen una nueva forma de representar las matrices BFGS que son principalmente útiles al resolver problemas con restricciones, por cuestiones de tiempo y espacio no mencionaremos estas ideas.

7.4 Métodos de Newton truncado

El método de Newton, como se ha mencionado en capítulos anteriores, es la base de muchos de los métodos de optimización, esto es, métodos que aproximan f localmente por una función cuadrática Ψ. Si el número de variables n, es grande, entonces el método de Newton puede ser problemático ya que requiere el cálculo y almacenamiento de la matriz Hesiana de segundas derivadas parciales y para funciones de gran escala esto representa grandes costos.

En esta sección presentamos una clase de métodos adecuados para problemas de gran escala llamados métodos de Newton Truncado, éstos métodos están basados en minimizar aproximadamente la función cuadrática Ψ usando un esquema iterativo tal como el algoritmo de Gradiente Conjugado lineal. Un algoritmo de Newton Truncado se compone de dos subalgoritmos: un algoritmo externo no lineal que controla la minimización y un algoritmo interno lineal para minimizar aproximadamente Ψ.

Recordemos que la minimización de la cuadrática Ψ, que determina el paso básico de Newton p_k^N, se obtiene resolviendo el sistema lineal simétrico de $n \times n$

$$\nabla^2 f(x_k) p_k = -\nabla f(x_k) \tag{7.27}$$

conocido como las ecuaciones de Newton. En algunas ocasiones no se trabaja con la matriz Hessiana directamente, sino con una buena aproximación a ésta, usualmente mas económica, se considera

$$B_k \approx \nabla^2 f(x_k)$$

y entonces las ecuaciones de Newton se transforman en

$$B_k p_k = -\nabla f(x_k) \tag{7.28}$$

7.4.1 Descripción básica del método

Como el método de Newton está basado en la expansión en serie de Taylor cerca de la solución del problema de minimización, no hay garantía de que la dirección

de búsqueda calculada sea crucial lejos de x^*. De hecho, al inicio del proceso una aproximación razonable a la dirección de Newton puede ser casi tan efectiva como la dirección de Newton misma. Gradualmente, conforme nos acerquemos a la solución, la dirección de Newton tiene mas significado.

Esto sugiere utilizar un método iterativo para resolver las ecuaciones de Newton. Además, debe ser un método iterativo con tolerancia variable tal que lejos de la solución (7.28) no se resuelva con mucha precisión. Solo cuando nos acerquemos a la solución consideraremos la solución de (7.28) con precisión. También conforme nos acerquemos a la solución x^*, el sistema de ecuaciones a resolver será mas similar, consecuentemente es posible que una aproximación cercana a la solución del sistema lineal pueda encontrarse sin mucho esfuerzo utilizando información anterior.

Como el sistema lineal no se resuelve exactamente, los pasos que resultan al resolver las ecuaciones de Newton aproximadamente, se conocen como pasos de Newton inexactos.

Se han propuesto varios métodos para la iteración interna, el mas recomendable para esta aplicación es el algoritmo de gradiente conjugado lineal, discutido ya anteriormente, debido a su principal atractivo: convergencia en a lo mas n pasos en aritmética exacta y no requiere el almacenamiento explícito de la matriz del sistema, solo es necesaria una rutina que realice el producto matriz–vector. Un requerimiento de este método es que la matriz de coeficientes debe ser positiva definida, se sabe que la matriz Hessiana tiene garantía de ser positiva semi–definida en la solución del problema de minimización, pero puede ser indefinida en cualquier otro lugar; entonces, el método iterativo para resolver (7.28) debe ser capaz de detectar los sistemas indefinidos.

Enseguida presentamos el algoritmo del Método de Newton truncado.

Algoritmo: Newton Truncado

Iteración principal
  for $k = 0, 1, \ldots$
    Calcular $f(x_k)$, $\nabla f(x_k)$, $\nabla^2 f(x_k)$
    Iteración menor
      Calcular p_k tal que $B_k p_k = -\nabla f(x_k)$
        donde B_k es alguna aproximación definida positiva
        a la hessiana en x_k
    end
    $x_{k+1} = x_k + p_k$
  end
end

Este algoritmo, al igual que el método de Newton estándar, converge localmente y se pueden aplicar estrategias de globalización como búsqueda lineal y región de confianza para obtener convergencia global. Antes de ver el método con las estrategias de globalización veremos cómo terminar la iteración lineal para obtener una solución aproximada de las ecuaciones de Newton.

7.4.2 Terminación del algoritmo lineal

Debido a que las ventajas del método de Newton son principalmente locales, parece no haber justificación para emplear tanto esfuerzo en obtener una solución exacta de las ecuaciones de Newton cuando se está lejos de la solución del problema. Sería entonces mas conveniente resolver dichas ecuaciones usando un método iterativo que calcule una aproximación de su solución y que tenga pocos requerimientos de memoria, dado que deseamos resolver problemas de gran escala. Por lo tanto, tendremos una estrecha relación entre la cantidad de trabajo necesario para calcular la dirección de búsqueda y la precisión con que se resuelvan las ecuaciones de Newton.

Una medida natural e independiente de la escala para terminar la iteración de las ecuaciones de Newton, es el residual relativo

$$\frac{\|r_k\|}{\|\nabla f(x_k)\|} \tag{7.29}$$

donde si p_k es la dirección de descenso, entonces r_k está dado por

$$r_k = B_k p_k + \nabla f(x_k)$$

El método de Newton truncado se basa en "truncar" la iteración que resuelve el sistema lineal cuando la solución aproximada satisface

$$\frac{\|r_k\|}{\|\nabla f(x_k)\|} \leq \eta_k \tag{7.30}$$

donde $\{\eta_k\}$, que es lo que se conoce como *"forcing sequence"*, controla la exactitud de la solución p_k.

Con estos elementos el algoritmo de Gradiente conjugado lineal precondicionado que se utiliza para resolver aproximadamente las ecuaciones de Newton es el siguiente.

Algoritmo de gradiente conjugado lineal precondicionado

1. $p_0 = 0$, $r_0 = -\nabla f(x_0)$, $i = 0$

2. $Mz_i = r_i$, $i = i + 1$
   if $i = 1$
   $d_i = z_0$, determinar η
   else
   $\beta_i = r_{i-1}^t z_{i-1}/(r_{i-2}^t z_{i-2})$
   $d_i = z_{i-1} + \beta_i d_{i-1}$
   end

3. if $d_i^t B d_i \leq (eps)^{\frac{1}{2}} d_i^t d_i$
   (Curvatura negativa)
   if $i = 1$
   $p = d_1$, Salida(1)
   else
   $p = p_i$, Salida (2)
   end
   else
   ir a 4
   end

4. Cálculo del tamaño de paso $\alpha_i = (r_{i-1}^t z_{i-1})/(d_i^t B d_i)$

5. $p_i = p_{i-1} + \alpha_i d_i$
   $r_i = r_{i-1} - \alpha_i B d_i$

6. Criterio de parada
   if $\frac{\|r_i\|}{\|M^{-1}\nabla f(x_k)\|} \leq \eta$
   (Se satisface el criterio de truncamiento)
   $p = p_i$, Salida(3)
   else
   Ir a 2.
   end

En este algoritmo tenemos tres salidas diferentes.
Salida(1). El gradiente apunta en una dirección de curvatura negativa, es decir $\nabla f(x_k)^t B \nabla f(x_k) < \epsilon(\delta)$, en este caso la dirección que se toma es la de descenso más rápido, esto es, $p_i = d_0 = -\nabla f(x_0)$.
Salida(2). La iteración de gradiente conjugado encontró una dirección de curvatura negativa, esto es, $d_i^t B d_i < \epsilon(\delta)$, se termina la iteración tomando la dirección de búsqueda como el último p_i obtenido.
Salida(3). El algoritmo termina porque se satisface el criterio de truncamiento para la dirección de Newton.

Al implementar este algoritmo en la práctica tenemos que restringir, además, el número de iteraciones de gradiente conjugado lineal permitidas ya que no es costeable realizar demasiadas.

7.4.3 Newton truncado con búsqueda lineal

El método de Newton truncado puede implementarse utilizando la estrategia de búsqueda lineal, esto es, dada una dirección p_k calculada con el algoritmo anterior, la nueva iteración está dada por

$$x_{k+1} = x_k + \alpha_k p_k$$

con α_k obtenida con alguna estrategia de búsqueda lineal, que se escoge tal que satisfaga las condiciones de Wolfe–Powell para garantizar convergencia.

El siguiente resultado, tomado de [15], muestra que cerca del mínimo el algoritmo termina usando la dirección de Newton truncado de la Salida(3), donde $\alpha_k = 1$ y no hay necesidad de búsqueda lineal.

Teorema. *Sea $x_k \to x^*$ donde $B(x^*)$ es positiva definida, entonces, para $\epsilon > 0$ pequeño, existe k_0 tal que el criterio de* Salida(3) *se satisface y $\alpha_k = 1$ es aceptable para $k \geq k_0$. Por lo tanto, para $k \geq k_0$ el método de Newton truncado se reduce a*

for $k = k_0, \ldots$
  Encontrar p_k *tal que* $B_k p_k = -\nabla f(x_k) + r_k$
  $x_{k+1} = x_k + p_k$
end

7.4.4 Elección de η_k

La elección de η_k es importante, ya que afecta directamente la razón de convergencia del método. El siguiente resultado, tomado de [35], muestra que se puede alcanzar

convergencia global si $\{\eta_k\}$ no tiende a 1.

Teorema. *Supongamos que $\nabla f(x)$ es continuamente diferenciable en una vecindad del mínimo x^* y suponga que $\nabla^2 f(x^*)$ es positiva definida. Considere la iteración $x_{k+1} = x_k + p_k$ donde p_k satisface (7.30) y si $\eta_k < \eta < 1$, entonces si el punto inicial está suficientemente cerca a x^*, la sucesión $\{x_k\}$ converge linealmente a x^*, esto es, para toda k suficientemente grande*

$$\|x_{k+1} - x^*\| \leq c\|x_k - x^*\|$$

para alguna constante $0 < c < 1$.

Notemos que la condición de la "forcing sequence" $\{\eta_k\}$ no es muy restrictiva en el sentido de que si se debilita un poco producirá un algoritmo que no converge. Específicamente, si se permiten $\eta_k \geq 1$ el paso $p_k = 0$ satisface (7.30) en cada iteración, pero el método producirá $x_k = x_0$ para toda k y no habrá convergencia a la solución.

El siguiente resultado proporciona un método constructivo para el algoritmo de Newton truncado, de forma que tendremos un orden de convergencia 1 o 2 según se haya establecido.

Teorema. *Sea $x_k \to x^*$ donde $B(x^*)$ es positiva definida y si suponemos que B_k cumple la condición de Lispchitz en x^*, esto es que existen constantes $c \in (0, 1]$ y L tales para toda y en una vecindad de x^* se satisface que*

$$\|B_k(y) - B_k(x^*)\| \leq L\|y - x^*\|^c$$

entonces

1. *$x_k \to x^*$ superlinealmente si y sólo si*

$$\frac{\|r_k\|}{\|\nabla f(x_k)\|} \to 0$$

   cuando $k \to \infty$.

2. *$x_k \to x^*$ con orden $(1 + t)$ si y sólo si*

$$\lim_{k\to\infty} sup \frac{\|r_k\|}{\|\nabla f(x_k)\|^{1-t}} < c$$

   con $c > 0$.

Entonces si seleccionamos

$$\eta_k = \min\{1/k, \|\nabla f(x_k)\|^t\}$$

con $t \in [0, 1]$, tendremos que un método de Newton truncado con orden de convergencia $1+t$. La sucesión $\{\eta_k\}$ da como consecuencia un algoritmo adaptivo para resolver el problema de optimización. Cuando nos encontramos lejos de la solución $\|\nabla f(x_k)\|$ es grande y requeriremos poco trabajo para obtener una solución que satisfaga el criterio de Salida(3) en el algoritmo de Newton truncado, mientras que cuanto mas nos acerquemos a la solución tendremos que $\|\nabla f(x_k)\| \to 0$ lo que implica que $\eta_k \to 0$ y se obliga al algoritmo de gradiente conjugado a calcular una dirección p_k cada vez mas cercana a la de Newton, tanto en dirección como en magnitud.

Como estamos suponiendo problemas de gran escala, usualmente no tendremos recursos disponibles para almacenar la matriz hessiana, sin embargo recordemos que la iteración de gradiente conjugado lineal solo necesita el producto matriz–vector, que en este caso se puede obtener aproximando las diferencias

$$B_k d = \frac{1}{\gamma}(\nabla f(x_k + \gamma d) - \nabla f(x_k))$$

donde γ se escoge tal que

$$\gamma = \frac{(eps)^{1/2}}{\|d\|}$$

De esta forma no tendremos que almacenar la matriz hessiana, pero necesitaremos una evaluación extra del gradiente en cada iteración del algoritmo de gradiente conjugado.

7.4.5 Método de Newton truncado con región de confianza

La idea de la sección anterior de "truncar" las ecuaciones de Newton usando una solución aproximada por medio de la iteración de gradiente conjugado lineal, puede también aplicarse a una iteración principal en la que en lugar de realizar búsqueda lineal, trabajemos la estrategia de región de confianza [51].

Los métodos de región de confianza obtienen una dirección de búsqueda tal que sea la solución de las ecuaciones de Newton, solo cuando este vector se encuentre dentro de una bola de radio Δ donde se confía que el modelo cuadrático aproxima bien la función no lineal, de no ser este el caso, se toma una dirección que es una combinación lineal de la dirección de Newton y la de descenso más rápido, tal que su longitud sea igual al radio de la región de confianza. De esta manera, el problema a resolver en cada paso es

$$\min_{\|p\| \leq \Delta} \Psi(p) \tag{7.31}$$

La idea de los métodos de Newton truncado, como hemos mencionado, es resolver las ecuaciones de Newton con mas exactitud en la medida en que estemos mas cerca del óptimo, o lo que es lo mismo, en la medida en que la función se parezca mas a una cuadrática. La solución de (7.31) puede encontrarse utilizando un método de gradiente conjugado lineal modificado que controle el tamaño de los vectores de sus iteraciones para adptarse a la restricción, como se vió en la sección 4.6, específicamente se utiliza el algoritmo de Steihaug para resolver este problema.

Algoritmo de gradiente conjugado lineal precondicionado con estrategia de región de confianza

1. $p_0 = 0$, $r_0 = -\nabla f(x_k)$, $i = 0$

2. $Mz_i = r_i$, $i = i + 1$
   if $i = i$
   $d_i = z_0$
   determinar η
   else
   $\beta_i = \frac{r_{i-1}^t z_{i-1}}{(r_{i-2}^t z_{i-2})}$
   $d_i = z_{i-1} + \beta_i d_{i-1}$
   end

3. if $(d_i B d_i) \leq (macheps)^{1/2}(d_i^t d_i)$
   Calcular $\tau > 0$ tal que $\|p_i + \tau d_i\| = \Delta$
   $p = p_i + \tau d_i$
   Salida(1)
   else
   Ir a 4
   end

4. $\alpha_i = \frac{(r_{i-1}^t z_{i-1})}{(d_i^t B d_i)}$

5. Hacer $p_i = p_{i-1} + \alpha_i d_i$
   if $\|p_i\| > \Delta$
   Calcular $\tau > 0$ tal que $\|p_i + \tau d_i\| = \Delta$
   $p = p_i + \tau d_i$
   Salida(2)
   else

Ir a 6
end

6. $r_i = r_{i-1} - \alpha_i B d_i$

7. Criterio de parada
if $\frac{\|r_i\|}{\|M^{-1}\nabla f(x_k)\|} \leq \eta$
Se satisface criterio de truncamiento
$p = p_{i+1}$
Salida(3)
else
Ir a 2
end

El seleccionar el paso a la frontera en las salidas (1) y (2) se justifica por el siguiente teorema, demostrado en 4.6.

Teorema. *Sea p_j, $j = 0, 1, \ldots, i$, el conjunto de iteraciones generadas por el algoritmo anterior, entonces $\Psi(p_j)$ es estrictamente decreciente y*

$$\Psi(p) \leq \Psi(p_i)$$

Más ún, $\|p_j\|$ es estrictamente creciente para $j = 0, \ldots, i$ y

$$\|p\| \geq \|p_i\|$$

La iteración principal del método de Newton truncado con región de confianza puede escribirse de la siguiente forma.

Iteración principal de Newton truncado con región de confianza
Dado x_0 y $\Delta > 0$

1. Calcular p_k tal que $B_k p_k \approx -\nabla f(x_k)$ usando el método de Steihaug

2. $x_{k+1} = x_k + p_k$

3. Actualizar el radio de la región de confianza

Referencias

R. Dembo and T. Steihaug, (1983). Truncated Newton algorithms for large-scale optimization, *Math. Programming* vol. 26, pp. 190-212.

S. Eisenstat and H. Walker (1994). Globally convergent inexact Newton methods, *SIAM J. Optimization* vol. 4, pp. 393–442.

D. Liu and J. Nocedal (1989). On the limited memory BFGS method for large scale optimization, *Math. Programming* vol. 45, pp. 503–528.

S. Nash (1982). *Truncated-Newton Methods.* Ph.D. thesis, Stanford University.

J.Nocedal (1980). Updating quasi–Newton matrices with limited storage, *Mathematics of Computation*, vol. 35, pp.773–782.

J. Nocedal and J. Wright (1999). *Numerical Optimization*, Springer Series in Operations Research.

Capítulo 8

Funciones Parcialmente Separables

Cuando se trabajan funciones con un gran número de variables, debemos tratar de explotar cualquier estructura que presente la función para tener algoritmos mas económicos. Muchas de las funciones de gran escala que aparecen en la práctica poseen una propiedad conocida como separabilidad parcial y en los últimos años se han desarrollado métodos que explotan esta estructura en la función objetivo, en particular los métodos Quasi–Newton han sido de los mas efectivos. Veremos la definición de funciones parcialmente separables, sus propiedades y métodos de solución cuando al función objetivo tenga esta propiedad.

En un problema de optimización separable la función objetivo puede descomponerse en una suma de funciones simples que pueden optimizarse de forma independiente, si tenemos

$$f(x) = f_1(x_1) + f_2(x_2) + \cdots + f_n(x_n)$$

entonces podemos encontrar el óptimo de $f(x)$ minimizando cada función f_i, $i = 1, \ldots, n$, de manera independiente. En muchos casos, realizar n minimizaciones unidimensionales es mas económico que el realizar una minimización n–dimensional.

En muchos problemas de gran escala, la función objetivo $f : \mathbb{R}^n \to \mathbb{R}$ no es separable, pero puede escribirse como suma de funciones simples conocidas como *funciones elementales*. Para cada función elemental es posible definir una base ortogonal en $\mathbb{R}^n$ de tal forma que el valor de la función no varíe al movernos sobre estas direcciones base. Si esta propiedad se satisface para todas las funciones elementales decimos que f es *parcialmente separable*. Todas las funciones cuyas hessianas son *sparse* son parcialmente separables, pero también existe la separabilidad parcial en funciones cuyas hessianas no son *sparse*.

La forma mas simple de separabilidad parcial aparece cuando la función objetivo

puede escribirse como

$$f(x) = \sum_{i=1}^{ne} f_i(x)$$

donde cada función elemental $f_i(x)$ depende solo de unos cuantas componentes de la variable x, entonces los gradientes ∇f_i y las matrices hessianas $\nabla^2 f_i$ de cada función elemental típicamente contendrán solo unos cuantos elementos distintos de cero. El gradiente y la matriz hessiana de la función f están dados por

$$\nabla f = \sum_{i=1}^{ne} \nabla f_i(x), \qquad \nabla^2 f = \sum_{i=1}^{ne} \nabla^2 f_i(x)$$

entonces por ejemplo si se aplica un método tipo Quasi–Newton para resolver el problema de optimización es mas económico actualizar por el método Quasi–Newton cada matriz hessiana elemental $\nabla^2 f_i(x)$ en lugar de actualizar toda la matriz hessiana total.

8.1 Variables internas

Existen muchas aplicaciones en la práctica donde no es fácil identificar la descomposición en funciones elementales de una función parcialmente separable. El éxito del método de optimización que explota la seprabilidad parcial depende fuertemente de la descomposición de f. Ilustramos la descomposición de una función como suma de funciones elementales con el ejemplo del problema de la superficie de área mínima.

8.1.1 El problema de la superficie de área mínima

El problema clásico de la superficie de área mínima consiste en encontrar la superficie de área mínima que interpola una función continua dada definida en la frontera del cuadrado unitario, I. Esto es, se tiene que resolver un problema de la forma

$$\min\{f(v) : v \in K\}$$

donde $f : K \to \mathbb{R}$ es la funcional

$$f(v) = \int_I (1 + \|\nabla v(z)\|^2)^{\frac{1}{2}} dz$$

y el conjunto K se define por

$$K = \{v \in C^1(I) : v(z) = v_I(z) \text{ para } z \in \partial I\}$$

para alguna función $v_I : \partial I \rightarrow \mathbb{R}$. La función definida sobre la frontera define unívocamente la solución al problema de superficie mínima.

Para resolver este problema se puede discretizar el cuadrado unitario en m^2 cuadrados pequeños, esto es, dividimos cada lado del cuadrado unitario en m intervalos de igual longitud, resultando una malla de $(m+1)^2$ puntos. A cada punto le asociamos una variable que representa la altura de la superficie en ese punto. Para los puntos en la frontera del cuadradro unitario los valores de las variables están determinados por la función dada, entonces el problema es determinar los valores de las variables internas, tal que el área de la superficie total sea mínima.

Por ejemplo, si consideramos $m = 3$, la malla generada y la numeración usual de las variables es como se ilustra en la figura 8.1

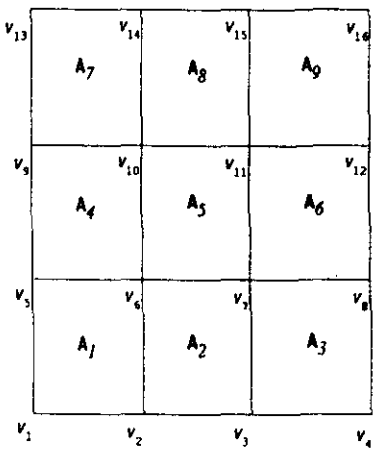


Figura 8.1: Discretización para el problema de superficie mínima

Un cuadrado pequeño de esta partición tiene la forma que se ilustra en la figura 8.2

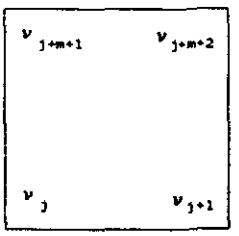


Figura 8.2: Numeración de variables en el problema de superficie mínima

Se puede obtener una aproximación al problema de superficie mínima por elemento finito minimizando f sobre el espacio de funciones lineales por pedazos v con valores v_j en z_j, donde $z_j \in \mathbb{R}^2$ son los vértices de la discretización de I. Los valores v_j se obtienen resolviendo el problema de minimización

$$\min \sum_{i=1}^{m^2} f_i(v) : v \in \mathbb{R}^n$$

donde las funciones f_i representan el área de la superficie sobre cada cuadrado pequeño de la discretización y se definen como

$$f_i(v) = \frac{1}{m^2}\left(1 + \frac{m^2}{2}[(v_j - v_{j+m+1})^2 + (v_{j+1} - v_{j+m})^2]\right)^{\frac{1}{2}} \tag{8.1}$$

donde j depende de i y de m.

Veamos que podemos decir de cada función elemental. Trabajaremos con el vector de incógnitas como x en lugar de v, esto es, cada función $f_i(x)$ como en (8.1). Consideremos el ejemplo de $m = 3$ y con $f_1(x)$, los demás casos son similares. Para el primer cuadrado A_1, tenemos que la función está definida por

$$f_1(x) = \frac{1}{m^2}\left(1 + \frac{m^2}{2}[(x_1 - x_6)^2 + (x_2 - x_5)^2]\right)^{\frac{1}{2}}$$

Notemos que la función elemental f_1 depende solo de cuatro componentes del vector x las cuales llamaremos variables elementales y que podemos acomodar en un vector $x_{[1]}$.

El gradiente de f_1 con respecto al vector completo x, contiene a lo mas cuatro elementos distintos de cero y tiene la forma

$$\nabla f_1(x) = \frac{1}{2m^2 f_1(x)} \begin{pmatrix} x_1 - x_6 \\ x_2 - x_5 \\ 0 \\ \vdots \\ -x_2 + x_5 \\ -x_1 + x_6 \\ \vdots \\ 0 \end{pmatrix}$$

Nótese que dos de estas componentes son las negativas de las otras dos. La matriz Hessiana de esta función elemental f_1 es *sparse* y tiene la forma

$$\begin{pmatrix} * & * & & * & * \\ * & * & & * & * \\ & & & & \\ & & & & \\ * & * & & * & * \\ * & * & & * & * \end{pmatrix}$$

que contiene a lo mas 16 elementos distintos de cero y se puede demostrar que algunas de las componentes difieren solo en su signo, solo hay tres magnitudes diferentes en los elementos no cero.

La primera forma para definir las variables elementales de f_1 es considerar las cuatro componentes de x que aparecen en la definición de f_1, pero como se cuenta con información duplicada en la derivada, se puede obtener una representación mas compacta de estas variables que contenga solo la información esencial.

La idea está en observar que f_1 no cambia de valor si movemos los valores de todas las variables excepto x_1, x_2, x_5, x_6. Esto es, si consideramos un vector $w \in \mathbb{R}^{16}$ tal que

$$w_1 = w_6, \qquad w_2 = w_5$$

entonces

$$f_1(x + w) = f_1(x)$$

es decir, f_1 es invariante en el subespacio

$$\mathcal{N}_1 = \{w \in \mathbb{R}^{16} : w_1 = w_6 \text{ y } w_2 = w_5\}$$

En otras palabras, cualquier movimiento a lo largo de una dirección en $\mathcal{N}_1$, el cual es un subespacio de $\mathbb{R}^{16}$ de dimensión $16 - 2$, no cambia el valor de f_1. Entonces, tiene sentido buscar una representación de f_1 que involucre solo dos variables.

Para ver esta representación definimos las variables internas u_1 y u_2 como

$$u_1 = x_1 - x_6, \qquad u_2 = x_2 - x_5 \tag{8.2}$$

y la función interna ϕ como

$$\phi(u_1, u_2) = \frac{1}{m^2}\left(1 + \frac{m^2}{2}(u_1^2 + u_2^2)\right)^{\frac{1}{2}}$$

Formalmente, podemos representar (8.2) en forma matricial

$$u_{[1]} = U_1 x$$

donde $u_{[1]}$ es el vector de variables internas, tiene dos componentes y U_1 es una matriz de 2×16 con componenetes cero, excepto

$$U_{1,1} = 1,\ U_{1,6} = -1,\ U_{2,2} = 1,\ U_{2,5} = -1,$$

Notemos que el espacio nulo de U_1 es precisamente el subespacio invariante $\mathcal{N}_1$, esto es,

$$\mathcal{N}_1 = ker(U_1)$$

Entonces la función elemental f_1 puede escribirse de manera compacta como

$$f_1(x) = \phi(U_1 x)$$

De manera similar, esto se satisface para cada f_i, se puede representar en forma compacta considerando solo las variables involucradas en su definición. En general, para una partición de $(m+1)$ puntos en cada lado del cuadrado unitario se tienen $n = (m+1)^2$ variables y la función objetivo puede escribirse como

$$f(x) = \sum_{i=1}^{m^2} f_i(x)$$

donde

$$f_i(x) = \phi(U_i x)$$

U_i una matriz $2 \times n$ con componentes cero excepto

$$U_{1,j} = 1,\ U_{1,j+m+2} = -1,\ U_{2,j+1} = 1,\ U_{2,j+m+1} = -1$$

de donde las variables internas vienen dadas por

$$u_j = x_j - x_{j+m+2}, \qquad u_{j+1} = x_{j+1} - x_{j+m+1} \tag{8.3}$$

El subespacio invariante de cada f_i es el espacio nulo de U_i ya que si $w \in ker(U_i)$ entonces

$$\begin{aligned} f_i(x+w) &= \phi(U_i(x+w)) \\ &= \phi(U_i x + U_i w) \\ &= \phi(U_i x) \\ &= f_i(x) \end{aligned}$$

En este caso,

$$\mathcal{N}_i = \{w \in \mathbb{R}^n : \quad w_j = w_{j+m+2} \quad \text{y} \quad w_{j+1} = w_{j+m+1}\} \tag{8.4}$$

Entonces la función objetivo del problema de superficie mínima puede escribirse como

$$f(x) = \sum_{i=1}^{ne} \phi(U_i x) \tag{8.5}$$

de donde el gradiente y la matriz hessiana están dados por

$$\nabla f(x) = \sum_{i=1}^{ne} U_i^t \nabla \phi(U_i x), \quad \nabla^2 f(x) = \sum_{i=1}^{ne} U_i^t \nabla^2 \phi(U_i x) U_i \tag{8.6}$$

los cuales presentan la misma estructura de la función. Toda la información está contenida en la transformación U_i y en los gradientes y las matrices hessianas de la función interna ϕ, los cuales contienen solo unos cuantos elementos distintos de cero. Esta es, entonces una forma de descomponer a f como suma de funciones elementales y sus correspondientes variables internas.

Se tienen algoritmos muy eficientes cuando se utilizan aproximaciones Quasi-Newton para las matrices internas $\nabla^2\phi$ con las cuales se forma la aproximación a la matriz hessiana total $\nabla^2 f$, aprovechando así la estructura de f.

8.2 Subespacios invariantes y separabilidad parcial

Describiremos formalmente el concepto de separabiliad parcial, para esto introducimos primero el de subespacio invariante.

Definición. El subespacio invariante $\mathcal{N}$ de una función $f(x) : \mathbb{R}^n \to \mathbb{R}$ es el conjunto de vectores $w \in \mathbb{R}^n$ tales que

$$f(x + w) = f(x)$$

cuando x y $x + w$ pertenezcan al dominio de f.

En el ejemplo de superficie mínima vimos un subespacio invariante en (8.4), otro ejemplo simple puede obtenerse de la función elemental f definida por

$$f(x_1, \ldots, x_n) = x_{50}^2, \qquad (n > 50) \tag{8.7}$$

para la cual

$$\mathcal{N} = \{w \in \mathbb{R}^n : w_{50} = 0\}$$

Este subespacio tiene dimensión $n - 1$, tal que toda la información esencial de la función está contenida en un subespacio unidimensional ortogonal a $\mathcal{N}_i$. En este caso, la representación compacta de f es la función

$$\phi(z) = z^2$$

donde la matriz de compactificación U_i es una matriz de $1 \times n$ definida por

$$U = (\ 0 \ \cdots \ 0 \ \ 1 \ \ 0 \ \cdots \ 0\ )$$

donde el elemento no cero está en la posición 50.

Usaremos el concepto de subespacio invariante para definir la separabilidad parcial.

Definición. Una función f se dice que es *parcialmente separable* si es la suma de funciones elementales

$$f(x) = \sum_{i=1}^{ne} f_i(x)$$

donde cada f_i tiene un subespacio invariante no trivial. En otras palabras, f puede escribirse en la forma

$$f(x) = \sum_{i=1}^{ne} \phi_i(U_i x)$$

donde las matrices U_i, cuyo espacio nulo coincide con el subespacio invariante $\mathcal{N}_i$, tiene dimensión $n_i \times n$ con $n_i < n$.

Por no trivial, entendemos que el subespacio invariante no es el conjunto $\{0\}$. Para propósitos prácticos, *nos interesará esta propiedad si la dimensión de todo el subespacio invariante es cercana a* n, esto es, si

$$n_i \ll n$$

ya que toda la información esencial se puede obtener con muy pocas variables.

En muchos problemas de gran escala que resultan de la discretización de problemas de dimensión infinita, el número de funciones elementales ne aumenta cuando se refina la discretización, mientras que el número de renglones en cada U_i se mantiene pequeño e independiente de la discretización, entonces se puede tener una representación compacta de cada función elemental independientemente de cuánto se refine la discretización.

8.3 Un ejemplo

El concepto de separabilidad parcial es mas general que el de sparcidad. Se puede demostrar [25] que cada función dos veces continuamente diferenciable $f : \mathbb{R}^n \to \mathbb{R}$ cuya hessiana es *sparse* es parcialmente separable. Pero el inverso no se satisface.

Considere la función elemental

$$f(x) = (x_1 + \cdots + x_n)^2$$

cuyos gradiente y hessiana son densos. El subespacio invariante para esta función es el conjunto

$$\mathcal{N} = \{w \in \mathbb{R}^n : e^t w = 0\}, \qquad e = (1, 1, \ldots, 1)^t$$

La dimensión de $\mathcal{N}$ es $n-1$, entonces toda la información esencial de esta función está contenida en un subespacio de dimensión uno. Se puede derivar una representación compacta de la función definiendo

$$U = (1\ 1\ \cdots\ 1), \qquad \phi(z) = z^2$$

Note que la representación compacta es la misma que para la función (8.7) solo que con diferente matriz U.

Este es un ejemplo de una función con una matriz hessiana densa, pero con un subespacio invariante grande y con una estructura muy simple. Se tiene entonces que el concepto de separabilidad parcial es mas general que el de sparcidad.

8.4 Aplicación en la generación de mallas

Consideremos el problema de generar una malla en una región plana irregular, éste problema puede verse como un problema de optimización de gran escala (ver Apeéndice A). Veremos que el funcional a optimizar resulta ser una función parcialmente separable.

Una malla en una región polígonal D es una subdivisión de la región en cuadriláteros, los vértices de los cuadriláteros son llamados puntos de la malla y los cuadriláteros se llaman celdas. Por ejemplo, la figura 8.3 ilustra una malla en una región dada.

Los puntos a la frontera permanecen fijos, el problema consiste en determinar los puntos interiores de la malla, de manera que la malla resultante presente ciertas características geométricas buscadas. El problema, como se trata en [7], consiste en optimizar un funcional discreto que refleje las características buscadas en las mallas. Entonces si por ejemplo, se tiene una malla de $(m \times n)$ puntos, entonces el número de

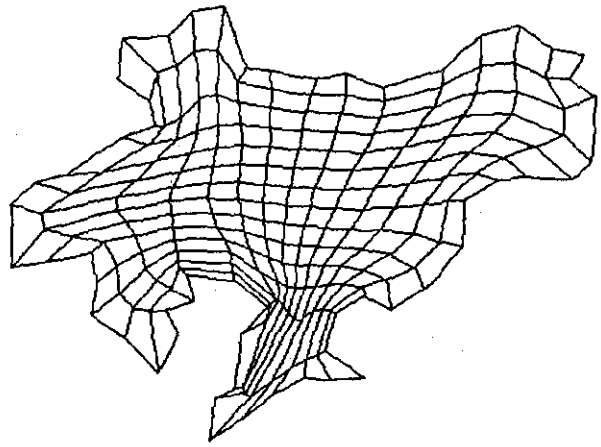


Figura 8.3: Una malla en una región dada

incógnitas es $2(m-2)(n-2)$, por lo que cuando m y n son grandes tenemos un problema de optimización de gran escala. El funcional a optimizar depende usualmente de las áreas y las longitudes de las celdas de la malla.

Si tenemos una malla de $(m \times n)$, m puntos horizontales y n verticales, representamos por $x \in \mathbb{R}^{2(m \times n)}$ el vector de coordenadas de todos los nodos de la malla. Los nodos y las celdas se numeran de izquierda a derecha y de abajo hacia arriba en la malla, así una celda c_{kl} en la malla está formada por los vértices $P_{kl}, P_{k,l+1}, P_{k+1,l+1}, P_{k+1,l}$ como se ilustra en la figura 8.4, donde $k = 1, \ldots, m-1$ y $l = 1, \ldots, n-1$.

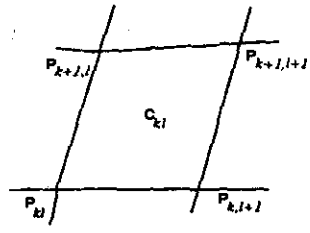


Figura 8.4: La celda c_{kl} de la malla

Para definir los funcionales utilizados para el proceso de optimización considera-

remos una celda genérica de la malla como la que se ilustra en la figura 8.5.

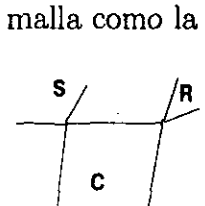


Figura 8.5: Una celda de la malla

Cada celda tiene asociada a ella cuatro triángulos que resultan de unir las diagonales de la celda como se muestra en la figura 8.6.

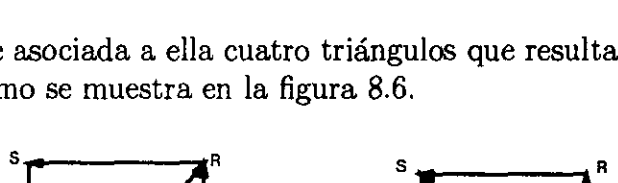


Figura 8.6: Los cuatro triángulos asociados a cada celda de la malla

La orientación al definir los triángulos en una celda es muy importante, éstos deben estar orientados positivamente, como se ilustra en la figura 8.6.

El funcional total a optimizar se describe como la suma de un funcional aplicado a cada celda o a cada triángulo de la malla, esto es, el funcional total a optimizar, trabajando por celdas, es de la forma

$$F(x) = \sum_{i=1}^{N_c} f_i(x)$$

con

$$f_i(x) = f(c_i)$$

donde c_i es una celda representativa de la malla y N_c es el número total de celdas, consideraremos un sólo índice para numerar las celdas, trabajaremos con

$$c_i = c_{kl}$$

con la relación

$$i = i(k,l) = k(m-1) + l - m + 1$$

donde $k = 1, \ldots, m-1$ y $l = 1, \ldots, n-1$.

Si consideramos una celda como en la figura 8.5 tenemos

$$f(c) = f(P, Q, R, S)$$

esto es, la función depende de los vértices que definen la celda. Como ejemplos de estos funcionales tenemos el funcional de Longitud, Area-Ortogonalidad y el de Area Clásica en donde en cada celda están definidos como

- Longitud

$$f_L(c) = \|Q - P\|^2 + \|R - Q\|^2 + \|S - P\|^2 + \|R - S\|^2$$

- Area–Ortogonalidad

$$f_{AO}(c) = \frac{1}{4}\left[(\|Q - P\|^2 + \|R - S\|^2)(\|R - Q\|^2 + \|S - P\|^2)\right]$$

- Area–Clásica

$$\begin{aligned} f_{AC}(c) &= 2 \text{ Area } (P, Q, R, S)^2 \\ &= [(P_1 - R_1)(Q_2 - S_2) - (P_2 - R_2)(Q_1 - S_1)]^2 \end{aligned}$$

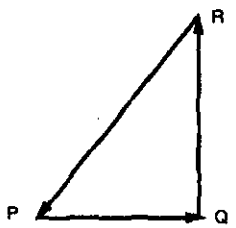


Figura 8.7: Un triángulo de una celda

Existen otros funcionales que operan sobre triángulos, éstos son de la forma

$$F(x) = \sum_{i=1}^{N_T} f_i(x)$$

con

$$f_i(x) = f(\Delta_i)$$

donde Δ_i un triángulo representativo de la malla y N_T el número total de triángulos en la malla. Consideremos un triángulo de la malla como en la figura 8.7 definimos para cada triángulo Δ

$$\begin{aligned} \alpha(\Delta) &= \alpha(Q,R,P) \\ &= 2 \text{ Area } (\Delta(Q,R,P)) \\ &= (R-Q)^t J_2 (P-Q) \end{aligned}$$

con

$$J_2 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

esto es

$$\alpha(Q,R,P) = (R_1 - Q_1)(P_2 - Q_2) - (R_2 - Q_2)(P_1 - Q_1) \tag{8.8}$$

y

$$\begin{aligned} l(Q,R,P) &= \|R-Q\|^2 + \|P-Q\|^2 \\ &= (R_1-Q_1)^2 + (R_2-Q_2)^2 + (P_1-Q_1)^2 + (P_2-Q_2)^2 \end{aligned}$$

El funcional $l(\Delta)$ mide la longitud de los lados del triángulo que pertenecen a la celda, no incluye la longitud de la diagonal de la celda la cual no pertenece a la malla. Es importante mencionar que la función $\alpha(\Delta)$ es simétrica respecto al orden en que se introduzcan los vértices del triángulo, siempre que éste mantenga la orientación positiva, en tanto que para la función $l(\Delta)$ esto no se satisface, es por esto que al definir estos funcionales trabajamos el triángulo (Q,R,P) iniciando con el vértice del centro de los lados de la celda.

Entonces como ejemplos de funcionales por triángulos, podemos mencionar

- t–Area

$$f_{tA}(\Delta) = \alpha^2$$

- k–Area

$$f_{kA}(\Delta) = \frac{1}{k+\alpha}$$

con $k \in \mathbb{R}$ constante tal que el denominador es siempre positivo.

- k-Suavidad

$$f_{ks}(\Delta) = \frac{l - 2\alpha}{k + \alpha}$$

con $k \in \mathbb{R}$ constante tal que el denominador es siempre positivo.

Nótese que

$$f(\Delta) = f(Q, R, P)$$

depende solo de 6 variables, los vértices del triángulo, entonces el problema puede verse como un problema de separabilidad parcial: la función objetivo es suma de funciones elementales donde cada función elemental depende solo de unas cuantas componentes del vector de variables.

Para ver que la función objetivo es suma de funciones elementales con subespacio invariante no trivial y la descomposición en variables internas de las funciones elementales trabajaremos por triángulos. Tenemos cuatro triángulos asociados a la celda c_i, la cual consideraremos como en la figura 8.8, donde numeramos los vértices como P_j, P_{j+1}, P_{m+j+1} y P_{m+j}.

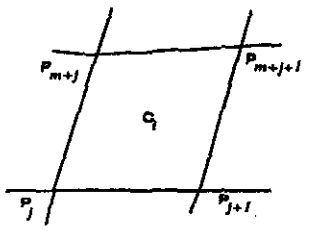


Figura 8.8: Los vérices de la celda c_i de la malla

En términos del vector de variables tenemos que los vértices de la celda están dados por

$$P_j = \begin{pmatrix} x_r \\ x_{r+1} \end{pmatrix}, \quad P_{j+1} = \begin{pmatrix} x_{r+2} \\ x_{r+3} \end{pmatrix}, \quad P_{m+j+1} = \begin{pmatrix} x_{2m+r+2} \\ x_{2m+r+3} \end{pmatrix}, \quad P_{m+j} = \begin{pmatrix} x_{2m+r} \\ x_{2m+r+1} \end{pmatrix}$$

Consideraremos la notación Δ_{ij}, $j = 1 : 4$, para denotar los cuatro triángulos de la celda c_i, los cuales definimos como

$$\Delta_{i1} = \Delta(P_{j+1}, P_{m+j+1}, P_j), \qquad \Delta_{i2} = \Delta(P_{m+j}, P_j, P_{m+j+1})$$
$$\Delta_{i3} = \Delta(P_j, P_{j+1}, P_{m+j}), \qquad \Delta_{i4} = \Delta(P_{m+j+1}, P_{m+j}, P_{j+1})$$

Entonces un funcional sobre toda la malla puede verse como

$$\begin{aligned} F(x) &= \sum_{i=1}^{N_c} f_{i1}(x) + \sum_{i=1}^{N_c} f_{i2}(x) + \sum_{i=1}^{N_c} f_{i3}(x) + \sum_{i=1}^{N_c} f_{i4}(x) \\ &= F_1(x) + F_2(x) + F_3(x) + F_4(x) \end{aligned} \tag{8.9}$$

con N_c el número total de celdas en la malla y $f_{ij}(x)$ representa el funcional aplicado sobre todos los triángulos con etiqueta j de las celdas i.

Veremos que cada uno de estos sumandos es una función parcialmente separable, con lo que la función total a optimizar resulta también parcialmente separable.

Nos será de gran utilidad definir las siguientes funciones auxiliares

$$\varphi(y_1, y_2, y_3, y_4) = y_1 y_2 - y_3 y_4$$

y

$$\psi(y_1, y_2, y_3, y_4) = y_1^2 + y_2^2 + y_3^2 + y_4^2$$

Con estas definiciones trabajaremos con detalle los funcionales de t–área, k–área y k–suavidad, para los funcionales por celdas el proceso es análogo.

Veremos primero el caso del funcional de t–área definido por triángulos.

8.4.1 Funcional de t–Area

Consideremos el funcional de t–área y una función elemental del primer sumando de 8.9, $F_1(x)$. Esto es, trabajaremos con la función

$$\begin{aligned} f_{i1}(x) &= f_{AC}(\Delta_{i1}) \\ &= \alpha(P_{j+1}, P_{m+j+1}, P_j)^2 \end{aligned}$$

El gradiente de esta función elemental respecto al vector total de variables, contiene a lo mas 6 elementos distintos de cero y la matriz hessiana es una matriz *sparse*.

Notemos que por la forma del funcional, si $w \in \mathbb{R}^{2(m \times n)}$ es tal que

$$w_r = w_{r+2} = w_{2m+r+2}$$

y

$$w_{r+1} = w_{r+3} = w_{2m+r+3}$$

entonces tenemos que para la i–ésima función elemental de área clásica, se tiene que

$$f_{i1}(x + w) = f_{i1}(x)$$

esto es, el subespacio invariante para esta función elemental está dado por

$$\mathcal{N}_{i1} = \{w \in \mathbb{R}^{2(m\times n)} : w_r = w_{r+2} = w_{2m+r+2},\ w_{r+1} = w_{r+3} = w_{2m+r+3}\} \tag{8.10}$$

el cual tiene dimensión $2(m \times n) - 4$. Para esta función elemental podemos definir las variables internas

$$\begin{aligned} u_r &= x_{2m+r+2} - x_{r+2}, \quad u_{r+1} = x_{r+1} - x_{r+3} \\ u_{r+2} &= x_{2m+r+3} - x_{r+3}, \quad u_{r+3} = x_r - x_{r+2} \end{aligned} \tag{8.11}$$

y la función interna

$$f_{i1}(x) = \varphi(u_r, u_{r+1}, u_{r+2}, u_{r+3})^2$$

La representación matricial de las variables internas está dada por

$$u_{[i1]} = U_{i1}x$$

con

$$u_{[i1]} = \begin{pmatrix} u_r \\ u_{r+1} \\ u_{r+2} \\ u_{r+3} \end{pmatrix}$$

y U_{i1} una matriz de $4 \times 2(m \times n)$ con componentes cero, excepto

$$\begin{aligned} U_{1,r+2} &= -1, & U_{1,2m+r+2} &= 1 \\ U_{2,r+1} &= 1, & U_{2,r+3} &= -1 \\ U_{3,r+3} &= -1, & U_{3,2m+r+3} &= 1 \\ U_{4,r} &= 1, & U_{4,r+2} &= -1 \end{aligned} \tag{8.12}$$

con estas definiciones tenemos que

$$F_1(x) = \sum_{i=1}^{N_c} \varphi(U_{i1}x)^2$$

esto es, $F_1(x)$ es parcialmente separable.

De manera similar se tiene que para

$$\begin{aligned} f_{i2}(x) &= f_{AC}(\Delta_{i2}) \\ &= \alpha(P_{m+j}, P_j, P_{m+j+1})^2 \end{aligned}$$

el subespacio invariante viene dado por

$$\mathcal{N}_{i2} = \{w \in \mathbb{R}^{2(m\times n)} : w_r = w_{2m+r} = w_{2m+r+2},\ w_{r+1} = w_{2m+r+1} = w_{2m+r+3}\} \quad (8.13)$$

con dimensión $2(m \times n) - 4$ y las variables internas como

$$\begin{aligned} u_r &= x_r - x_{2m+r}, \quad u_{r+1} = x_{2m+r+3} - x_{2m+r+1} \\ u_{r+2} &= x_{r+1} - x_{2m+r+1}, \quad u_{r+3} = x_{2m+r+2} - x_{2m+r} \end{aligned} \quad (8.14)$$

Para la función interna se puede definir

$$f_{i2}(x) = \varphi(u_r, u_{r+1}, u_{r+2}, u_{r+3})^2$$

Para representar las variables internas en este caso se tiene que

$$u_{[i2]} = U_{i2}x$$

y $U_{i2} \in \mathbb{R}^{4\times 2(m\times n)}$ con componentes cero, excepto

$$\begin{aligned} U_{1,r} &= 1, & U_{1,2m+r} &= -1 \\ U_{2,2m+r+1} &= -1, & U_{2,2m+r+3} &= 1 \\ U_{3,r+1} &= 1, & U_{3,2m+r+1} &= -1 \\ U_{4,2m+r} &= -1, & U_{4,2m+r+2} &= 1 \end{aligned}$$

y tenemos que

$$F_2(x) = \sum_{i=1}^{N_c} \varphi(U_{i2}x)^2$$

$F_2(x)$ es parcialmente separable.

De igual forma se tiene que

$$F_3(x) = \sum_{i=1}^{N_c} \varphi(U_{i3}x)^2$$

donde una función elemental tiene la forma

$$\begin{aligned} f_{i3}(x) &= f_{AC}(\Delta_{i3}) \\ &= \alpha(P_j, P_{j+1}, P_{m+j})^2 \end{aligned}$$

Las variables internas se definen como

$$\begin{aligned} u_r &= x_{r+2} - x_r, \quad u_{r+1} = x_{2m+r+1} - x_{r+1} \\ u_{r+2} &= x_{r+3} - x_{r+1}, \quad u_{r+3} = x_{2m+r} - x_r \end{aligned}$$

La matriz para representar las variables internas se define como $U_{i3} \in \mathbb{R}^{4\times 2(m\times n)}$ con componentes cero, excepto

$$\begin{aligned} U_{1,r} &= -1, & U_{1,r+2} &= 1 \\ U_{2,r+1} &= -1, & U_{2,2m+r+1} &= 1 \\ U_{3,r+1} &= -1, & U_{3,r+3} &= 1 \\ U_{4,r} &= -1, & U_{4,2m+r} &= 1 \end{aligned}$$

de donde el subespacio invariante viene dado por

$$\begin{aligned} \mathcal{N}_{i3} &= ker(U_{i3}) \\ &= \{w \in \mathbb{R}^{2(m\times n)} : w_r = w_{r+2} = w_{2m+r},\ w_{r+1} = w_{r+3} = w_{2m+r+1}\} \end{aligned}$$

se tiene entonces $F_3(x)$ es parcialmente separable.

Y por último para $F_4(x)$, se tiene

$$\begin{aligned} f_{i4}(x) &= f_{AC}(\Delta_{i4}) \\ &= \alpha(P_{m+j+1}, P_{m+j}, P_{j+1})^2 \end{aligned}$$

sus variables internas

$$\begin{aligned} u_r &= x_{2m+r} - x_{2m+r+2}, & u_{r+1} &= x_{2m+r+1} - x_{2m+r+3} \\ u_{r+2} &= x_{2m+r+2} - x_{r+2}, & u_{r+3} &= x_{2m+r+3} - x_{r+3} \end{aligned}$$

La matriz para identificar las variables internas es U_{i4} de $4\times 2(m\times n)$ con componentes cero, excepto

$$\begin{aligned} U_{1,2m+r} &= 1, & U_{1,2m+r+2} &= -1 \\ U_{2,2m+r+1} &= 1, & U_{2,2m+r+3} &= -1 \\ U_{3,r+2} &= -1, & U_{3,2m+r+2} &= 1 \\ U_{4,r+3} &= -1, & U_{4,2m+r+3} &= 1 \end{aligned}$$

y

$$\begin{aligned} \mathcal{N}_{i4} &= ker(U_i 4) \\ &= \{w \in \mathbb{R}^{2(m\times n)} : w_{r+2} = w_{2m+r} = w_{2m+r+2},\ w_{r+3} = w_{2m+r+1} = w_{2m+r+3}\} \end{aligned}$$

entonces

$$F_4(x) = \sum_{i=1}^{N_c} \varphi(U_{i4}x)^2$$

Entonces la función total es suma de funciones elementales donde cada función elemental tiene un subespacio invariante no trivial, esto es,

$$F_{tA}(x) = \sum_{i=1}^{N_c} \sum_{j=1}^{4} \varphi(U_{ij}x)^2$$

de donde la función es parcialmente separable.

8.4.2 Funcional de k–Area

El estudio para el funcional de k–área utiliza gran parte del análisis anterior.

Veamos el funcional por triángulos y una función elemental de $F_1(x)$ de (8.9), en este caso está definida como

$$\begin{aligned} f_{i1}(x) &= f_{kA}(\Delta_{i1}) \\ &= \frac{1}{k + \alpha(P_{j+1}, P_{m+j+1}, P_j)} \end{aligned}$$

con $k \in \mathbb{R}$ una constante tal que el denominador es siempre positivo.

El funcional de k–área tiene una estrecha relación con el de t–área, comparten las mismas variables internas, como en (8.11), se definen para el caso de k–área como

$$\begin{aligned} u_r &= x_{2m+r+2} - x_{r+2}, \quad u_{r+1} = x_{r+1} - x_{r+3} \\ u_{r+2} &= x_{2m+r+3} - x_{r+3}, \quad u_{r+3} = x_r - x_{r+2} \end{aligned}$$

La función interna sí varía con respecto a la correspondiente de t–área, en este caso se define como

$$f_{i1}(x) = \frac{1}{k + \varphi(u_r, u_{r+1}, u_{r+2}, u_{r+3})} \tag{8.15}$$

para la representación matricial de las variables internas, tenemos que

$$u_{[i1]} = U_{i1}x$$

con $U_{i1} \in \mathbb{R}^{4\times 2(m\times n)}$ con componentes como en (8.12).

El subespacio invariante de $f_{i1}(x)$ está dado por

$$\mathcal{N}_{i1} = \{w \in \mathbb{R}^{2(m\times n)} : w_r = w_{r+2} = w_{2m+r+2} \text{ y } w_{r+1} = w_{r+3} = w_{2m+r+3}\}$$

que es precisamente (8.10) el subespacio invariante de t–área trabajando sobre los triángulos Δ_{i1}.

Entonces tenemos que para este funcional de k–área

$$F_1(x) = \sum_{i=1}^{N_c} \frac{1}{k + \varphi(U_{i1}x)}$$

también resulta parcialmente separable.

De forma análoga para $f_{i2}(x)$, en el caso de k–área se tiene que el subespacio invariante es igual al definido en (8.13) y las variables internas como en (8.14) con su respectiva representación matricial donde la función interna está dada como en (8.15), con lo que queda completamente determinada la descomposición de $F_2(x)$ como suma de funciones elementales.

De igual manera resulta para $f_{i3}(x)$ y $f_{i4}(x)$ para el funcional de k–área, están relacionados con $f_{i3}(x)$ y $f_{i4}(x)$ del funcional de t–área solo cambiando la función interna como en (8.15).

Entonces tenemos que

$$F_{kA}(x) = \sum_{i=1}^{N_c} \sum_{j=1}^{4} \frac{1}{k + \varphi(U_{ij}x)}$$

8.4.3 Funcional de k–Suavidad

Para el funcional de k–suavidad tenemos una situación similar, al trabajar por triángulos se tiene que

$$\begin{aligned} f_{i1}(x) &= f_{kA}(\Delta_{i1}) \\ &= \frac{l(\Delta_{i1}) - 2\alpha(\Delta_{i1})}{k + \alpha(\Delta_{i1})} \end{aligned}$$

con $k \in \mathbb{R}$ una constante tal que el denominador es siempre positivo.

El subespacio invariante de esta función elemental está dado por

$$\mathcal{N}_{i2} = \{w \in \mathbb{R}^{2(m\times n)} : w_r = w_{2m+r} = w_{2m+r+2},\ w_{r+1} = w_{2m+r+1} = w_{2m+r+3}\}$$

que es precisamente el definido en (8.10). Las variables internas se pueden definir como

$$\begin{aligned} u_r &= x_{2m+r+2} - x_{r+2}, \quad u_{r+1} = x_{r+1} - x_{r+3} \\ u_{r+2} &= x_{2m+r+3} - x_{r+3}, \quad u_{r+3} = x_r - x_{r+2} \end{aligned}$$

que corresponden a las definidas en (8.11) para el funcional de t–área y de k–área, al trabajar con el triángulo Δ_{i1}. La función interna en este caso está dada por

$$f_{i1}(x) = \frac{\psi(u_r, u_{r+1}, u_{r+2}, u_{r+3}) - 2\varphi(u_r, u_{r+1}, u_{r+2}, u_{r+3})}{k + \varphi(u_r, u_{r+1}, u_{r+2}, u_{r+3})} \tag{8.16}$$

y la representación matricial

$$u_{[i1]} = U_{i1}x$$

con U_{i1} una matriz como en (8.12). Con estos elementos se tiene que

$$F_1(x) = \sum_{i=1}^{N_c} \frac{\psi(U_{i1}x) - 2\varphi(U_{i1}x)}{k + \varphi(U_{i1}x)}$$

Para este funcional de k–suavidad las demás funciones elementales $f_{i2}(x)$, $f_{i2}(x)$ y $f_{i4}(x)$ se tiene que los subespacios invariantes son iguales a los correspondientes para las funcionales de t–área o k–área, las variables internas también coinciden al igual que la matriz que realiza la representación compacta de estas variables. El único cambio se presenta en la función interna, la cual coincide con (8.16). Entonces se tiene que para el funcional de k–suavidad se satisface

$$F_{kS}(x) = \sum_{i=1}^{N_c} \sum_{j=1}^{4} \frac{\psi(U_{ij}x) - 2\varphi(U_{ij}x)}{k + \varphi(U_{ij}x)}$$

8.4.4 Funcional de t–Area por celdas

El análisis anterior define las funciones elementales como el funcional aplicado a cada triángulo de la malla, también se puede obtener una representación compacta del funcional por celdas. Veremos el caso para el funcional de t–área, los demás funcionales son similares, en este caso trabajamos con la función

$$F_{tA}(x) = \sum_{i=1}^{N_c} f_{tA_i}(x)$$

El funcional sobre cada celda es la suma del funcional sobre los cuatro triángulos de la celda, esto es,

$$f_{tA_i}(x) = f_{tA}(c_i) = \sum_{j=1}^{4} f_{tA}(\Delta_{ij})$$

podemos definir las variables internas para esta función elemental como

$$\begin{aligned} u_r &= x_{2m+r+2} - x_{r+2} \\ u_{r+1} &= x_{r+1} - x_{r+3} \\ u_{r+2} &= x_{2m+r+3} - x_{r+3} \\ u_{r+3} &= x_r - x_{r+2} \\ u_{r+4} &= x_r - x_{2m+r} \\ u_{r+5} &= x_{2m+r+3} - x_{2m+r+1} \\ u_{r+6} &= x_{r+1} - x_{2m+r+1} \\ u_{r+7} &= x_{2m+r+2} - x_{2m+r} \end{aligned} \tag{8.17}$$

La función interna en este caso se define como

$$\begin{aligned} \phi(u_r, u_{r+1}, \ldots, u_{r+7}) = \; & \varphi(u_r, u_{r+1}, u_{r+2}, u_{r+3})^2 \\ & + \varphi(u_{r+4}, u_{r+5}, u_{r+6}, u_{r+7})^2 \\ & + \varphi(u_{r+3}, u_{r+6}, u_{r+1}, u_{r+4})^2 \\ & + \varphi(u_{r+7}, u_{r+2}, u_{r+5}, u_r)^2 \end{aligned}$$

donde podemos representar

$$u_{[i]} = (u_r, u_{r+1}, \ldots, u_{r+7})^t$$

con

$$u_{[i]} = U_i x$$

en este caso la matriz de compactificación U_i tiene dimensión $8 \times 2(m \times n)$ con elementos cero a excepción de

$$\begin{aligned} U_{1,r+2} &= -1, & U_{1,2m+r+2} &= 1 \\ U_{2,r+1} &= 1, & U_{2,r+3} &= -1 \\ U_{3,r+3} &= -1, & U_{3,2m+r+3} &= 1 \\ U_{4,r} &= 1, & U_{4,r+2} &= -1 \\ U_{5,r} &= 1, & U_{5,2m+r} &= -1 \\ U_{6,2m+r+1} &= -1, & U_{6,2m+r+3} &= 1 \\ U_{7,r+1} &= 1, & U_{7,2m+r+1} &= -1 \\ U_{8,2m+r} &= -1, & U_{8,2m+r+2} &= 1 \end{aligned} \tag{8.18}$$

El subespacio invariante en este caso viene dado por

$$\begin{aligned} \mathcal{N}_i &= ker(U_i) \\ &= \{w \in \mathbb{R}^{2(m \times n)} : w_r = w_{r+2} = w_{2m+r} = w_{2m+r+2} \\ &\quad \text{y} \;\; w_{r+1} = w_{r+3} = w_{2m+r+1} = w_{2m+r+3}\} \end{aligned}$$

el cual tiene dimensión $2(m \times n) - 6$. Con estos elementos, podemos escribir

$$f_{tA_i} = \phi(U_i x)$$

Esta es otra forma de representar el funcional de tárea por celdas, tenemos

$$F_{tA}(x) = \sum_{i=1}^{N_c} \phi(U_i x)$$

donde se disminuye el número de funciones elementales respecto a la representación por triángulos y aumenta el número de variables internas por función elemental.

Para los funcionales de k–área y k–suavidad el proceso es análogo, las mismas variables internas, solo cambia la función interna en cada caso.

Entonces tenemos que los funcionales de optimización de mallas, operando sobre triángulos, resultan todos parcialmente separables.

8.4.5 Funcional de Area Clásica por celdas

Los funcionales que operan sobre celdas, también resultan ser parcialmente separables. Veremos el caso del funcional de Area Clásica, en donde la función a optimizar está dada por

$$F_{AC}(x) = \sum_{i=1}^{N_c} f_{AC_i}(x)$$

donde cada función elemental está dada por

$$f_{AC_i}(x) = f_{AC}(c_i)$$

En este caso se pueden definir las variables internas como

$$u_r = x_r - x_{2m+r+2}, \quad u_{r+1} = x_{r+3} - x_{2m+r+1}$$
$$u_{r+2} = x_{r+1} - x_{2m+r+3}, \quad u_{r+3} = x_{r+2} - x_{2m+r}$$

las cuales podemos acomodar en el vector de variables internas

$$u_{[i1]} = (u_r, u_{r+1}, u_{r+2}, u_{r+3})^t$$

que puede representarse como

$$u_{[i1]} = U_{i1} x$$

con U_{i1} una matriz de $4 \times 2(m \times n)$ con componentes cero, a excepción de

$$\begin{aligned} U_{1,r} &= 1, & U_{1,2m+r+2} &= -1 \\ U_{2,r+3} &= 1, & U_{2,2m+r+1} &= -1 \\ U_{3,r+1} &= 1, & U_{3,2m+r+3} &= -1 \\ U_{4,r+2} &= 1, & U_{4,2m+r} &= -1 \end{aligned}$$

de donde se tiene que como función interna podemos definir

$$f_{AC_i}(x) = \varphi(U_i x)^2$$

El subespacio invariante en este caso viene dado por

$$\begin{aligned} \mathcal{N}_{i1} &= ker(U_i) \\ &= \{w \in \mathbb{R}^{2(m \times n)} : w_r = w_{2m+r+2},\ w_{r+2} = w_{2m+r+3}, \\ &\quad w_{r+2} = w_{2m+r},\ w_{r+3} = w_{2m+r+1}\} \end{aligned}$$

el cual tiene dimensión $2(m \times n) - 4$. Entonces podemos escribir

$$F_{AC}(x) = \sum_{i=1}^{N_c} \varphi(U_i x)^2$$

Para los funcionales de longitud y área–ortogonalidad las variables internas coinciden con las definidas para el funcional de t–área al trabajar por celdas, dadas por (8.17) con la matriz de compactificación como en (8.18) y las funciones internas se definen como

$$f_{L_i}(x) = \psi(U_i x(1:4)) + \psi(U_i x(5:8))$$

y

$$F_{AO_i}(x) = \frac{1}{4}[\psi(U_i x(1:4))\psi(U_i x(5:8))]$$

Los subespacios invariantes también coinciden con los de t–área, k–área y k–suavidad al trabajar por celdas.

De donde tenemos que todos los funcionales utilizados en la generación de mallas óptimas resultan parcialmente separables.

Consideramos que la separabilidad parcial es la responsable de que el método de Newton punto a punto" para generar mallas, descrito en la siguiente sección, sea razonablemente bueno y muy eficaz para obtener mallas subóptimas que en la práctica son suficientes para la aplicación que se desea.

8.5 Sistema UNAMALLA

El sistema UNAMALLA es un sistema computacional que resuelve el problema de generar mallas óptimas en regiones planas irregulares. La forma de generar la malla óptima es por medio de un proceso de optimización, en donde se parte de una malla inicial, generada usualmente por interpolación transfinita, y optimizando algún funcional que refleja alguna propiedad geométrica buscada en las mallas, se genera una sucesión de mallas que esperamos converja a una malla óptima. Recordemos que al trabajar con mallas de $m \times n$ el número de variables a determinar es $2(m-2)(n-2)$ por lo que cuando m y n son grandes, tenemos un problema de optimización de gran escala y por lo visto en la sección anterior, todos los funcionales a optimizar resultan parcialmente separables.

Se cuenta con dos versiones del sistema: una versión bajo Unix y otra versión para PC, ambas programadas en Fortran con interfases con C. La programación del sistema UNAMALLA es modular. El módulo que nos interesa en este trabajo es el módulo de optimización, esto es, la forma de obtener la malla óptima por medio de la optimización de algún funcional. Describimos enseguida esta parte del sistema.

8.5.1 Módulo de optimización

Como mencionamos se cuenta con dos versiones del sistema una bajo Unix y otra para PC. La versión bajo Unix contempla varios métodos de optimización, algunos aplicados a problemas de gran escala como L–BFGS y Newton Truncado, y una nueva propuesta para la obtención de mallas óptimas: la optimización por el método de Newton punto a punto, la cual es la única con que cuenta el sistema para PC, mas adelante describimos en qué consiste esta forma de optimización.

Proceso de homotopía

Para los funcionales que dependen de un parámetro k real, la optimización se realiza por medio de un proceso de homotopía, esto es, podemos ver el proceso de optimización en función de un parámetro real k. Queremos determinar la trayectoria en el espacio de mallas que tiene la sucesión de mallas generada por el proceso de optimización en función de k.

El proceso que se sigue para determinar k es el siguiente: partiendo de una malla inicial x_0 se determina k_0, entonces se inicia el proceso de optimización considerando x_0 como punto inicial y manteniendo el parámetro $k = k_0$ fijo, una vez que se está cerca del óptimo para esta $k = k_0$, se actualiza el valor de k, éste está en función de la

malla de la iteración actual, $x^*_{k_0}$, se encuentra entonces k_1. Iniciando ahora el proceso de optimización con el punto inicial $x^*_{k_0}$ se realizan varias iteraciones considerando $k = k_1$ fijo. Una vez que se optimizó para el valor de $k = k_1$, se actualiza k de acuerdo a la malla actual $x^*_{k_1}$ y se encuentra k_2 y de nuevo se reinicia el proceso de optimización iniciando en $x^*_{k_1}$ y así sucesivamente.

Geométricamente, estamos siguiendo una trayectoria $x^*_{k_0}, x^*_{k_1}, \ldots$, sobre el espacio de mallas en la región considerada hasta llegar a la malla óptima para esta región, en donde $k = 0$ y ya no interviene en la optimización, se ilustra esto en la figura 8.9.

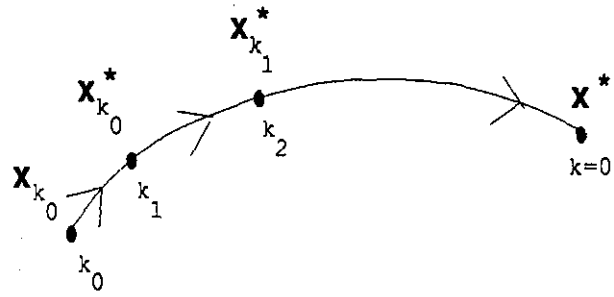


Figura 8.9: Proceso de homotopía

El valor de k depende de la malla actual, el objetivo es construir mallas convexas y suaves, esto es, que todas las celdas sea convexas y las líneas coordenadas que definen las celdas sean suaves. Se puede demostrar que una celda será convexa si el valor de $\alpha(\Delta)$, definido en (8.8) es positivo para los cuatro triángulos que se pueden obtener de una celda. Entonces se escoge k tal que $k + \alpha$ sea positivo para todo triángulo de la mallas, de ahí que se tiene que

$$k = k(\alpha_-)$$

donde α_- representa el valor del área mas pequeña de un triángulo de la malla.

Si definimos

$$\mathcal{D}_k = \{x : \alpha_-(x) > k\}$$

Entonces lo que tenemos es lo que se ilustra en la figura 8.10

Esta es, entonces la forma de realizar la optimización para los funcionales que dependen de un parámetro real k, el cálculo de k se realiza de acuerdo a la formulación obtenida por Tinoco y Barrera [52].

En la práctica no se optimiza un solo funcional sobre la malla, lo que se hace es trabajar con combinaciones convexas de funcionales para poder reflejar dos propiedades

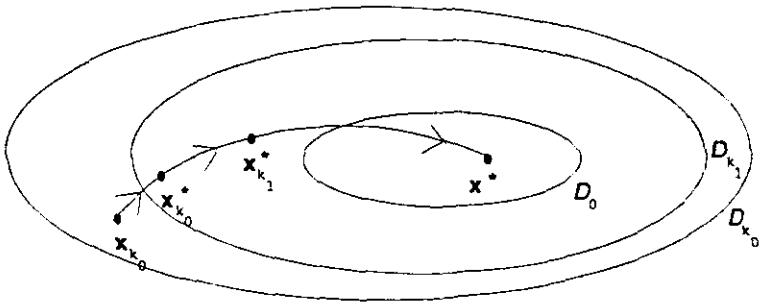


Figura 8.10: Trayectoria de la iteración en el espacio de mallas

buscadas sobre las mallas, la experiencia ha sugerido utilizar las siguientes combinaciones de funcionales

$$\begin{aligned} w(\text{k–Suavidad}) &+ (1-w)\ \text{t–Area} \\ w(\text{k–Area}) &+ (1-w)\ \text{Longitud} \\ w(\text{t-Area}) &+ (1-w)\ \text{Longitud} \\ \text{t–Area} &- \text{Ortogonalidad} \end{aligned}$$

Donde w es tal que

$$0 \leq w \leq 1$$

representa el peso asociado al funcional.

El proceso realiza otra homotopía, ahora en función de w para los funcionales de k–Suavidad y k–Area, se inicia con un peso de $w = 1$ y conforme avanza el proceso de optimización este peso se va disminuyendo hasta llegar el peso deseado, esto acelera la convergencia y produce mallas muy buenas.

Métodos de optimización

Se tienen implementados varios métodos para realizar el proceso de optimización.

L–BFGS

El sistema cuenta con una versión del método de L-BFGS descrito con detalle en la sección 7.3.1, en esta implementación se utilizan $m = 3$ pares de vectores para la actualización BFGS y se hace uso de la estrategia de región de confianza para garantizar convergencia.

Newton truncado

Otro de los métodos de optimización de gran escala implementados en el sistema UNAMALLA es el método de Newton truncado, descrito en la sección 7.4. En este caso se tiene la implementación utilizando búsqueda lineal y la solución aproximada del sistema lineal se obtiene con algunas iteraciones del método de Gradiente Conjugado utilizando solo la diagonal de la matriz hessiana para realizar el producto matriz–vector.

Newton punto a punto

Cuando se tienen mallas de grandes dimensiones los requerimientos en memoria son bastante altos ya que se están realizando procesos de optimización de gran escala, entonces se buscó realizar una optimización que trabaje bien al encontrar mallas óptimas y que al mismo tiempo no necesite grandes requerimientos para operar, esto es, que funcione cuando existan pocos recursos en memoria disponibles. Una forma de optimización con estas propiedades es la optimización de Newton punto a punto, la cual describimos enseguida.

Los funcionales a optimizar para generar las mallas óptimas, como ya hemos mencionado, son funciones parcialmente separables, tienen un subespacio invariante grande, entonces se pensó en realizar una optimización considerando la función como una función separable, donde cada función elemental dependiera solo de dos variables: las coordenadas de un nodo interior. Esto es, realizamos una optimización nodo a nodo.

La idea es la siguiente: al optimizar una malla los nodos a la frontera permanecen fijos y solo nos interesa realizar la optimización para los nodos interiores de la malla. Si consideramos una malla de 3×3 como en la figura 8.11, entonces al realizar la optimización resulta un problema de dos variables: la posición del nodo $z = (z_1, z_2) \in \mathbb{R}^2$.

Si tenemos una malla de $m \times n$ el valor de un funcional sobre una submalla de 3×3 no varía si movemos un nodo interior de la malla ajeno a esta submalla de 3×3, entonces resulta razonable realizar la optimización del funcional solo de manera local, esto es, para cada nodo interior P_{ij} extraemos la submalla de 3×3 formada por sus nodos vecinos, como se muestra en la figura 8.12, y optimizamos el funcional restringido a esta submalla de 3×3 con lo que como ya mencionamos se realiza una optimización de solo dos variables.

Una vez realizada la optimización sobre este nodo se reinserta la submalla de 3×3 ya optimizada a su posición en la malla original, de esta forma solo se alterará el nodo

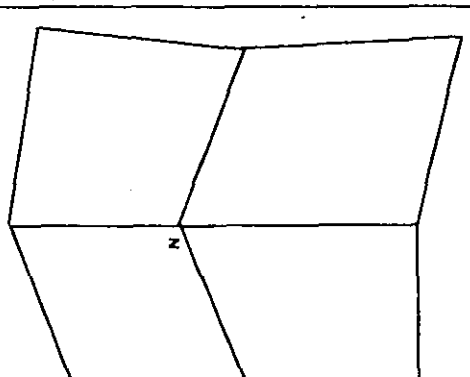


Figura 8.11: Una malla de 3×3

P_{ij} en toda la malla.

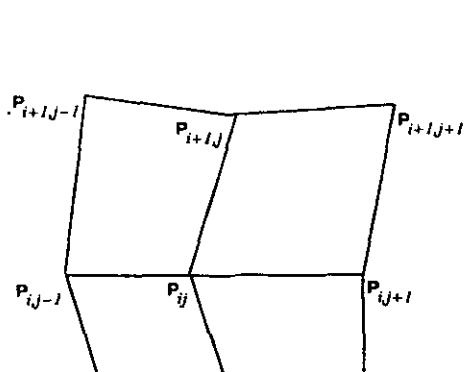


Figura 8.12: Malla local del 3×3 con los nodos vecinos del punto P_{ij}

Este proceso se realiza para cada nodo interior de la malla, con lo que los requerimientos de memoria son muy bajos, no necesitamos almacenar matrices de grandes dimensiones, solo vectores y matrices de 2×2 que se utilizan en la optimización local.

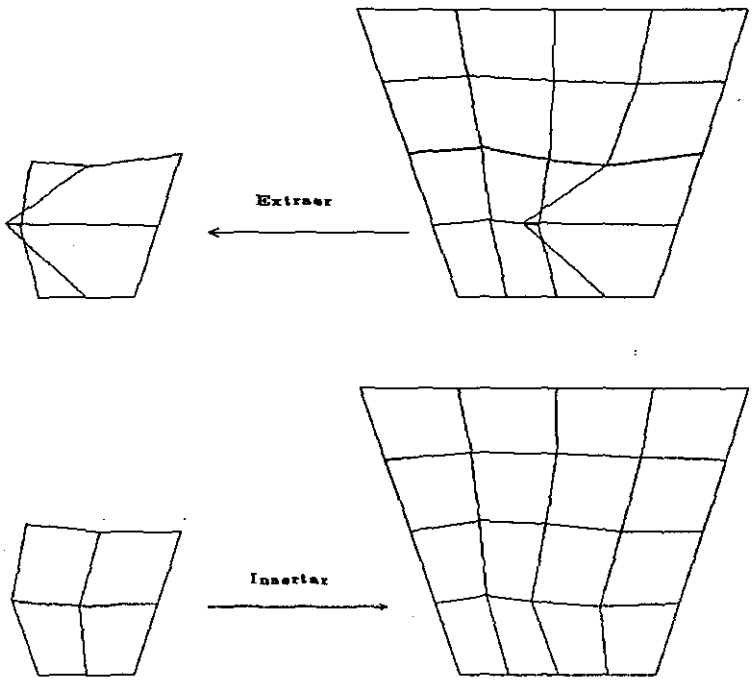


Figura 8.13: Extracción e Inserción de la submalla de 3×3

Los resultados obtenidos han sido muy satisfactorios, hemos generado mallas óptimas de grandes dimensiones con este proceso de optimización punto a punto, al final de esta sección presentamos algunas de ellas, asi como tablas comparativas con los resultados obtenidos con los métodos de optimización de gran escala.

Optimización local

Veamos ahora cómo es el proceso de optimización local. Tenemos una malla de 3×3 y un funcional definido sobre esta malla. Trabajaremos con funcionales por triángulos, la idea para celdas es similar. Hay 12 triángulos en la malla de 3×3 en los que interviene el nodo interior P_{ij}, éstos se ilustran en la figura 8.14.

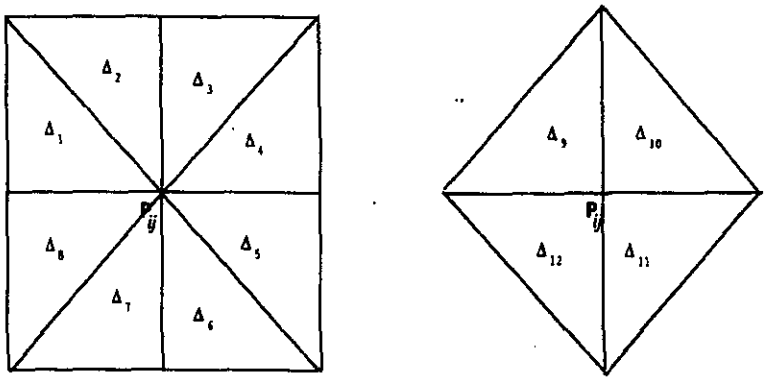


Figura 8.14: Los doce triángulos asociados al punto P_{ij}

Entonces el funcional local a optimizar es de la forma

$$f = \sum_{i=1}^{12} f(\Delta_i)$$

con $f(\Delta_i)$ el funcional sobre el triángulo Δ_i. Calculando el gradiente y la matriz hessiana locales de 2×2 analíticamente, el sistema realiza la optimización local utilizando algunas iteraciones del método de Newton en dos dimensiones, por lo que como ya mencionamos, no necesitamos almacenar mas que un vector de grandes dimensiones: el vector de variables totales, localmente solo se trabajan vectores y matrices de 2×2, por lo que el sistema puede operar con pocos recursos.

Otra de las características de la optimización punto a punto es que es un proceso paralelizable, se puede aplicar en máquinas en paralelo optimizando varios puntos interiores a la vez de forma independiente, con lo cual se reduciría considerablemente el tiempo de optimización. Esto forma parte de un trabajo a futuro.

8.5.2 Resultados numéricos: una muestra

Presentamos enseguida los resultados obtenidos al generar algunas mallas con este método de optimización Newton punto a punto, comparado con los resultados de optimizar las mallas con métodos clásicos de gran escala como L–BFGS y Newton Truncado. Se presentan dos tablas una para reportar el número de iteraciones necesitadas para lograr convergencia y otra para reportar el tiempo necesitado en la optimización. En todos los casos se trabajaron mallas de 40×40 y el funcional a optimizar es

$$w(\text{k–Suavidad}) + (1 - w)\ \text{t–Area}$$

con $w = 0.5$

Los resultados de obtuvieron en una computadora Pentium II a 233 Mhz con 64MB de memoria RAM con sistema operativo Linux Intel Red Hat 5.2.

Tabla 1. Número de iteraciones utilizadas en la optimización.

	L–BFGS	Newton Trun.	Punto a Punto
England	1214	419	314
Habana	620	68	184
Rusia	825	63	265
Sudamérica	103	341	153
Gato	2285	124	536

Tabla 2. Tiempo utilizado para la optimización.

	L–BFGS	Newton Trun.	Punto a Punto
England	1:41.1	58.5	1:04.0
Habana	50.1	56.3	25.4
Rusia	1:08.1	51.8	23.3
Sudamérica	33.6	40.0	35.9
Gato	3:19.3	46.7	2:16.6

Se observa de las tablas anteriores que el método Newton punto a punto opera razonablemente bien comparado con L–BFGS y con Newton Truncado, L–BFGS es el que realiza mayor número de iteraciones para converger y es, en general, el que mas tiempo necesita para operar. Newton Truncado trabaja mas rápido con buenos resultados, sin embargo como hemos mencionado el método Newton punto a punto requiere muy pocos recursos para operar, lo que lo hace un método atractivo si no se cuenta con grandes recursos para la obtención de la malla óptima.

Presentamos enseguida las mallas obtenidas al realizar el proceso de optimización para las regiones reportadas.

Al final de este trabajo, en el Apéndice D, presentamos el manual de operación para el sistema UNAMALLA en su versión para PC.

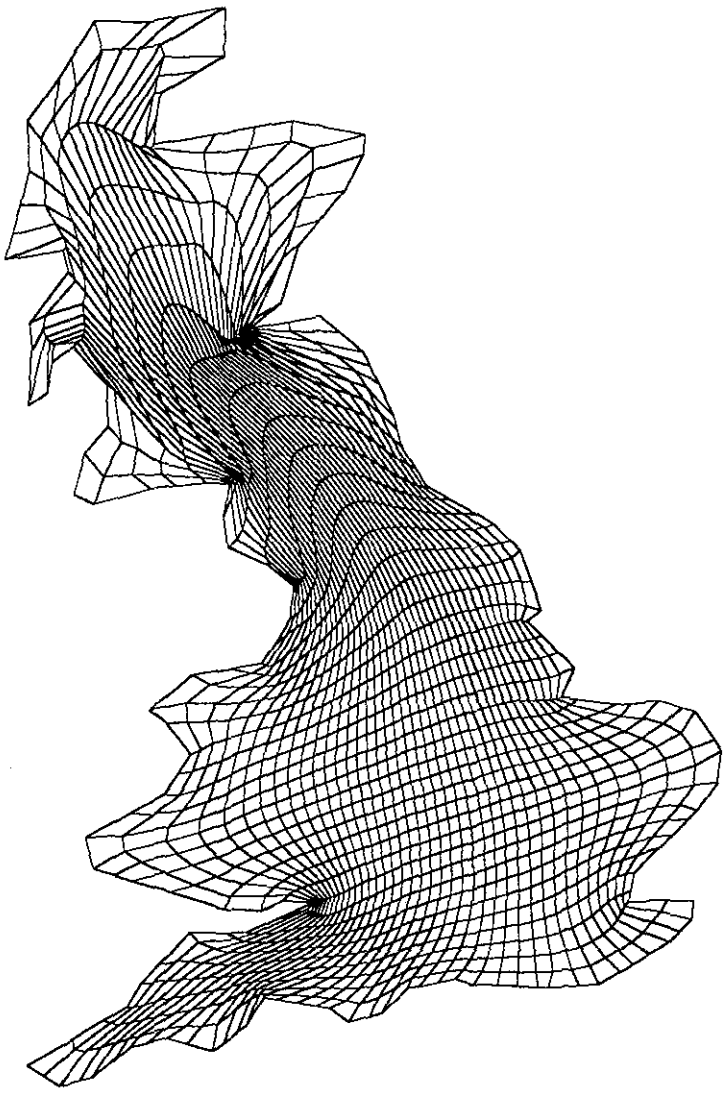


Figura 8.15: Malla de 40 × 40 obtenida sobre la región *England*

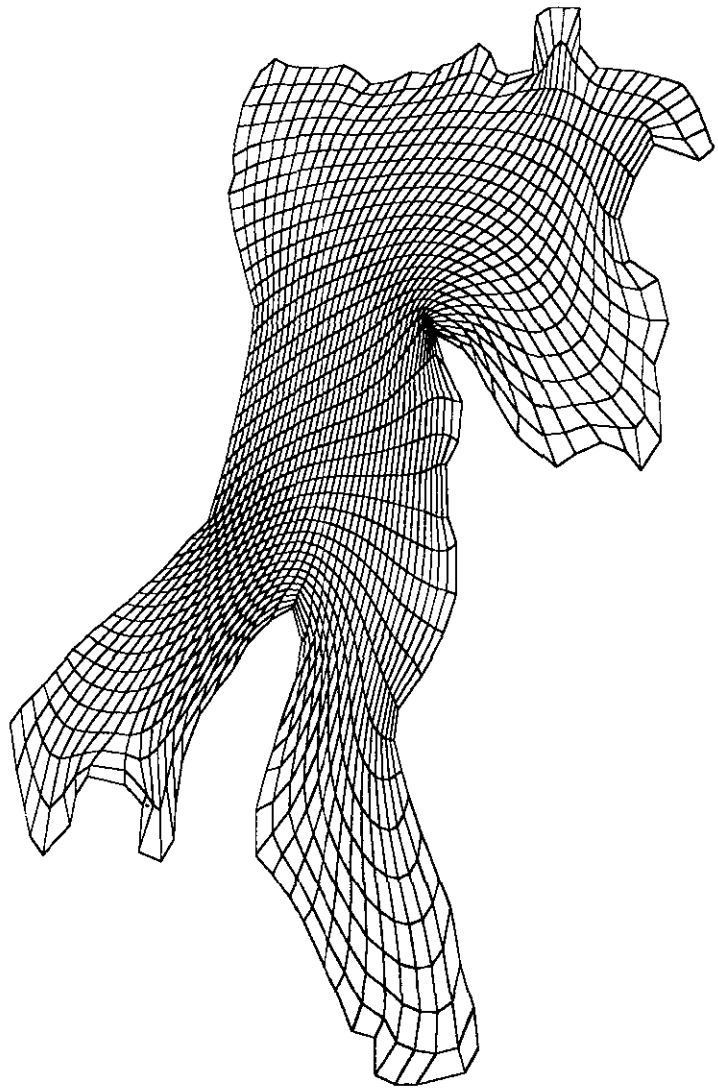


Figura 8.16: Malla de 40×40 obtenida sobre la región *Habana*

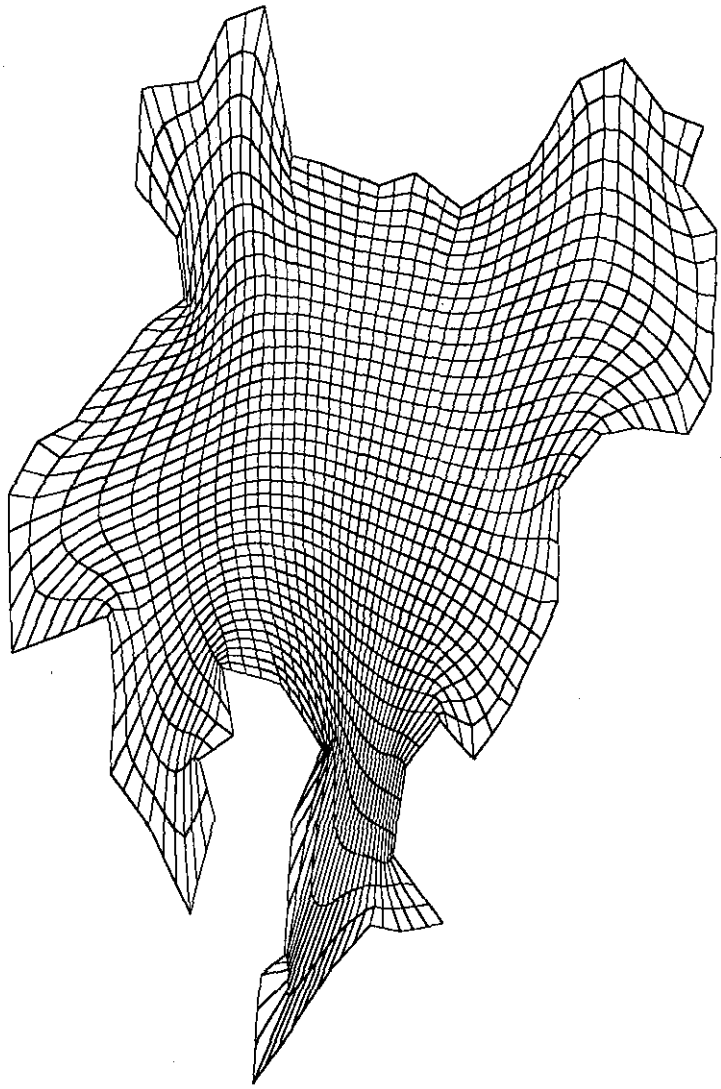


Figura 8.17: Malla de 40×40 obtenida sobre la región *Rusia*

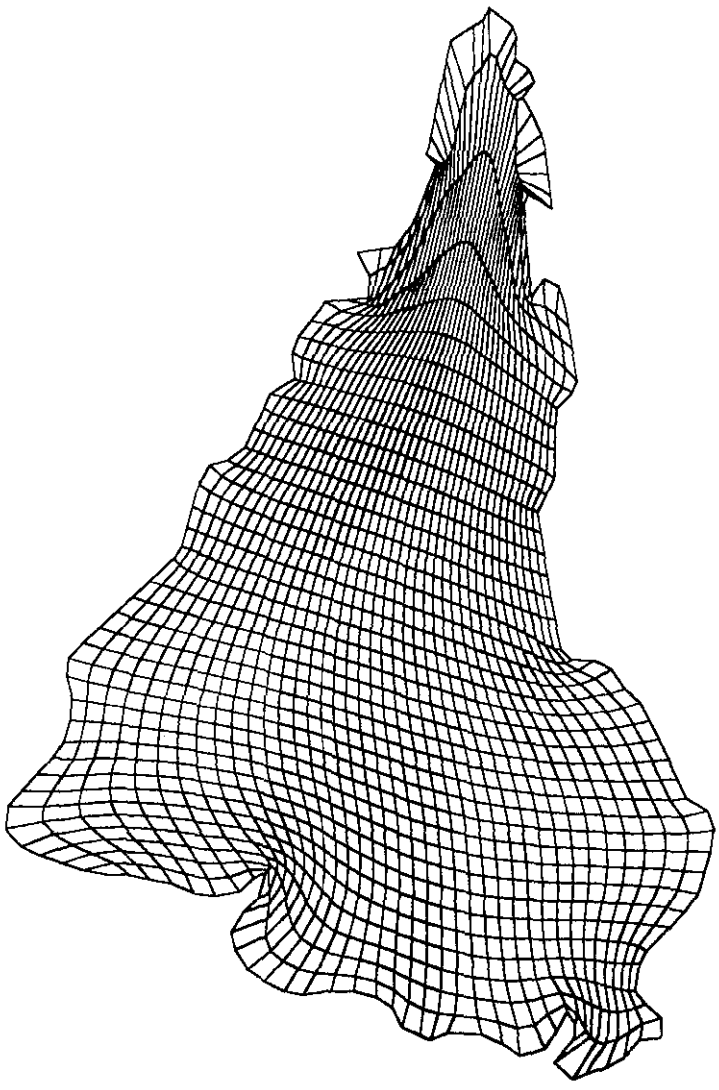


Figura 8.18: Malla de 40 x 40 obtenida sobre la región *Sudamérica*

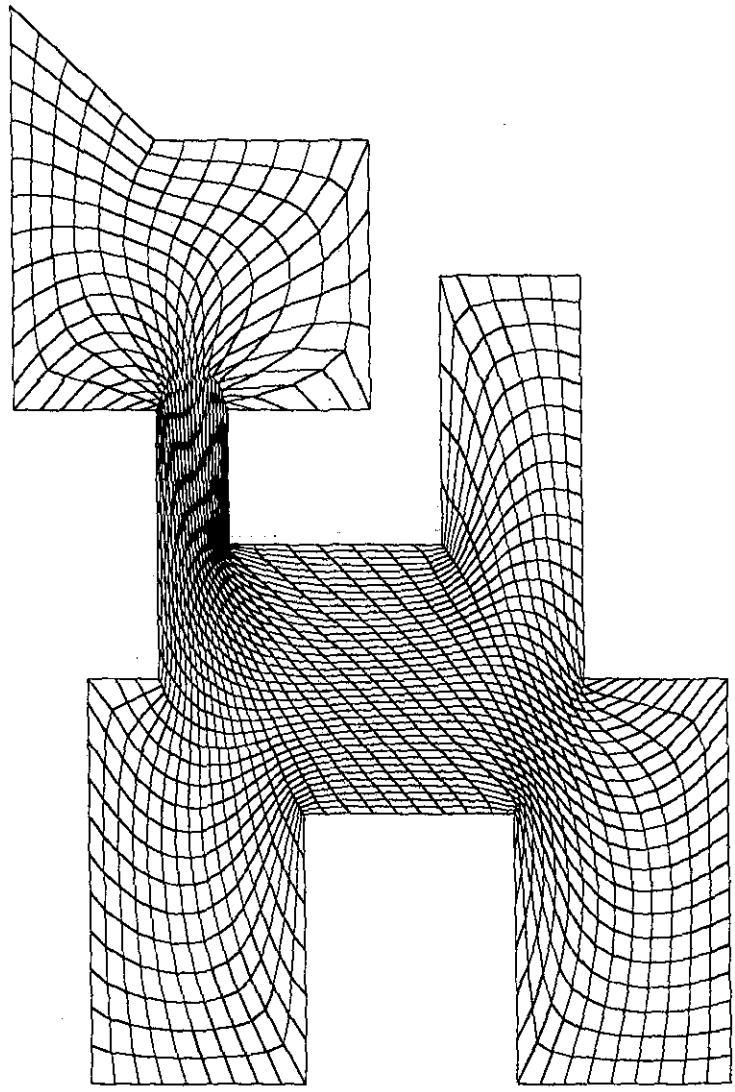


Figura 8.19: Malla de 40×40 obtenida sobre la región *Gato*

De la experimentación con el sistema computacional se han observado buenos resultados realizando la optimización de Newton punto a punto, solo en regiones muy irregulares es donde se ha presentado dificultades del método al operar, pero en general, funciona bien este método.

Se ha intentado considerar submallas más grandes, en lugar de solo las de 3×3 al realizar la optimización local, pero no hemos obtenido, hasta el momento, buenos resultados; seguiremos trabajando en esta dirección.

Referencias

P. Barrera, L. Castellanos, A. Pérez (1994). Métodos Variacionales Discretos para la Generación de Mallas, Fac. Ciencias, UNAM, México.

P. Barrera, G. Tinoco (1997). Smooth and Convex Grid Generation over General Plane Regions, *Mathematics and Computers in Simulation*, no. 46, pp. 87–102.

A. Griewank and L. Toint, (1982). On the unconstrained optimization of partially separable objetive functions, *Nonlinear Optimization 1981,* M.J.D. Powell ed., Academic Press (London), pp. 301–312.

J.G. Tinoco (1997). Funcionales Discretos para la Generación de Mallas Suaves y Convexas sobre Regiones Planas Irregulares, Tesis Doctoral, CIMAT, Guanajuato, México.

Bibliografía

[1] M. Al–Baali (1985). Descent property and global convergence of the Fletcher–Reeves method with inexact line search, *IMA J. Numer. Anal.* vol. 5, pp. 121–124.

[2] P. Barrera, L. Castellanos, R. Alvarez (1992). *Un punto de vista sobre el desarrollo de los métodos de gradiente conjugado no lineal* Ed. Academia Cuba.

[3] P. Barrera, L. Castellanos, R. Ojeda, A.Pérez (1989). A New Discrete Functional for Grid Generation, *Proceedings of The Fifth México–United States Workshop on held Advances in Numerical Partial Differential Equations and Optimization*, Mérida, Yucatán, México.

[4] P. Barrera, L. Castellanos, A. Pérez (1993). Curvilinear Coordinate System Generation over plane irregular regions, Fac. Ciencias, UNAM, México.

[5] P. Barrera, L. Castellanos, A. Pérez (1994). Manual de usuarios del sistema UNAMALLA: Generación de mallas planas sobre regiones irregulares, Fac. Ciencias, UNAM, México.

[6] P. Barrera, L. Castellanos (1994). Métodos de Optimización de Gran Escala para el Problema de la Generación de Redes Optimas, Fac. Ciencias, UNAM, México.

[7] P. Barrera, L. Castellanos, A. Pérez (1994). Métodos Variacionales Discretos para la Generación de Mallas, Fac. Ciencias, UNAM, México.

[8] P. Barrera, G. Tinoco (1998). Area control in Generating Smooth and Convex Grid over General Plane Regions, Preprint.

[9] P. Barrera, G. Tinoco (1997). Smooth and Convex Grid Generation over General Plane Regions, *Mathematics and Computers in Simulation*, no. 46, pp. 87–102.

[10] E. Beale (1972). *A derivation of Congujate Gradients* Numerical Methods for nonlinear optimization, Academic Press, London, pp.39–43.

[11] R. Byrd, J. Nocedal and R. Schnabel (1994). Representations of Quasi–Newton matrices and their use in limited memory methods, *Mathematical Programming* no. 63, pp. 129–156.

[12] R. Byrd, J. Nocedal and Y. Yuan (1987). Global Convergence of a Class of Quasi–Newton Methods on convex problems, *SIAM J. Numer. Anal.* vol. 24, no. 5, pp. 1171–1190.

[13] J. Castillo, (1991). A Discrete Variational Grid Generation Method, *SIAM J. Sci. Stat. Comp.* vol. 12 no. 2, pp. 454–468.

[14] R. Dembo, S. Eisenstat and T. Steihaug, (1982). Inexact Newton methods, *SIAM J. Numer. Anal.* vol. 19, pp. 400-408.

[15] R. Dembo and T. Steihaug, (1983). Truncated Newton algorithms for large–scale optimization, *Math. Programming* vol. 26, pp. 190-212.

[16] J. Dennis, J. Moré, (1977). Quasi–Newton Methods, Motivation and Theory, *SIAM review*, vol. 19, No.1, pp.46–89.

[17] J. Dennis, R. Schnabel, (1979). Least Change Secant Updates for Quasi–Newton Methods, *SIAM Review* vol. 21, no. 4, pp. 443–459.

[18] J. Dennis, R. Schnabel, (1983) *Numerical Methods for Unconstrained Optimization and nonlinear equations* Prentice–Hall.

[19] L. Dixon, (1972). Variable metric algorithms: Necessary and sufficient conditions for identical behavior on nonquadratic functions, *J. Optimization Theory Appl.* vol. 10, pp.34–40.

[20] S. Eisenstat and H. Walker (1994). Globally convergent inexact Newton methods, *SIAM J. Optimization* vol. 4, pp. 393–442.

[21] R. Fletcher (1987). *Practical Methods of Optimization* John Wiley & Sons, 2nd.Edition.

[22] R. Fletcher and C. Reeves (1964). Function Minimization by Conjugate Gradientes, it Comput. J. no. 7, pp. 149–154.

[23] J. Gilbert and C. Lemaréchal (1989). Some numerical experiments with variable storage Quasi–Newton algorithms, *Math. Programming* vol. 45, pp. 407–436.

[24] G. Golub, Ch Van Loan (1989). *Matrix Computations* The Johns Hopkins University Press, 2nd. Edition.

[25] A. Griewank and L. Toint, (1982). On the unconstrained optimization of partially separable objetive functions, *Nonlinear Optimization 1981,* M.J.D. Powell ed., Academic Press (London), pp. 301–312.

[26] C. Kelley (1995) *Iterative Methods for linear and non linear equation* SIAM Publications.

[27] C. Kelley (1999) *Iterative Methods for optimization* SIAM Publications.

[28] P. Knupp and S. Steinberg (1996). *Fundamentals of Grid,* Elseiver Publishing.

[29] D. Liu and J. Nocedal (1989). On the limited memory BFGS method for large scale optimization, *Math. Programming* vol. 45, pp. 503–528.

[30] J. Moré (1990). A collection of nonlinear model problems in Computational Solution of Nonlinear Systems of Equations, E. Allgower and K. Georg Eds., *Lectures in Applied Mathematics* vol. 26, pp. 723–762. American Mathematical Society, Providence, Ri.

[31] J. Moré (1983). Recent Developments in Algorithms and Software for Trust Region Methods, *Mathematical Programming, The State of the Art Bonn 1982* Ed. by A. Bachem, M. Grötschel and B. Korte, Springer–Verlag, pp. 258–287.

[32] J. Moré, D. Sorensen, (1983). Computing a Trust Region Step, *SIAM J. Sci. Stat. Comput.* vol. 4, no. 3, pp. 553–572.

[33] J. Moré, D. Sorensen (1984). Newton's Method, *Studies in NUmerical Analysis,* Ed. G.H. Golub, pp. 29–82.

[34] J.Moré, D.J. Tuente, (1994). Line Search Algorithms with guaranteed sufficient decrease, *ACM Transactions on Mathematical Software* vol. 20, pp. 286–307.

[35] S. Nash (1982). *Truncated–Newton Methods.* Ph.D. thesis, Stanford University.

[36] S. Nash and J. Nocedal (1991). A Numerical Study of the Limited Memory BFGS Methods and the Truncated–Newton Method for large scale optimization, *SIAM Journal on Optimization* vol. 1, no. 3, pp. 358–372.

[37] L. Nazareth (1979). A Relationship between the BFGS and Conjugate Gradients Algorithms and its implications for new algorithms, *SIAM J. Numer. Anal.* vol. 16, pp. 294–300.

[38] J.Nocedal (1980). Updating quasi–Newton matrices with limited storage, *Mathematics of Computation*, vol. 35, pp.773–782.

[39] J. Nocedal and J. Wright (1999). *Numerical Optimization*, Springer Series in Operations Research.

[40] E. Polak and G. Ribiére (1969). Note sur la convergence de méthodes de directions conjuguées, *Revue Franaise d'Informatique et de Recherche Opérationnelle* vol. 16, pp. 35–43.

[41] M. Powell (1971). On the convergence of the variable metric algorithm, *J. Inst. Math. Appl.* no. 7, pp. 21–36.

[42] M. Powell (1972). Some properties of the variable metric method, *Numerical Methods for Non-linear optimization* F.A. Loostma ed. Academic Press, London

[43] M. Powell (1976). Some global properties of a variable metric algorithm for minimization without exact line search, *Nonlinear Programming, SIAM–AMS Proceedings* vol. 9 American Mathematical Society, Providence, R.I.

[44] M.Powell (1977). Restart Procedures for the Conjugate Gradient Method, *Math. Programming* vol. 12, pp. 241–254.

[45] M. Powell (1984). Nonconvex minimization calculations and the congujate gradient method, *Lectures Notes in Mathematica 1006* Springer–Verlag, Berlin, pp. 122-141.

[46] M. Powell (1976). Some global convergence properties of a variable metric algorithm without line searches, *SIAM–AMS Proceedings* vol. IX, Math. Prog., pp.53–72.

[47] D. Shanno (1978). Conjugate Gradient Methods with inexact searches, *Mathematics of Operations Research* vol. 3, no. 3, pp. 244–256.

[48] D. Shanno and K. Phua (1977). Matrix Conditioning and Nonlinear Optimization, *Math. Programming* vol. 14, pp. 149–160.

[49] T.Schlick and A. Fogelson (1992). TNPACK – A truncated Newton package for large–scale problems: I.Algorithms ans usage, *ACM Transactions on Mathematical Software*, vol. 18, no.1, pp.46–70.

[50] D.Sorensen, (1982). Newton's Method with a Model Trust Region Modification, *SIAM Numer. Anal.* vol. 19, no. 2, pp. 409–426.

[51] T.Steihaug (1983). The conjugate gradient method and trust regions in large scale optimization, *SIAM J.Numer. Anal.*, vol. 20, pp.626–637.

[52] J.G. Tinoco (1997). Funcionales Discretos para la Generación de Mallas Suaves y Convexas sobre Regiones Planas Irregulares, Tesis Doctoral, CIMAT, Guanajuato, México.

[53] G. Zoutendij (1970). Nonlinear Programming, computational methods, *Integer and Nonlinear Programming* J. Abadie ed. North–Holland, Amsterdam, pp. 37–86.

Apéndice A

Funcionales en la generación de mallas

Una malla en una región poligonal D es una subdivisión de la región en cuadriláteros. Los vértices de los cuadriláteros son llamados puntos de la malla y los cuadriláteros se llaman celdas.

Una malla G de *mxn* en una región D es un conjunto de *mxn* puntos $\{P_{ij}\}$, $1<= i <=m$, $1<=j<=n$, tales que pueden ser puestos en correspondencia uno a uno con una malla uniforme en el cuadrado unitario tal que los puntos en la frontera corresponden a la frontera de D. El problema es determinar los puntos interiores de la malla.

Estamos interesados en mallas que sean suaves y convexas, para lograrlo construimos un funcional F definido sobre todos los puntos de la malla $\{P_{ij}\}$, usualmente F depende de las áreas y las longitudes de los lados de la celda, de tal forma que F tendrá un valor óptimo en una malla suave y convexa, lo que escribimos

$$G^* = \arg\min_{G} F$$

La dimensión del problema de optimización es $N=2(m-2)(n-2)$, lo que representa un problema de optimización de gran escala.

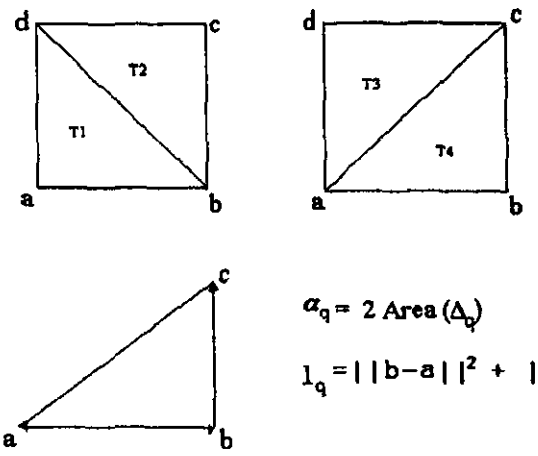
Funcionales

Como se ha mencionado, la malla óptima, suave y convexa, se obtiene de la optimización de un funcional que depende de todos los nodos de la malla, donde el funcional a optimizar tiene la forma

$$F = \sum_{q=1}^{N_c} f(c_q)$$

con N_c el número de celdas sobre G y c_q una celda representativa.

Para definir el funcional sobre cada celda trabajaremos sobre los triángulos asociados a cada una de ellas y obtenidas a partir de las diagonales; con esto definimos



Los funcionales utilizados en el sistema están dados por

- Area Clásica

  Geométricamente esta función mide el área de los cuatro triángulos de la celda. El

$$f_{AC}(c_q) = \sum_{q=1}^{4} \alpha_q^2$$

  óptimo del funcional F_{AC} se alcanza, en principio, en una malla con todas sus celdas de igual área.

- Longitud

$$f_L(c_q) = \| b-a \|^2 + \| c-d \|^2 + \| d-a \|^2 + \| c-b \|^2$$

  Esta función geométricamente mide la tensión de las líneas de cada celda, así el óptimo del funcional tensa las líneas coordenadas.

- k-Suavidad

$$f_{kS}(c_q) = \sum_{q=1}^{4} \frac{l_q - 2\alpha_q}{k + \alpha_q}$$

  donde k es un parámetro real que se escoge de forma tal que el denominador es

siempre positivo. El óptimo del funcional se alcanza en una malla con líneas curvilíneas muy suaves.

- k-Area

$$f_{kA}(c_q) = \sum_{q=1}^{4} \frac{1}{\alpha_q + k}$$

donde k es un parámetro real que se escoge para que el denominador sea siempre positivo. Este funcional guarda una relación con el funcional de Area Clásica en cuanto a los puntos críticos, en el óptimo se evitan celdas con área muy pequeña.

- Area-Ortogonalidad

$$f_{AO}(c_q) = \frac{1}{4}\left(\| b-a \|^2 + \| c-d \|^2\right) * \left(\| d-a \|^2 + \| c-b \|^2\right)$$

Este funcional es una combinación entre Area Clásica y Ortogonalidad con un peso asociado a cada uno de ellos de .5. Este funcional busca en el óptimo mallas "casi" ortogonales y "casi" uniformes en área. Por su composición es muy fácil de implementar y tiene la característica de que las líneas curvilíneas generadas son suaves.

El sistema explota la propiedad de separabilidad parcial de los funcionales realizando el proceso de optimización de manera puntual logrando con ello reducir el tiempo y recursos computacionales. Con esta técnica hemos obtenidos buenos resultados.

Apéndice B

Formato de entrada y salida de los archivos claves.

Los dos tipos de archivos fundamentales que se manejan en UNAMALLA son:

- El archivo de coordenadas de la frontera, o archivo de Contorno.

  Las coordenadas de la frontera se deben dar en la siguiente forma: Se tiene que indicar cuáles son los vértices que definen las cuatro fronteras de la región, y por supuesto, su orden. En este caso el archivo tiene la forma:

```
n ibflag nb1 nb2 nb3 nb4
x_1      y_1
x_2      y_2
.        .
.        .
.        .
x_n-1    y_n-1
x_1      y_1
```

donde

`n`	Es el total de puntos de la frontera (contando el primero dos veces)
`ibflag`	`=1` Se van a dar los vértices de la región y las fronteras 1, 2, 3 y 4 son distintas.
	`=13` Si las fronteras 1 y 3 son iguales (esto es para el caso de regiones con un hueco lo que hace que se tengan dos fronteras iguales).
	`=24` Si las fronteras 2 y 4 son iguales (similar al caso anterior).
	`=0` Elección automática de las fronteras.
`nb1`	Cantidad de puntos en la frontera 1.
`nb2`	Cantidad de puntos en la frontera 2.

nb3 Cantidad de puntos en la frontera 3.

nb4 Cantidad de puntos en la frontera 4.

x_i, y_i Coordenadas de los puntos de la frontera.

Un ejemplo de un archivo donde se indican los vértices que definen la frontera para la figura B1 es el siguiente:

```
13 13 2 7 2 5
    0.0      0.0
    9.0      0.0
   12.0      4.0
   16.0      4.0
   19.0      1.0
   15.0     -3.0
   11.0     -3.0
    9.0      0.0
    0.0      0.0
   20.0    -15.0
   30.0      1.0
   12.0     15.0
    0.0      0.0
```

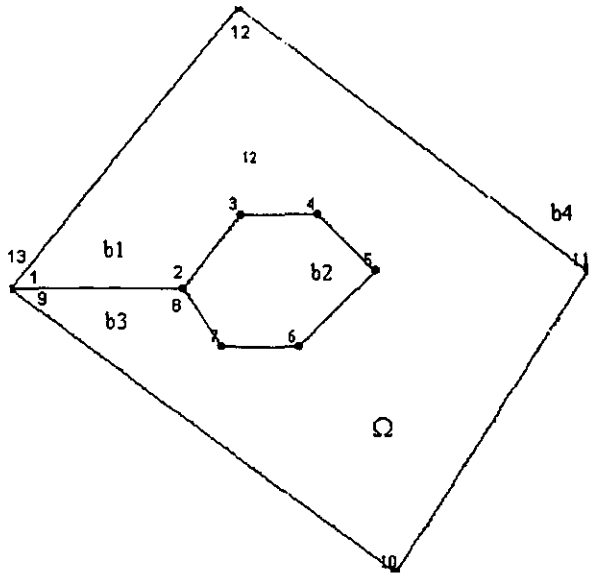


Figura B1. En éste ejemplo las fronteras 1 y 3 coinciden.

Por supuesto no es la única forma de generar mallas sobre regiones con agujeros pero la idea de unir por líneas curvilíneas las fronteras opuestas dos a dos va en este sentido.

Un hecho que no debemos pasar por alto es la orientación del contorno. Dado que deseamos mallas convexas una manera de garantizar que la aproximación a la solución sea confiable es considerar que el contorno tenga orientación positiva, esto es, contraria a las manecillas del reloj, vease [3] y [4] para los detalles. Bajo esto, el sistema detecta si el contorno tiene orientación positiva, en caso de no ser así el sistema, previo aviso, cambia la orientación del contorno.

- El archivo de coordenadas de la malla, tiene la siguiente estructura:

```
il jl
[nombre_archivo]
x_11   y_11   x_21   y_21      x_31      y_31      x_41
y_41   x_51   .      .         .         .         .
.      .      .      .         .         .         .
.      .      .      .         .         .         .
x_22   y_22   x_23   y_23      x_24      y_24      x_24
x_25   .      .      .         .         .         .
.      .      .      .         .         .         .
.      .      .      .         .         .         .
.      .      .      x_m,1n-1  y_m,n-1   x_mn      y_mn
```

(véase [5] y [6] para una discusión a detalle) donde `mn=2*(il*jl)`,

`il`	Es la cantidad de puntos horizontales de la malla
`jl`	Es la cantidad de puntos verticales de la malla
`Nombre_archivo`	Nombre del archivo. Esta línea es opcional, más sin embargo, la línea debe *permanecer*.
`x_i, y_i`	Son las coordenadas de los nodos de la malla, que se imprimen siete cada línea, primero las correspondientes a la frontera 1, después las correspondientes a la frontera2, luego las de la frontera 3 y después las de la frontera 4, enseguida los puntos interiores los cuales se imprimirán por líneas verticales empezando con la línea paralela siguiente a la frontera 4 y siguiendo hacia la derecha. Todas las coordenadas *tienen que estar en un* formato de 5 enteros y 4 decimales más un espacio para el signo y el punto decimal (en FORTRAN, el el formato 7F11.4).

Estos archivos tienen el mismo formato, ya sea en la entrada o en la salida, y para todos los módulos que los lean o guarden resultados de este tipo.

Apéndice C

Detalles técnicos y requerimientos mínimos del sistema

El sistema UNAMALLA v. 2.0 para PC está escrito en su totalidad en lenguaje FORTRAN 77 de Microsoft FORTRAN v. 5.1 usando la interfase entre las bibliotecas de Microsoft C 6.00A, en particular la gráfica y la de sistema para facilitar el despliegue y las opciones de captura de datos.

Este sistema emplea únicamente la memoria disponible de la memoria base de la computadora por lo que está limitada la dimensión máxima posible a operar dentro del sistema por la memoria disponible en el instante de ejecución.

Los requerimientos mínimos del sistema van en considerar que se cuenta con al menos una computadora 386, con sistema operativo 5.0, una tarjeta de video que soporte VGA, la memoria base de 640kb y un espacio en disco de 1.7Mb para los archivos de sistema, la información del sistema y los ejemplos de contorno y de mallas. En el aspecto del manejo de archivos se recomienda modificar el **config.sys** del sistema operativo en las líneas de files y de buffers a fin de que al menos se cuente con `FILES=40` y de `BUFFERS=50`.

Para su instalación basta copiar el archivo de distribución **sistema.exe** a algún directorio libre de archivos y al ejecutarlo se descomprimirán una colección de archivos entre los que se encuentra el programa ejecutable malla.exe, los archivos de contorno y de mallas para experimentar con el sistema.

Este sistema es resultado de los últimos años en investigaciones por parte de profesores de la Universidad Nacional Autónoma de México, de la Universidad Michoacana de San Nicolás de Hidalgo y de la Universidad Autónoma de Coahuila, cuyos resultados han venido presentándose en diversos foros nacionales e internacionales.

Actualmente se cuenta con una versión del sistema para distintas Estaciones de Trabajo para su desarrollo se hace uso de la biblioteca gráfica `OpenGL` y los Graphical User Interface de `Xforms`, este es `UNAMALLA System v. 2.0 for X Windows`, también se cuenta con una versión del sistema para MATLAB, ésta hace uso de los Graphical User Interface propios de MATLAB, así como del Aplication Program Interface, ya que se realiza una interfase con Fortran al realizar la optimización, lo cual reduce el tiempo de ejecución considerablemente.

Todos los documentos relativos al sistema están a disposición del interesado en el URL: http://www.mathmoo.unam.mx/unamalla, en este lugar puede hacerse de una copia en línea de este manual y del sistema así como la versión del sistema para Estaciones de Trabajo y para MATLAB. Para contactar con el grupo UNAMALLA envíe un correo electrónico a la dirección unamalla@athena.fciencias.unam.mx

Apéndice D

Manual Operativo del Sistema UNAMALLA v. 2.0 para PC

D1. Introducción

El Sistema UNAMALLA v.2.0 para PC es un sistema computacional para resolver el problema de generar mallas suaves y convexas, en regiones planas irregulares.

El sistema está escrito en su totalidad en Microsoft FORTRAN 77 v.5.1 y haciendo uso de la interface gráfica de las bibliotecas de Microsoft C v. 6.00A ha resultado ser un sistema amigable al usuario mediante el uso de menús de entrada para las opciones frecuentes y el uso del ratón para generar el contorno de la malla.

La malla generada es una malla óptima que resulta de la optimización de un funcional que depende únicamente de los puntos de la malla (ver Apéndice A para la descripción de los funcionales). Para realizar esta optimización se utiliza un método iterativo partiendo de una malla inicial, generada usualmente por interpolación. Se utiliza un método de optimización para generar aproximaciones a la malla óptima.

Las versiones anteriores utilizan técnicas de optimización de gran escala para encontrar la malla óptima, estas técnicas requieren del almacenamiento de varios vectores de dimensión al menos el número total de variables. El presente sistema explota la estructura de la función objetivo que depende fuertemente de los puntos alrededor suyo y no optimiza todas las variables a la vez, realiza una "optimización local". Esto es, se ve a la malla total como la unión de submallas de *3x3* y se optimizan estas submallas. Cada optimización local es un problema de dos variables, éste es resuelto aplicando algunas iteraciones del método de Newton en dos dimensiones y haciendo uso de la matriz Hessiana y del vector gradiente calculados de forma exacta, por lo que sólo se hace uso de un vector y una matriz de *2x2* para la optimización local. Esta forma de optimización hace que el sistema pueda operar con pocos recursos. Hasta el momento, se han obtenido excelentes configuraciones de mallas de grandes dimensiones utilizando este sistema.

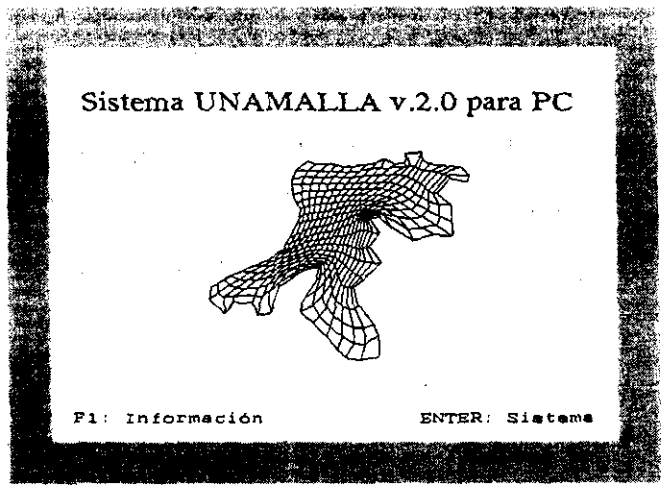
Se consideran dos modalidades de interacción con el sistema, el modo Manual y el modo Automático, en la modalidad automática, basta pulsar un par de teclas para obtener una malla óptima, a diferencia del modo manual en el que se puede experimentar con diversos parámetros y comparar las mallas obtenidas.

El grupo UNAMALLA está formado por profesores y estudiantes de posgrado de distintas Universidades del país, los cuales han venido presentando resultados de sus investigaciones en diferentes foros tanto nacionales como internacionales.

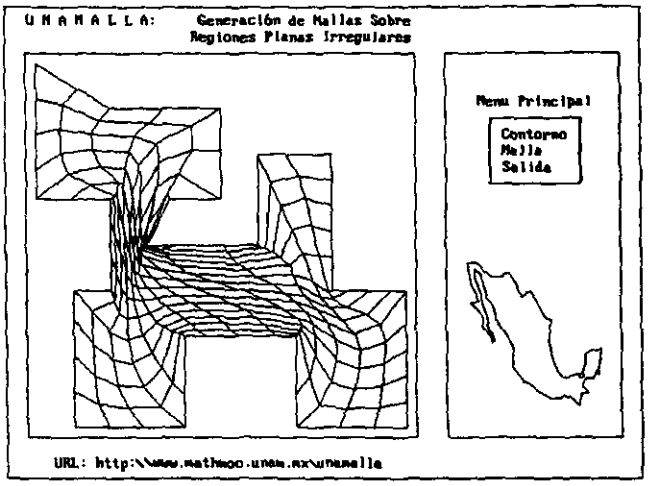
D2. Modo de Uso del Sistema

El presente manual está pensado como una ayuda primera al operador del sistema en el proceso de construir, generar y obtener una malla óptima bajo las características de los nuevos funcionales para la generación de mallas [52], para la estructura de los datos de entrada y salida puede consultar el Apéndice B.

Al ejecutar el programa `malla.exe` se observará la pantalla de presentación del sistema:



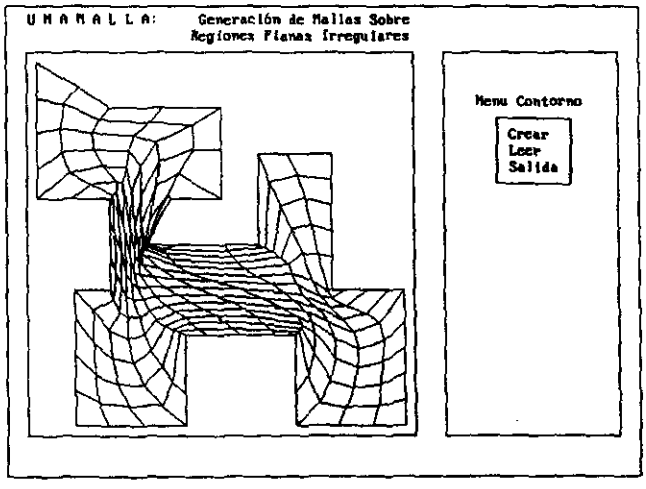
Al pulsar la tecla de función `F1` aparecerá la pantalla de información del sistema y con la tecla `ENTER` pasaremos a la pantalla principal del programa donde podremos elegir, usando las flechas del cursor, la opción a seguir.



Este es el menú principal. Aquí se presentan las opciones a los módulos de `Contorno` y `Malla`. En el módulo `Contorno` podemos generar o leer desde un archivo el contorno y pasar inmediatamente a generar una malla inicial sobre de él. En la opción del módulo `Malla` podemos leer una malla ya generada, guardar la malla actual del proceso, así como optimizarla. Pasemos ahora a describir cada uno de estos módulos.

D2.1 Menú de Contorno

De seleccionar la opción `Contorno` obtendremos la siguiente pantalla:

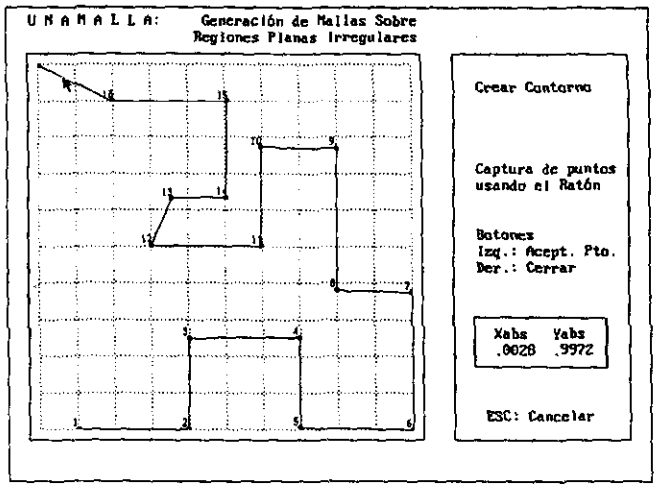


Aquí se cuenta con dos opciones, a saber: `Crear` y `Leer`. En la primera podemos generar un contorno de prueba y ser éste el empleado a lo largo de la ejecución del sistema, o bien, si la información de un contorno la tenemos ya en archivo podemos leer éste mediante la opción `Leer`.

Veamos las opciones que a su vez se presentan en cada una.

D2.1.1 Crear Contorno

En la opción `Crear` aparecerá la siguiente pantalla:



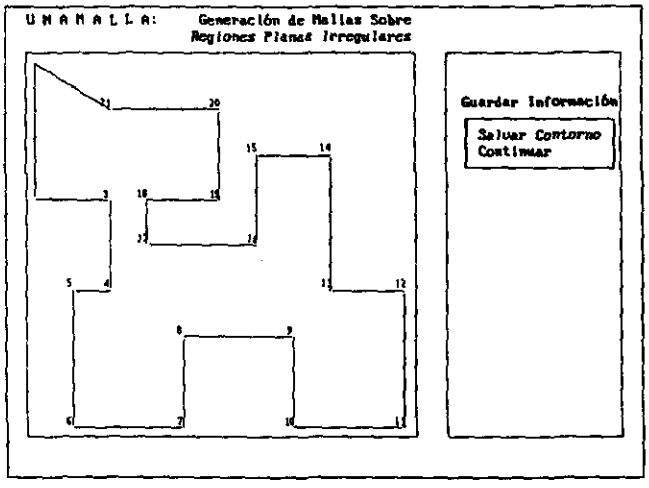
Aquí, con la ayuda del ratón, podremos generar una figura poligonal cerrada que será el contorno de la malla. Al presionar el botón de la izquierda del ratón se indicará que el punto donde éste esté situado será un punto del contorno. Los puntos se irán enumerando de acuerdo al orden de aparición. Una vez que se ha generado la figura poligonal deseada, presionando el botón derecho del ratón se cierra el polígono uniendo el último punto capturado con el primero.

Un contorno admisible para el sistema es aquel que cuando menos consta de cuatro puntos (para un triángulo debe considerarse al cuarto punto sobre alguno de sus lados), en caso contrario el contorno no es aceptable y se podrá observar sobre la pantalla el mensaje correspondiente. Si el contorno resulta en orientación contraria a las manecillas de reloj se desplegará un mensaje al respecto y se realizará el cambio del sentido de orientación del Contorno, ver Apéndice B.

En caso de ser aceptable el contorno se pasará al proceso de definir las cuatro fronteras de la malla, es decir, desde donde partirán las líneas para generar la malla.

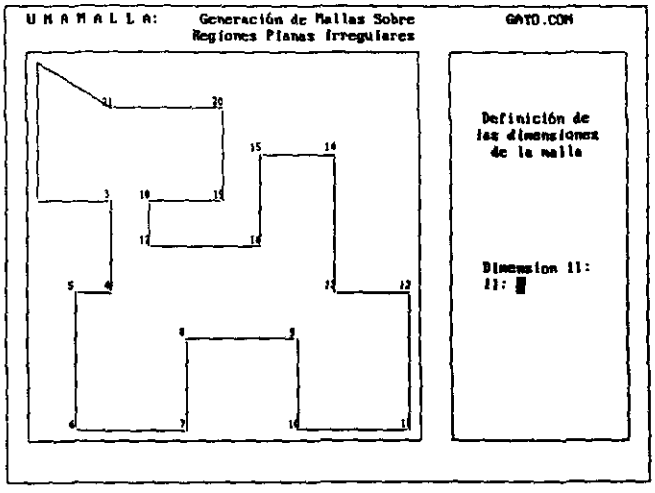
Para definir las cuatro componentes de la frontera de la región basta conocer la numeración de los puntos de los vértices donde inicie cada subfrontera. En la primera frontera, si no es el primer vértice el indicado, se hará una reenumeración de los vértices para considerar al vértice señalado como el vértice de inicio de la primera frontera. En caso de teclear números

incorrectos aparecerá en pantalla un mensaje de error. Si las fronteras están bien definidas, aparecerá enseguida el siguiente menú:



En este punto se tiene la opción de salvar el contorno generado, en cuyo caso se pedirá al usuario el nombre del archivo para guardar los datos, éstos serán del tipo ASCII y la información se salvará en un archivo con la extensión .CON, este archivo podrá ser leído posteriormente con la opción `Leer` en el `Menú de Contorno`. Si no desea salvar el contorno deberá elegir la opción `Continuar`.

Posterior a este paso de contorno aparecerá la pantalla



Aquí se genera la malla inicial con el contorno actual. Para ello se pide al usuario las dimensiones de la malla, **il** corresponde a las líneas que irán entre las fronteras pintadas de azul o líneas verticales de la malla genérica y **jl** a las fronteras amarillas o líneas horizontales

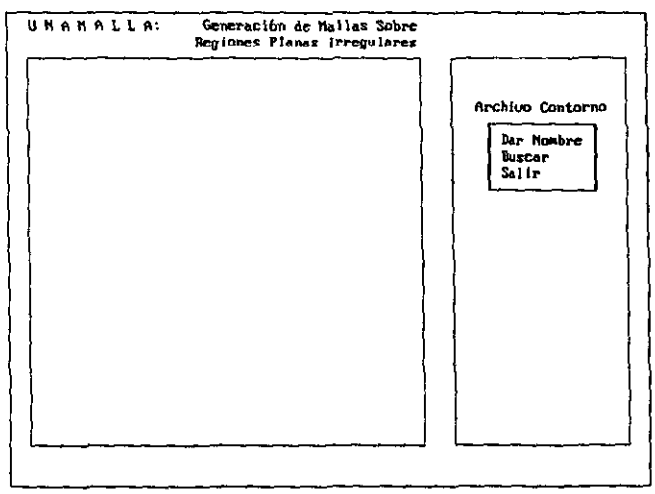
de la malla genérica. La única restricción para generar la malla será la memoria necesaria para ello, en caso de no poder manipular una malla de la dimensión requerida aparecerá el mensaje en pantalla: `Not enough memory` y se pedirá al usuario que intente con otras dimensiones.

Si las dimensiones son aceptables, aparecerá un mensaje de aceptación y enseguida se verificará la admisibilidad de la malla generada. Para que una malla sea admisible, es necesario que las cuatro celdas correspondientes a las esquinas de la malla sean convexas [4].

Si la malla no es admisible aparecerá el mensaje correspondiente para posteriormente, regresar al `Menú Principal` sin que se haya generado la malla. Si la malla es admisible, se regresa al `Menú Principal` con la malla generada.

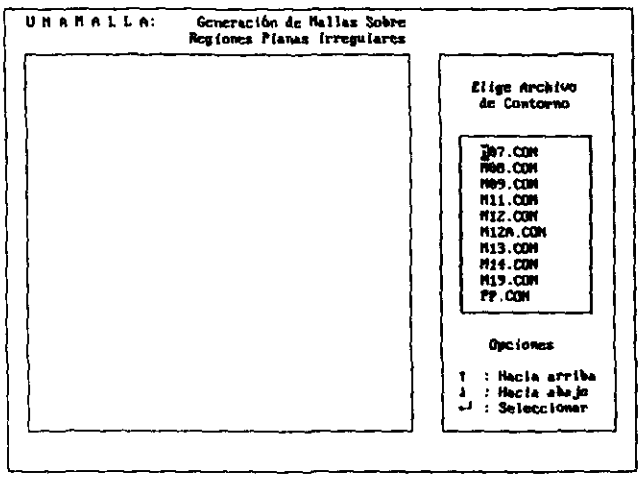
D2.1.2 Leer Contorno

Otra opción del `Menú de Contorno` es `Leer`, este es el caso cuando se cuenta con una lista de los archivos de contorno en el directorio actual que pueden ser utilizados. De elegir esta opción aparecerá la siguiente pantalla:

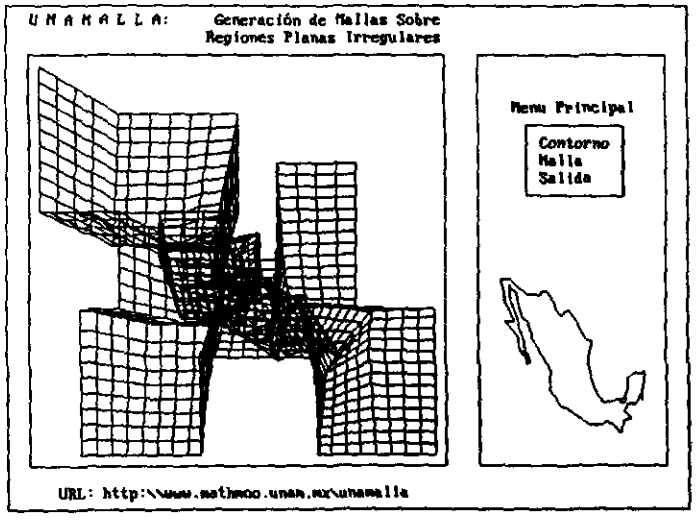


Se contemplan dos opciones: `Dar Nombre` y `Buscar`. En caso de que ya se conozca el archivo que se va a trabajar, se puede dar directamente el nombre del archivo con la opción `Dar Nombre`, el nombre del archivo se da sin extensión. Si el archivo no existe aparecerá el correspondiente mensaje y se regresa al `Menú de Contorno`.

La otra opción de lectura de contorno es `Buscar`, con esta opción aparecen en una ventana los nombres de los archivos de contorno disponibles donde debe seleccionarse el archivo deseado. La pantalla con la lista de archivos es la siguiente:



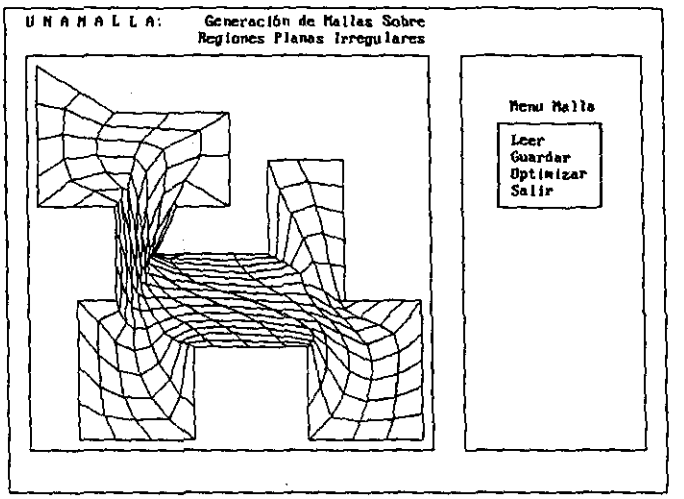
Una vez que se tiene el contorno ya generado enseguida se realiza un proceso similar a cuando se creó el contorno, se pedirán las dimensiones de la malla y se genera la malla inicial, verificando su admisibilidad.



Una vez hecho esto se regresa al Menú `Principal` con una malla ya generada.

D2.2 Menú Malla

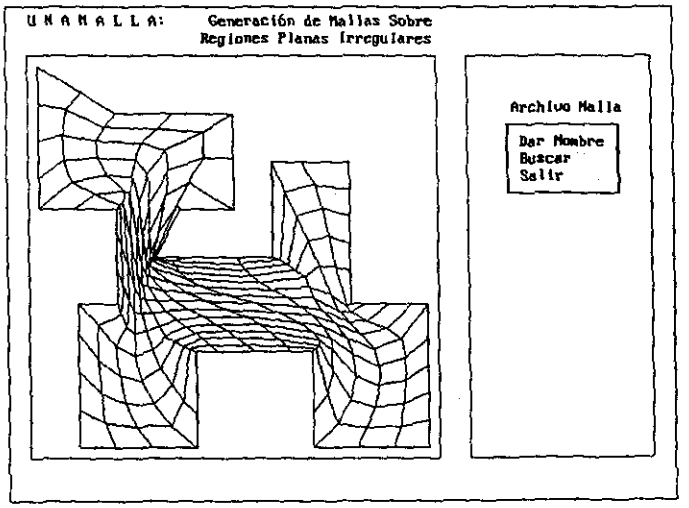
La otra opción del Menú `Principal` es `Malla`. Al seleccionar esta opción aparecerá la pantalla



donde se tienen las opciones: `Leer`, `Guardar` y `Optimizar`. Si en el estado actual del sistema existe una malla podemos manipularla sea para guardar la información o bien para optimizarla. Si hasta el momento no hay malla en memoria, las opciones `Guardar` y `Optimizar` no estarán disponibles.

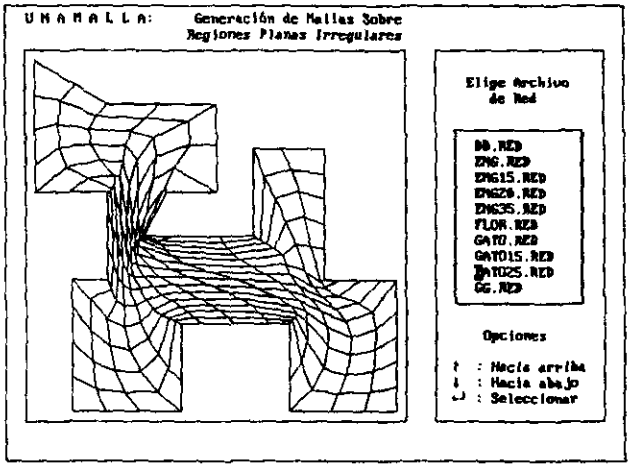
D2.2.1 Leer Malla

En la opción `Leer` aparece el siguiente menú



Aquí se leerá el archivo que contenga una malla. De igual forma que en el caso del módulo de contorno se tienen dos opciones: `Dar Nombre` y `Buscar`. En caso de conocer el nombre del archivo se puede seleccionar `Dar Nombre` e introducirlo directamente desde el teclado; se deberá dar el nombre del archivo sin extensión, el archivo deberá tener la extensión .RED en el directorio de trabajo. Enseguida aparecerá la gráfica de la malla generada con los datos del archivo dado.

Si no se tiene de antemano el nombre del archivo, lo puede buscar entre los archivos de mallas ya existentes con la opción `Buscar`, en este caso aparece la pantalla.



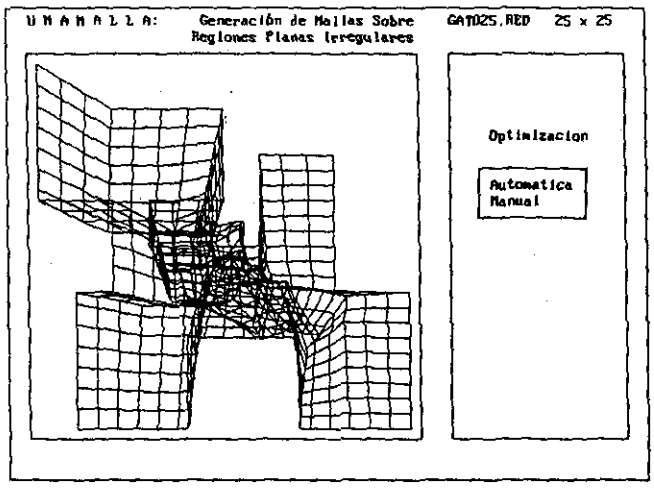
Aquí se escoge un archivo con una malla y enseguida aparecerá la gráfica de la malla contenida en el archivo elegido, si no es la malla buscada, puede volver a buscar otro archivo hasta que se tenga la malla deseada. Enseguida se regresa al `Menú de Mallas`, teniendo en memoria la malla sobre la que deseamos trabajar.

D2.2.2 Guardar Malla

Con la opción `Guardar` podemos salvar la malla que se tiene en este momento, se salvará con la extensión .RED en un archivo ASCII que contendrá las coordenadas de los nodos de la malla. La opción `Guardar` sólo se podrá utilizar si se cuenta con una malla en ese momento, de no ser así aparecerá un mensaje de error señalando esto.

D2.2.3 Optimizar Malla

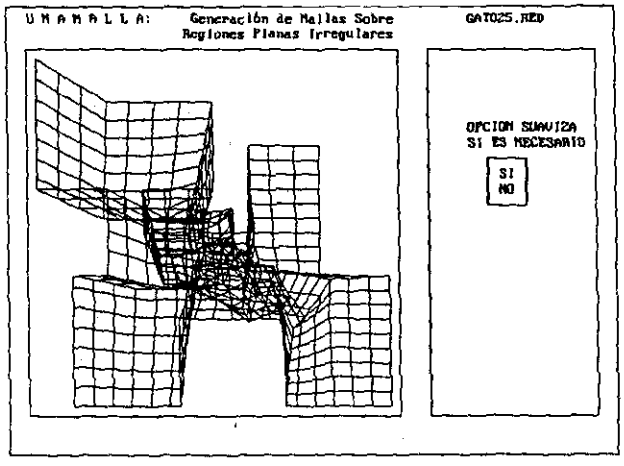
Otra opción disponible es `Optimizar`. En esta opción se procede a optimizar la malla actual; al igual que la opción `Guardar` sólo puede realizarse si se cuenta con la malla en memoria. Al solicitar esta opción aparecerá la siguiente pantalla.



Aquí se tienen dos opciones para realizar la optimización, en forma `automática` y en forma `manual`; éstas se refieren a la elección de los parámetros para realizar la optimización. Si se escoge en forma `manual` el sistema preguntará por cada uno de los parámetros necesarios para realizar la optimización; de elegir la que de forma `automática` se tomarán los parámetros por *default* inciando así el proceso de optimización.

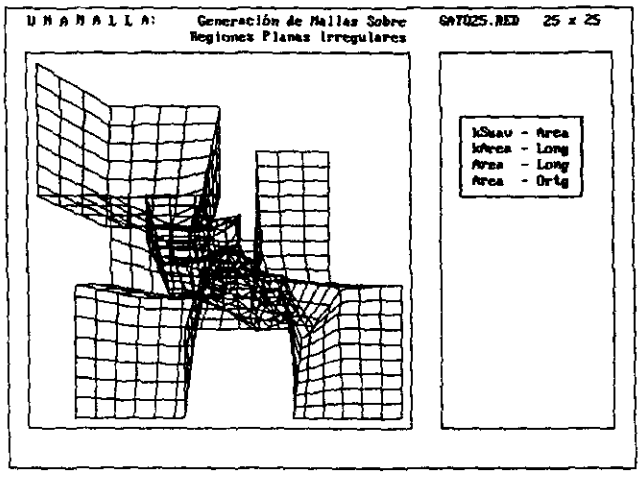
Optimización Manual

Al solicitar la opción manual se tendrá la siguiente pantalla.



Aquí se tiene la opción de un suavizamiento previo de la malla antes de realizar la optimización. Este proceso de suavizamiento se logra mediante el uso del funcional de longitud; la razón de realizar este suavizamiento previo lo sugiere nuestra experiencia al usar la Interpolación Transfinita en la generación de la malla inicial y en las distintas pruebas de convergencia de nuestros métodos con mallas iniciales muy perturbadas. El sistema contempla que el usuario pueda experimentar con esta opción de suavizamiento previo.

Enseguida aparece la siguiente pantalla.

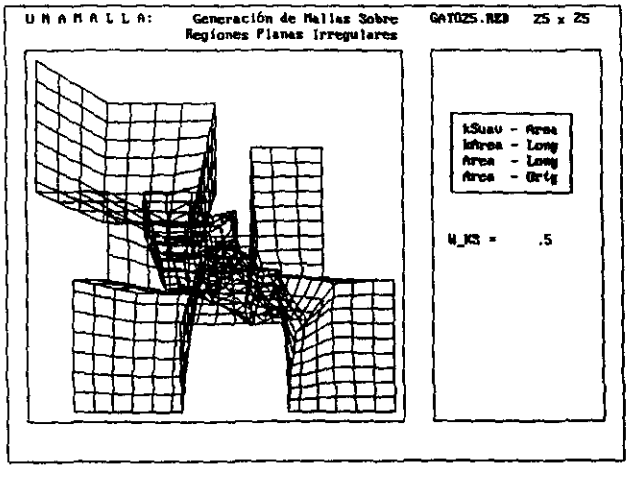


Aquí se selecciona el funcional a trabajar, se dispone de cuatro combinaciones entre los funcionales básicos de Area, k -Suavidad, Longitud, k-Area y Ortogonalidad.

w(k-Suavidad) + (1-w) Area
w(Area) + (1-w) Longitud
w(k-Area) + (1-w) Longitud
Area-Ortogonalidad

donde w representa el peso para cada funcional en la combinación convexa a trabajar.

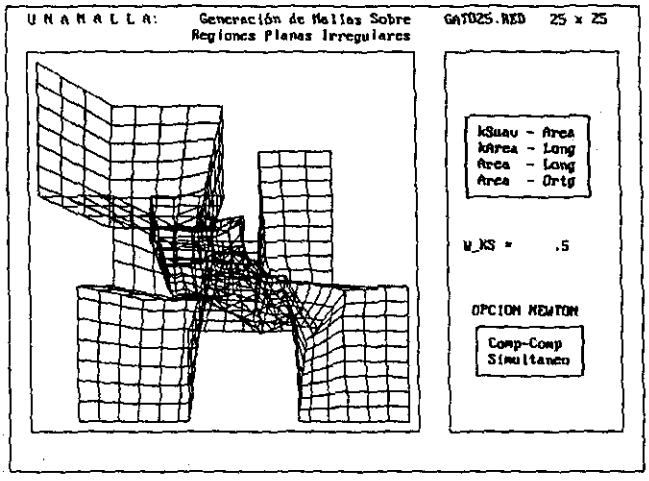
Una vez seleccionado el funcional aparece la pantalla.



Aquí es donde el usuario tiene que dar el peso asociado al primer funcional en la combinación convexa a trabajar, se tiene que

$$0 \leq w \leq 1.$$

Enseguida aparece la siguiente pantalla.



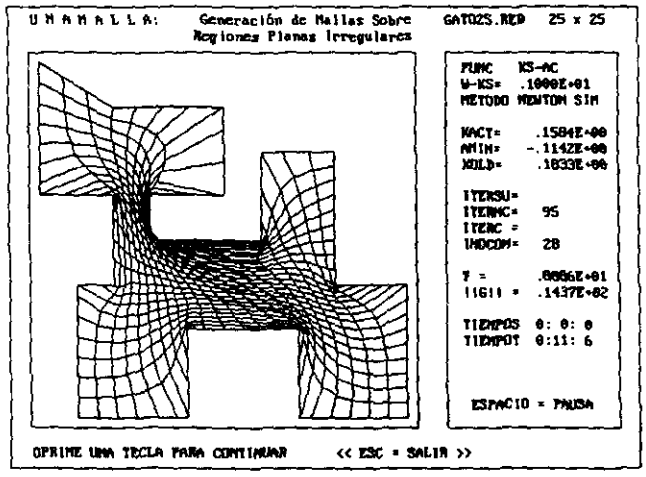
La optimización en este sistema se realiza de manera local, lo que se hace es considerar a cada nodo interior de forma independiente, esto es, se calculan sus nodos vecinos y se optimiza la submalla de 3x3 formada por éstos; de esta forma sólo estamos calculando las coordenadas del nodo interior; es decir, estamos resolviendo un problema de sólo dos variables. El proceso de optimización se realiza utilizando el método de Newton, con una sola iteración y se hace uso tanto de la matriz Hessiana y del gradiente del funcional calculados analíticamente. Se tienen dos opciones para resolver el sistema lineal, a saber, resolver el sistema `Componente a Componente` utilizando sólo la diagonal de la matriz, o resolver el sistema de forma `Simultánea`, en donde se utiliza toda la información de la Hessiana. En esta opción el usuario puede escoger la manera de resolver el sistema lineal.

Estos son todos los parámetros necesarios para realizar la optimización.

Optimización Automática

En caso de haber elegido el modo automático se consideran los parámetros por *default*, en este caso el funcional a optimizar es el de w(k-Suavidad) + (1-w)Area con un peso de w=0.5, no se considera la opción de suavizamiento y se resuelve el sistema lineal asociado al método de Newton de forma `Simultánea`.

En el ejemplo que hemos estado trabajando, `gato25.red`, escogiendo el modo automático para la optimización y al cabo de 95 iteraciones se tiene la siguiente gráfica.



Como se observa en la figura anterior, son desplegados varios parámetros asociados a la malla que se está optimizando, aquí debemos mencionar del proceso de homotopía que se emplea en el peso del funcional, siempre que se trabaje con la combinación que involucre k-Suavidad o k-Area. En estos casos siempre se inicia la optimización con un peso w=1 y de acuerdo a un criterio se inicia la homotopía y el peso empieza a aproximarse al peso indicado el inicio del proceso.

Los parámetros que se despliegan en el proceso de optimización son los siguientes:

FUNC: Tipo de Funcional que se está optimizando

W: Peso para el funcional en la iteración actual

METODO: Forma de resolver el sistema Lineal de Newton, ya sea `Componente a Componente` o `Simultáneo`.

KACT: Parámetro k actual para los funcionales de k-Suavidad y k-Area.

MINA: Alfa mínima de la malla actual.

KOLD: Parámetro k anterior.

ITERS: Número de iteraciones en el Suavizamiento previo.

ITERNC: Número de iteraciones cuando todavía existen celdas no convexas.

ITERC: Número de iteraciones cuando ya no existen celdas no convexas.

INOCON: Número de celdas no convexas de la malla actual.

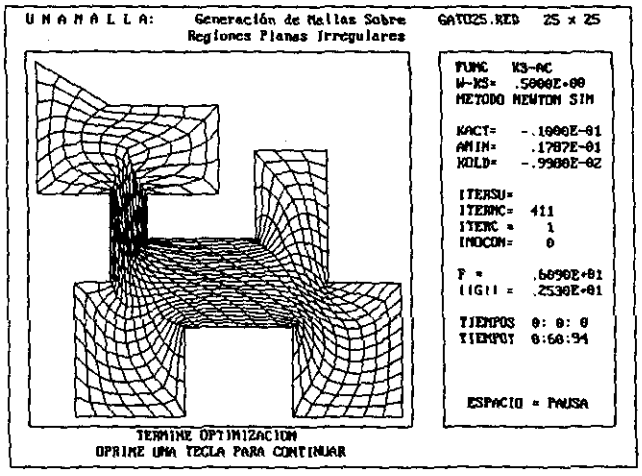
F: Valor total del funcional sobre toda la malla.

||G||: Norma del gradiente total del funcional sobre toda la malla.

TIEMPS: Tiempo empleado en el suavizamiento previo.

TIEMPT: Tiempo total empleado en la optimización.

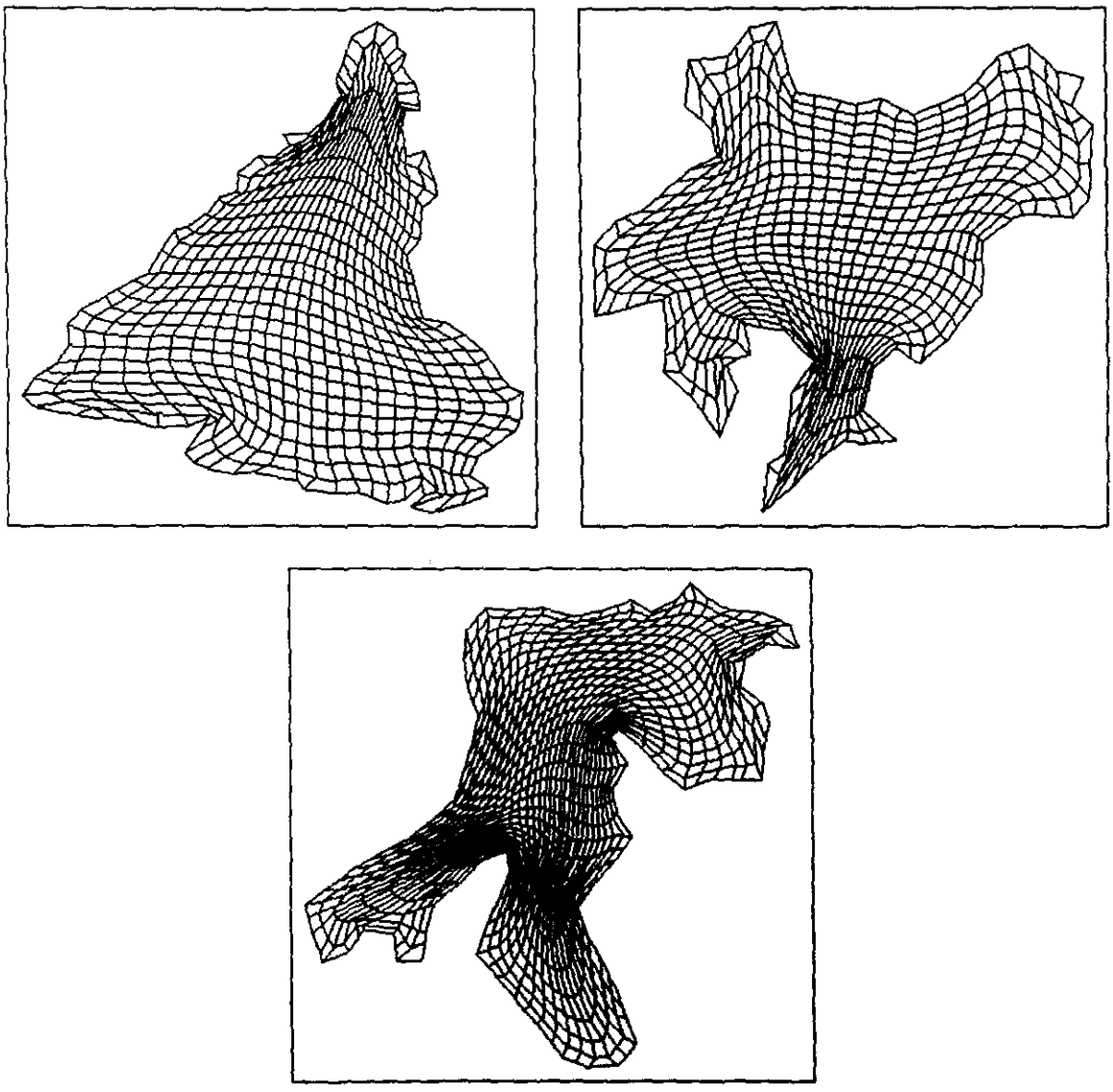
El usuario podrá interrumpir el proceso de optimización al pulsar la barra espaciadora y en su caso tendrá opción a continuar el proceso o a interrumpirlo definitivamente. Al final de la optimización se despliegan los parámetros actuales de la malla.



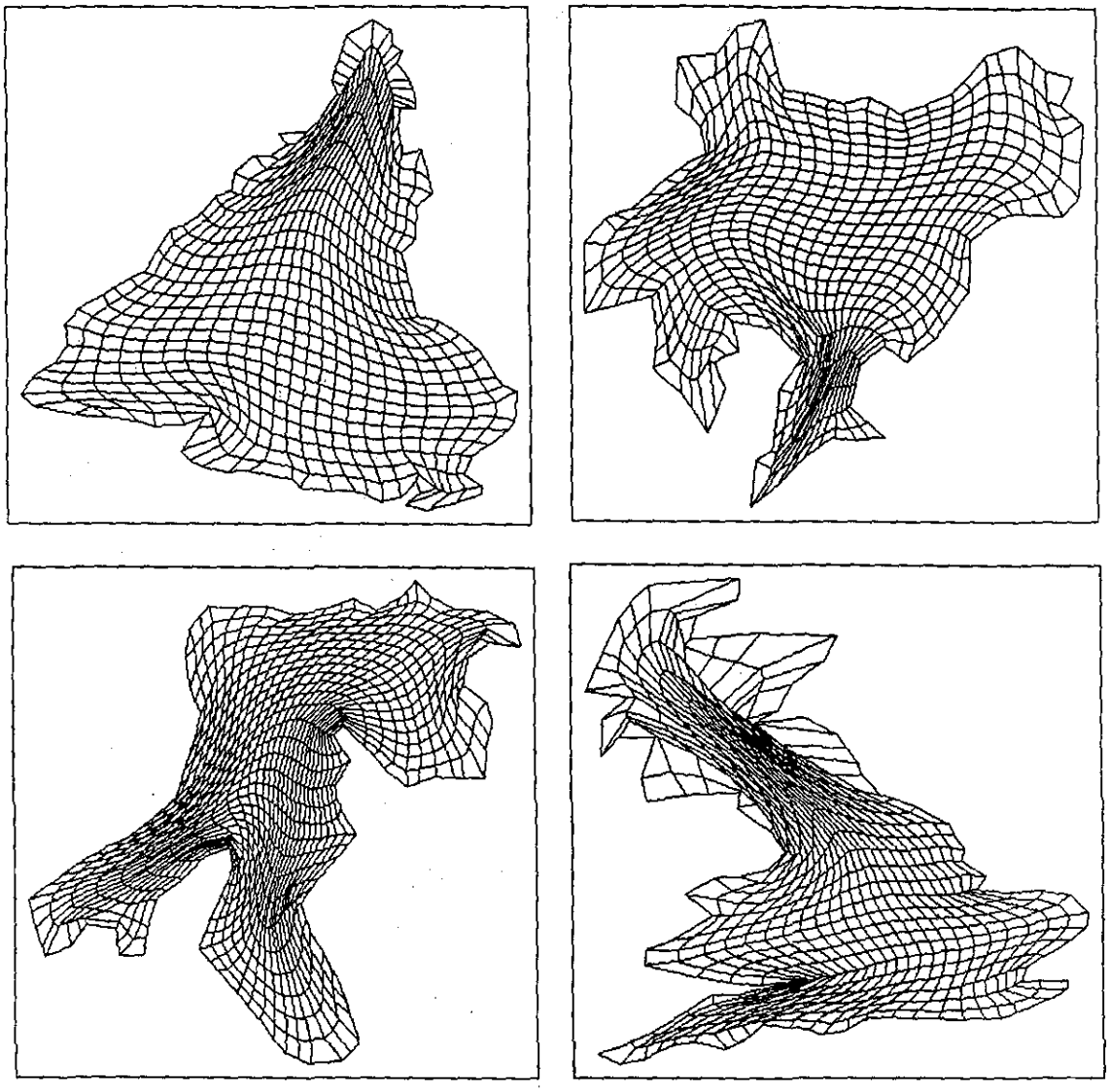
Una vez concluido el proceso de optimización pulsando una tecla se regresa al menú MALLA donde se puede guardar la información en un archivo o bien probar los resultados con otros funcionales.

D3. Galería de Mallas

Enseguida se presenta una galería de mallas generadas por el sistema empleando la combinación entre el funcional de k-Suavidad y Area Clásica con w=.25 y la combinación entre k-Area y Longitud con w=.75.



Mallas obtenidas con los funcionales de k-Suavidad y Area Clásica



Mallas obtenidas con los funcionales de k-Area y Longitud