OPTIMIZACIÓN NO–LINEAL (VER 1.10)

NOTAS

presenta

Dr. Félix Calderón Solorio

Universidad Michoacana de San Nicolás de Hidalgo

Septiembre 2012

Contenido

Contenido II
Lista de Figuras VII
Lista de Tablas IX
Lista de Algoritmos XI
Lista de Símbolos XIII
Lista de Tareas XV

1. Búsqueda uní–direccional de mínimos 1
1.1. Búsqueda exhaustiva 1
1.1.1. Ejemplo 2
1.2. Búsqueda de la sección dorada 4
1.2.1. Ejemplo 4
1.3. Método de Newton 6
1.3.1. Ejemplo 6

2. Búsqueda basadas en gradiente 9
2.1. Derivadas direccionales y el gradiente 9
2.2. Gradiente 10
2.3. Reconociendo un mínimo local 11
2.3.1. Condición necesaria de primer orden 12
2.3.2. Condición necesaria de segundo orden 13
2.3.3. Ejemplo 14
2.4. Dirección de búsqueda para métodos de búsqueda en línea 14
2.4.1. Curvas de Nivel y el vector Gradiente 15
2.5. Métodos de búsqueda en una dirección 17
2.5.1. Ejemplo 17
2.5.2. Ejemplo 19
2.6. Escalamiento 20
2.7. Las condiciones de Wolfe 23
2.7.1. Ejemplo 25
2.7.2. Ejemplo 25
2.7.3. Algoritmo de Suficiente decaimiento y muestreo hacia atrás 27
2.8. Algoritmos para seleccionar la longitud del paso 27
2.8.1. Máximo descenso 28

2.8.2. Interpolación . . . 28
2.8.3. Método de Newton . . . 29
2.8.4. Ejemplo . . . 30
2.9. Método de descenso de gradiente . . . 34
2.9.1. Ejemplo . . . 35
2.9.2. Convergencia del Método de descenso de gradiente . . . 36
2.9.3. Mejora del Método de descenso de gradiente . . . 37
2.9.4. Ejemplo . . . 39
2.10. Descenso de gradiente con búsqueda lineal basada en condiciones de Wolfe . 41
2.10.1. Ejemplo . . . 42

3. Método del gradiente conjugado 45
3.1. Método de direcciones conjugadas . . . 45
3.2. El método del gradiente conjugado para funciones cuadráticas . . . 47
3.2.1. Algoritmo de GC . . . 50
3.2.2. Ejemplo . . . 52
3.3. Precondicionamiento . . . 53
3.3.1. Precondicionadores Prácticos . . . 57
3.3.2. Ejemplo . . . 59
3.4. Manejo de Dispersidad . . . 62
3.4.1. Sustitución hacia adelante . . . 64
3.4.2. Sustitución hacia atras . . . 65
3.4.3. Ejemplo de Filtrado de Imágenes . . . 66
3.5. Gradiente conjugado no lineal . . . 67
3.5.1. El método de Polak - Ribiere . . . 69
3.5.2. Ejemplo . . . 69

4. Métodos de Newton 71
4.1. Método de Newton . . . 71
4.1.1. Ejemplo . . . 72
4.1.2. Ejemplo . . . 73
4.2. Problemas de convergencia del Método de Newton . . . 73
4.2.1. Ejemplo . . . 73
4.2.2. Ejemplo . . . 75
4.3. Método de Newton Modificado . . . 75
4.4. Método de Broyden's o secante . . . 76
4.4.1. Algoritmo . . . 78
4.4.2. Ejemplo . . . 79
4.5. Método de Secante con actualización BFGS . . . 80
4.6. Mínimos cuadrados no lineales . . . 81
4.6.1. Método de Gauss-Newton . . . 82
4.6.2. Método de Levenberg-Marquardt . . . 82

5. Algoritmos Genéticos 85
5.1. Búsqueda aleatoria . . . 85
5.2. Problema del Agente Viajero . . . 87
5.2.1. Simulated Annealing . . . 87
5.3. Algoritmos Genéticos . . . 89
5.4. Algoritmos Genéticos con codificación binaria . . . 90
5.4.1. Representación de parámetros . . . 90
5.4.2. Población inicial . . . 91
5.4.3. Ejemplo . . . 92
5.4.4. Apareamiento . . . 93
5.4.5. Mutación . . . 95
5.5. AG con codificación Real . . . 96
5.5.1. Población Inicial . . . 96
5.5.2. Apareamiento . . . 97
5.5.3. Cruzamiento . . . 97
5.5.4. Mutación . . . 98

6. Optimización con Restricciones 99
6.1. Teoría de la optimización con restricciones . . . 99
6.2. Una simple restricción de desigualdad . . . 100
6.3. Multiplicadores de Lagrange . . . 104
6.3.1. Ejemplo . . . 105
6.4. Restricciones de desigualdad . . . 106
6.5. Condiciones Necesarias de Primer Orden . . . 107
6.6. Programación Cuadrática . . . 108
6.6.1. Restricciones de igualdad QP . . . 108
6.7. Conjunto Activo . . . 110
6.7.1. Algoritmo Conjunto Activo . . . 112
6.7.2. Ejemplos . . . 113
6.7.3. ejemplo dos . . . 113
6.7.4. Ejercicio (del libro de Jorge Nocedal) . . . 116
6.7.5. Ejercicio (del libro de Jorge Nocedal) . . . 118
6.8. El método de Barrera Logaritmica . . . 121
6.8.1. Calculo de Gradiente y Hessiano con Restricciones Lineales . . . 123
6.8.2. Ejemplo . . . 125
6.8.3. Método . . . 125
6.8.4. Ejemplo . . . 127
6.9. Métodos de Punto Interior. . . . 127
6.9.1. Ejemplo . . . 130

A. Algoritmo de factorización de Cholesky 133
A.1. Factorización de Cholesky . . . 133
A.1.1. Ejemplo . . . 133
A.2. Algoritmo . . . 135
A.3. Algoritmo de Cholesky incompleto . . . 138

B. Ejemplos 141
B.1. Resultados 141
B.2. Ejemplo 2 144
B.3. Ejemplo 145

Lista de Figuras

1.1. Función $-2sen(x) + x^2/10$ 2

2.1. Curvas de nivel para la función de Rosenbrock 16
2.2. Función $f(x, y) = e^{-x^2-y^2}x$ 18
2.3. Función $\phi(X_0)$ 19
2.4. Función $f(x) = x_1^2 + x_2^2$ 21
2.5. Curvas de nivel de la función $f(x) = x_1^2 + x_2^2$ 22
2.6. Curvas de nivel de la función $f(x) = 2x_1^2 + x_2^2$ 22
2.7. Condición de decrecimiento suficiente 23
2.8. Condición de curvatura 24
2.9. Condiciones de wolfe 24
2.10. Condiciones de Wolfe 26
2.11. Condiciones fuertes de wolfe 26
2.12. Función $\phi(\alpha)$ para la función $xe^{-x^2-y^2}$, en la dirección u_{30} 34
2.13. Direcciones de búsqueda para descenso de gradiente 36
2.14. Efecto de zigzag creado por el método de descenso de gradiente 37
2.15. Algoritmo de Forsyte de dos pasos 39
2.16. Comportamiento de Maximo descenso y Maximo descenso con condiciones de wolfe 43
2.17. Comportamiento de Maximo descenso y Maximo descenso con condiciones de wolfe 44

3.1. Ejemplo de como trabajan las direcciones conjugadas 47
3.2. Comparación entre el método de GC y GC precondicionado. 62
3.3. Comparación entre el método de GC y GC precondicionado para el caso del filtrado de imagenes. 66

4.1. Función $f(x) = x_1e^{-x_1^2-x_2^2}$. 74
4.2. Gradiente con una aproximación de secantes 77

5.1. Problema del agente Viajero 87
5.2. función $f(x) = x_1seno(4x_1) + 1.1x_2seno(2x_2)$ 92
5.3. Solución usando AG para la función $f(x) = x_1seno(4x_1) + 1.1x_2seno(2x_2)$. 96
5.4. Solución usando AG para la función $f(x) = x_1seno(4x_1) + 1.1x_2seno(2x_2)$. 98

6.1. Función lineal con restricción cuadrática 100
6.2. Función cuadrática con restricción lineal 102
6.3. Función cuadrática con restricción cuadrática 103
6.4. Solución del ***Método del Conjunto Activo*** para $q(x) = (x_1 - 2)^2 + (x_2 - 2)^2$; $x^{(0)} = [2, 4]^T$. 114
6.5. Solución del ***Método del Conjunto Activo*** para $q(X) = (x_1 - 2)^2 + (x_2 - 2)^2$; $x^{(0)} = [4, 3]^T$. 115
6.6. Solución del ***Método del Conjunto Activo*** para $q(X) = (x_1 - 1)^2 + (x_2 - 2.5)^2$; $X_0 = [2, 0]^T$. 122
6.7. Función lineal con diferentes valores de μ 126

Lista de Tablas

1.1. solución para la función $f(x) = -2seno(x) + x^2/10$ 3
1.2. solución para la función $f(x) = -2seno(x) + x^2/10$ utilizando razón dorada 5
1.3. solución para la función $f(x) = -2seno(x) + x^2/10$ utilizando el método de Newton 7

2.1. solución para la función $f(x) = xe^{(-x^2-y^2)}$ utilizando descenso de gradiente 40
2.2. solución para la función $f(x) = xe^{(-x^2-y^2)}$ utilizando descenso de gradiente utilizando la mejora de Forsyte–Luenberger 40
2.3. Comparacion de iteraciones entre tamano de paso α de MD y Wolfe para minimizar $f(x)$ 42
2.4. Comparacion de iteraciones entre tamano de paso utilizando α de MD y Wolfe para minimizar $-f(x)$ 43

3.1. Comparación entre el gradiente Conjugado y el Gradiente Conjugado Precondicionado 61

6.1. Resultados del ***Método del Conjunto Activo*** para $q(x) = (x_1-2)^2+(x_2-2)^2$; $x_0 = [2,4]^T$. 114
6.2. Resultados del ***Método del Conjunto Activo*** para $q(x) = (x_1-2)^2+(x_2-2)^2$; $x^{(0)} = [4,3]^T$. 115
6.3. Resultados del ***Método del Conjunto Activo*** para $q(X) = (x_1-1)^2 + (x_2-2.5)^2$; $X_0 = [2,0]^T$. 122
6.4. Mínimo de la función 128

Lista de Algoritmos

1. Algoritmo de Búsqueda exhaustiva 2
2. Algoritmo de Búsqueda con Razón Dorada 5
3. Algoritmo de Newton Unidimensional 7
4. Algoritmo de Búsqueda en la dirección del gradiente 18
5. Búsqueda-lineal-con muestreo-hacia-atras 27
6. Algoritmo de Newton Unidimensional 31
7. Método de descenso de Gradiente 35
8. Mejora del Método de descenso de Gradiente 38
9. Algoritmo de Búsqueda lineal basado en condiciones de Wolfe 41
10. Algoritmo de Gradiente Conjugado preliminar 48
11. Algoritmo de Gradiente Conjugado 51
12. Algoritmo de Gradiente Conjugado Precondicionado 56
13. Algoritmo de Gradiente Conjugado no lineal 68
14. Algoritmo de Newton 72
15. Algoritmo de Broyden 78
16. Algoritmo BFGS 80
17. Algoritmo de Gauss Newton 82
18. Algoritmo de Levenberg-Marquardt 83
19. Método de Barrera 126

Lista de Símbolos

AG	Algoritmos Genéticos
F	Conjunto de funciones
T	Conjunto de Terminales
PG	Programación Genética
GEP	Gene Expression Programming

Lista de Tareas

Tarea 1 6
Tarea 2 17
Tarea 3 20
Tarea 4 27
Tarea 5 37
Tarea 6 40
Tarea 7 67
Tarea 8 ..... **??**
Tarea 9 ..... **??**

Capítulo 1

Búsqueda uní–direccional de mínimos

Dada una función $f : \mathbb{R} \to \mathbb{R}$, deseamos calcular el mínimo de esta función. Una manera de hacerlo es hacer una búsqueda en una dirección (normalmente en la que decrece la función) utilizando una heurística, que puede ir desde la búsqueda exhaustiva, búsqueda de razón dorada, etc.

1.1. Búsqueda exhaustiva

Una manera de realizar esta tarea es proponer una sucesión dada por la ecuación 1.1

$$x_{k+1} = x_k + h \tag{1.1}$$

donde:h es un incremento con el cual nos movemos, de manera lo suficientemente lenta, para no pasar sobre el mínimo. El valor de h es positivo cuando tenemos evidencia que el mínimo esta a la derecha y será negativa cuando el mínimo esta a la izquierda (ver [**?**]).

El algoritmo de búsqueda, para un mínimo será el algoritmo 1:

Algoritmo 1 Algoritmo de Búsqueda exhaustiva

BUSQUEDAEXAUSTIVA$(x_0, h, f(x))$

1 **mientras** $f(x_{k+1}) < f(x_k)$
2 $\quad x_{k+1} \leftarrow x_k + h$
3 $\quad k \leftarrow k + 1$
4 $\quad$ **regresar** x_{k+1}

1.1.1. Ejemplo

Utilizando la búsqueda exhaustiva, encuentre el mínimo de la función $f(x) = -2seno(x) + x^2/10$ en el intervalo[0,4]. En la figura 1.1 se muestra esta ecuación en el intervalo mencionado.

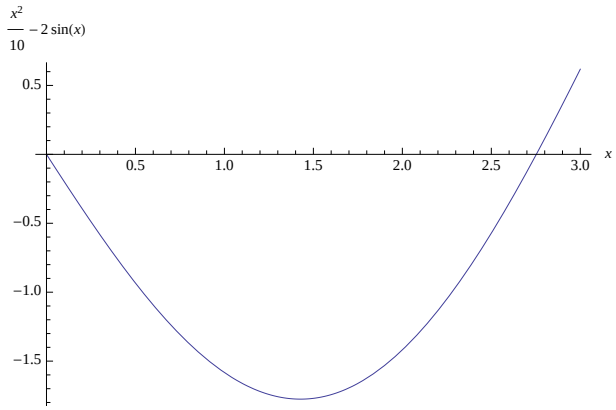


Figura 1.1: Función $-2sen(x) + x^2/10$

Para un incremento de búsqueda $h = 0.1$, de acuerdo con la tabla 1.1, el máximo es: $x = 1.4$. Note que la precisión de la solución esta limitada por el incremento y si se desea mayor precisión se necesitarán de un incremento más pequeño.

x	**f(x)**
0.0	0.
0.1	-0.1986
0.2	-0.3933
0.3	-0.5820
0.4	-0.7628
0.5	-0.9338
0.6	-1.0932
0.7	-1.2394
0.8	-1.3707
0.9	-1.4856
1.0	-1.5829
1.1	-1.6614
1.2	-1.7200
1.3	-1.7581
1.4	**-1.7749**
1.5	-1.7699
1.6	-1.7431
1.7	-1.6943
1.8	-1.6237
1.9	-1.5316

Tabla 1.1: solución para la función $f(x) = -2seno(x) + x^2/10$

1.2. Búsqueda de la sección dorada

La búsqueda de la sección dorada es una técnica simple de búsqueda de una sola variable de propósito general. La clave para hacer eficiente este procedimiento es la mejor elección de los puntos intermedios. Esta meta se puede alcanzar al especificar que las siguientes dos condiciones se cumplan (ver [**?**]).

$$l_0 = l_1 + l_2 \tag{1.2}$$

$$\frac{l_1}{l_0} = \frac{l_2}{l_1} \tag{1.3}$$

Sustituyendo la ecuación 1.2 en la ecuación 1.3 obtenemos:

$$\frac{l_1}{l_1 + l_2} = \frac{l_2}{l_1} \tag{1.4}$$

Definimos $R = \frac{l_2}{l_1}$ y sustituimos en la ecuación 1.4 para obtener $R^2 + R - 1$, cuya solución positiva es $R = \frac{\sqrt{5}-1}{2}$. R es definido como la razón dorada y fue un número ampliamente utilizado por los Griegos en su arquitectura.

El algoritmo de la razón dorada es:

1.2.1. Ejemplo

Use la búsqueda de la sección dorada para encontrar el mínimo de la función $f(x) = -2seno(x) + \frac{x^2}{10}$ en el intervalo [0,4].

Los resultados de este ejemplo se muestran en la tabla 1.2 y se puede observar que se logra mejor precisión que con razón dorada en menos iteraciones.

Según los resultados de la tabla 1.2, la solución es $x = 1.4267$ con $f(x) = -1.7757$.

Tarea 1

Comparación de Búsqueda exhaustiva y Razón Dorada

Dada la función $f(x) = x^2 - seno(x)$, determinar:

- Su gráfica en el intervalo $[0, 1]$ y verificar si el punto de inflexion es un mínimo o un máximo,

Algoritmo 2 Algoritmo de Búsqueda con Razón Dorada

BúsquedaSecciónDorada$(x_l, x_u, f(x), Tol)$

```
1  mientras x_u − x_l > Tol
2    {
3    d ← R(x_u − x_l)
4    x_1 ← x_l + d
5    x_2 ← x_u − d
6    k ← k + 1
7    si f(x_1) < f(x_2)
8      entonces
9                x_l ← x_2
10     sino     x_u ← x_1
11   }
12   si f(x_1) < f(x_2)
13     entonces regresar x_1
14     sino     regresar x_2
```

k	x_l	f_l	x_2	f_2	x_1	f_1	x_u	f_u	d
0	0.0000	0.0000	1.5279	-1.7647	2.4721	-0.6300	4.0000	-3.1136	2.4721
1	0.0000	0.0000	0.9443	-1.5310	1.5279	-1.7647	2.4721	-0.6300	1.5279
2	0.9443	-1.5310	1.5279	-1.7647	1.8885	-1.5432	2.4721	-0.6300	0.9443
3	0.9443	-1.5310	1.3050	-1.7595	1.5279	-1.7647	1.8885	-1.5432	0.5836
4	1.3050	-1.7595	1.5279	-1.7647	1.6656	-1.7136	1.8885	-1.5432	0.3607
...	...	...	...	...	...	...	...	...	...
17	1.4267	-1.7757	1.4271	-1.7757	1.4274	-1.7757	1.4278	-1.7757	0.0007

Tabla 1.2: solución para la función $f(x) = -2seno(x) + x^2/10$ utilizando razón dorada

- Buscar el punto de inflexión utilizando búsqueda exhaustiva y
- Razón dorada calcular su mínimo
- Comparar los resultados

1.3. Método de Newton

Consideremos que la función $f : \mathbb{R} \to \mathbb{R}$ es de clase dos, es decir que la segunda derivada puede ser calculada. La idea consiste en reemplazar en la vecindad del punto x^k de la función f por una aproximación cuadrática $q(x)$ dada por

$$q(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k)^2$$

llamaremos a x_{k+1} el mínimo de $q(x)$. Para calcularlo, es necesario que la matriz $f''(x_k)$ sea positivo. La función $q(x)$ es entonces estrictamente convexa y tiene un mínimo único en x_{k+1} dado por

$$q'(x_{k+1}) = 0$$

Esto da lugar sistema lineal de ecuaciones:

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

La solución para x_{k+1} es el mínimo, lo cual da lugar a la recurrencia

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \tag{1.5}$$

1.3.1. Ejemplo

Calcular el máximo de la función $f(x) = -2seno(x) + \frac{x^2}{10}$ utilizando el método de Newton con un valor inicial $x_0 = 0$.

Para resolver tenemos que calcular $f'(x)$ y $f''(x)$

$$f'(x) = \frac{x}{5} - 2cos(x)$$

$$f''(x) = \frac{1}{5} + 2sin(x)$$

Algoritmo 3 Algoritmo de Newton Unidimensional

NEWTON UNIDIMENSIONAL$(f(x), x_0)$

1 `Dado una función diferenciable dos veces` $f(x)$ `y un valor inicial` x_0
2 **repetir**
3 $\quad x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$
4 $\quad k = k + 1$
5 **hasta** $\|\nabla f(x_k)\| \leq \varepsilon$
6 **regresar** x_k

x_k	$f(x_k)$	$f'(x_k)$	$f''(x_k)$	x_{k+1}
0.5000	-0.9338	-1.6551	1.1588	1.9282
1.9282	-1.5017	1.0854	2.0735	1.4047
1.4047	-1.7751	-0.0495	2.1725	1.4275
1.4275	-1.7757	8.20126E-05	2.1795	1.4275
1.4275	-1.7757	2.02094E-10	2.1795	1.4275

Tabla 1.3: solución para la función $f(x) = -2seno(x) + x^2/10$ utilizando el método de Newton

Capítulo 2

Búsqueda basadas en gradiente

2.1. Derivadas direccionales y el gradiente

Sea una función $f : \mathbb{R}^2 \to \mathbb{R}$ de dos variables x y y y sea $P(x, y)$ un punto en el plano x,y. Definimos u como el vector unitario que forma un ángulo de θ radianes con el eje x. Entonces:

$$u = cos\theta\hat{i} + \sin\theta\hat{j} \tag{2.1}$$

La derivada direccional de f los podemos definir como:

$$\mathrm{D}_u f(x, y) = \lim_{h \to 0} \frac{f(x + h\cos\theta, y + h\sin\theta) - f(x, y)}{h}$$

La derivada direccional de la razón de cambio de los valores de la función $f(x, y)$ con respecto a la distancia en el plano x, y, medida en la dirección del vector unitario u.

Si queremos calcular la derivada en la dirección de x hacemos $u = \hat{i}$ con $\theta = 0$

$$\mathrm{D}_{\hat{i}} f(x, y) = \lim_{h \to o} \frac{f(x + h, y) - f(x, y)}{h}$$

Ahora si queremos valuar la deriva en la dirección y hacemos $u = \hat{j}$

$$\mathrm{D}_{\hat{j}} f(x, y) = \lim_{h \to 0} \frac{f(x, y + h) - f(x, y)}{h}$$

Ahora obtendremos una fórmula que nos permita calcular una derivada direccional en una forma más corta.

$$g(t) = f(x + t\cos\theta, y + t\sin\theta)$$

y para un vector unitario $u = \cos\theta\hat{i} + \sin\theta\hat{j}$. Entonces por definición de una derivada ordinaria tenemos:

$$\begin{aligned} g'(0) &= \lim_{h\to 0} \frac{f(x+(0+h)\cos\theta, y+(0+h)\sin\theta)}{h} - \\ &\quad \frac{f(x+0\cos\theta, y+0\sin\theta)}{h} \\ &= \lim_{h\to 0} \frac{f(x+h\cos\theta, y+h\sin\theta) - f(x,y)}{h} \\ g'(0) &= \mathrm{D}_u f(x,y) \end{aligned}$$

Calculamos $g'(t)$ aplicando la regla de la cadena:

$$\begin{aligned} g'(t) &= f_x(x + t\cos\theta, y + t\sin\theta)\frac{\partial(x+t\cos\theta)}{\partial t} \\ &\quad + f_y(x + t\cos\theta, y + t\sin\theta)\frac{\partial(y+t\sin\theta)}{\partial t} \end{aligned}$$

Por lo tanto:

$$g'(0) = f_x(x,y)cos\theta + f_y(x,y)\sin\theta$$

En general

$$\mathrm{D}_u f(x,y) = f_x(x,y)\cos\theta + f_y(x,y)\sin\theta$$

Para detalles ver [**?**]

2.2. Gradiente

Si f es una función de dos variables x y y, y f_x y f_y existen, entonces el gradiente de f denotado por ∇f está definido por:

$$\nabla f(x,y) = f_x(x,y)\hat{i} + f_y(x,y)\hat{j}$$

Dado que la derivada direccional puede escribirse como:

$$\mathrm{D}_u f(x,y) = \left(\cos\theta\hat{i} + \sin\theta\hat{j}\right)\left[f_x(x,y)\hat{i} + f_y(x,y)\hat{j}\right]$$

entonces:

$$\mathrm{D}_u f(x,y) = u \cdot \nabla f(x,y)$$

2.3. Reconociendo un mínimo local

Una herramienta utilizada para estudiar mínimos es la serie de Taylor, la cual puede ser descrita por:

$$\begin{aligned} f(x) &= f(a) + f'(a)(x-a) + \tfrac{1}{2!}f''(a)(x-a)^2 \\ &\quad \tfrac{1}{3!}f'''(a)(x-a)^3 + \tfrac{1}{4!}f^{\mathrm{IV}}(a)(x-a)^4 + \ldots \end{aligned}$$

La serie de Taylor es una aproximación polinomial dada por

$$f(x+p) = c_0 + c_1 p + c_2 p^2 + c_3 p^3 + c_4 p^4 + c_5 p^5 + \cdots$$

donde x es un punto y p es un incremento de x en alguna dirección. Si calculamos las derivadas de $f(x+p)$ tenemos

$$\begin{aligned} f^{i}(x+p) &= c_1 + 2c_2 p + 3c_3 p^2 + 4c_4 p^3 + 5c_5 p^4 + \cdots \\ f^{ii}(x+p) &= 2c_2 + 2 \times 3c_3 p + 3 \times 4c_4 p^2 + 4 \times 5c_5 p^3 + \cdots \\ f^{iii}(x+p) &= 2 \times 3c_3 + 2 \times 3 \times 4c_4 p + 3 \times 4 \times 5c_5 p^2 + \cdots \\ f^{iv}(x+p) &= 2 \times 3 \times 4c_4 + 2 \times 3 \times 4 \times 5c_5 p + \cdots \\ f^{v}(x+p) &= 2 \times 3 \times 4 \times 5 \times c_5 + \cdots \end{aligned}$$

Si evaluamos las ecuaciones anteriores en $p = 0$

$$\begin{aligned} f(x+0) &= c_0 \\ f^i(x+0) &= c_1 \\ f^{ii}(x+0) &= 2c_2 \\ f^{iii}(x+0) &= 2 \times 3c_3 \\ f^{iv}(x+0) &= 2 \times 3 \times 4c_4 \\ f^{v}(x+0) &= 2 \times 3 \times 4 \times 5 \times c_5 \end{aligned}$$

Con esto temos el conjunto de coeficientes c_i para la serie de Taylor

$$\begin{aligned} c_0 &= f(x) \\ c_1 &= f^i(x) \\ c_2 &= \frac{f^{ii}(x)}{2!} \\ c_3 &= \frac{f^{iii}(x)}{3!} \\ c_4 &= \frac{f^{iv}(x)}{4!} \\ c_5 &= \frac{f^{v}(x)}{5!} \end{aligned}$$

y una formula cerrada para la serie de Taylor dada como:

$$f(x+p) = f(x) + f^i(x)p + \frac{f^{ii}(x)}{2!}p^2 + \frac{f^{iii}(x)}{3!}p^3 + \cdots \tag{2.2}$$

En el caso de una función de varias variables:

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

Existen condiciones necesarias para determinar un mínimo, estas están relacionadas con el gradiente ∇f y el Hessiano $\nabla^2 f$.

2.3.1. Condición necesaria de primer orden

Si x^* es un mínimo local y f es diferenciable en una vecindad de x^* entonces $\nabla f(x^*) = 0$.

Prueba

Supongamos por contradicción que $\nabla f(x^*) \neq 0$. Definimos el vector $p = -\nabla f(x^*)$ y notamos que:

$$p^T \nabla f(x^*) = - \parallel \nabla f(x^*) \parallel^2 < 0$$

Dado que ∇f es continuo cerca de x^*, existe un escalar $T > 0$ tal que:

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

Para algún $\bar{t} \in [0, T]$, tenemos que la serie de Taylor nos da:

$$f(x^* + \bar{t}p) = f(x^*) + \bar{t}p^T \nabla f(x^* + tp)$$

$$f(x^*) - \bar{t}p^T \nabla f(x^* + tp) > f(x^*)$$

Por lo tanto $f(x^* + \bar{t}p) < f(x^*)$ para toda $\bar{t} \in (0, T]$. Tenemos una contradicción, hemos encontrado una dirección que se aleja de x^* y f decrece, por lo cual, x^* no es un mínimo local. Llamaremos x^* a un punto estacionario si $\nabla f(x^*) = 0$.

2.3.2. Condición necesaria de segundo orden

Si x^* es un mínimo local de f y $\nabla^2 f$ es continua en la vecindad de x^*, entonces $\nabla f(x^*) = 0$ y $\nabla^2 f^2(x^*)$ es definida semi positiva.

Por contradicción, asumimos que $\nabla^2 f(x^*)$ no es definido positivo. Entonces seleccionamos un vector p tal que $p^T \nabla^2 f(x^*) p < 0$ para todo $t \in [0, T]$

$$f(x^* + \bar{t}p) = f(x^*) + \bar{t}p^T \nabla f(x^* + \bar{t}p) + \frac{1}{2}\bar{t}^2 p^T \nabla^2 f(x^* + \bar{t}p)p \tag{2.3}$$

$$f(x^*) + \frac{1}{2}\bar{t}^2 p^T \nabla^2 f(x^* + \bar{t}p)p > f(x^*)$$

por lo tanto

$$f(x^* + \bar{t}p) < f(x^*) \tag{2.4}$$

por lo cual $f(x^*)$ no es un mínimo.

2.3.3. Ejemplo

Dada la función $f(x, y) = x^2 + y^2 + 5xy + 2x - 3y - 20$ determinar:

1. La expresión del gradiente $\nabla f(x, y)$,
2. la expresión del Hessiano $\nabla^2 f(x, y)$ y
3. el punto de inflexion.

El vector gradiente queda como:

$$\nabla f(x, y) = \begin{bmatrix} \text{2x+5y +2} \\ \text{2y+5x-3} \end{bmatrix}$$

La matriz Hessiana es

$$\nabla^2 f(x, y) = \begin{pmatrix} 2 & 5 \\ 5 & 2 \end{pmatrix}$$

El punto extremo lo calculamos resolviendo el sistema de ecuaciones

$$\begin{pmatrix} 2 & 5 \\ 5 & 2 \end{pmatrix} \begin{bmatrix} \text{x} \\ \text{y} \end{bmatrix} = \begin{bmatrix} -2 \\ 3 \end{bmatrix}$$

La solución es $x = \frac{19}{21}$ y $y = \frac{-16}{21}$. El determinante del Hessiano $det(\nabla^2 f(x, y)) = -21$ por lo cual se trata de un máximo (ver [**?**]).

2.4. Dirección de búsqueda para métodos de búsqueda en línea

La dirección de descenso $-\nabla f_k$ es la opción más obvia como dirección de búsqueda. Es intuitivo que nos movamos en la dirección en la que más rápido decrece f. Para verificar esto, utilizaremos el teorema de Taylor:

$$f(x_k + \alpha p) = f(x_k) + \alpha p^T \nabla f(x_k) + \frac{1}{2}\alpha^2 p^T \nabla^2 f(x_k) p$$

El rango de cambio de f, en la dirección p en el punto x_k, es simplemente el coeficiente α. Por lo tanto la dirección unitaria p de más rápido descenso es la solución del problema:

$$\min_p p^T \nabla f_k \qquad para \qquad \|p\| = 1 \tag{2.5}$$

Recordemos $p^T \nabla f_k = \|p\| \|\nabla f_k\| \cos\theta$, donde θ es el ángulo entre p y ∇f_k tenemos que $\|p\| = 1$ por lo cual:

$$p^T \nabla f_k = \|\nabla f_k\| \cos\theta$$

por lo tanto, la función objetivo es minimizada cuando $\cos\theta$ toma el valor de -1, lo cual sucede, cuando $\theta = \pi$ *radianes*. Sustituyendo en la ecuación tenemos:

$$\begin{aligned} pp^T \nabla f_k &= p\|\nabla f_k\| \cos\pi \\ \nabla f_k &= -p\|\nabla f_k\| \\ p &= -\frac{\nabla f_k}{\|\nabla f_k\|} \end{aligned}$$

El vector unitario que minimiza la ecuación 2.5 es:

$$p = -\frac{\nabla f_k}{\|\nabla f_k\|}$$

El método de descenso de gradiente es un método de búsqueda lineal, que se mueve en dirección $p = -\nabla f_k$ en cada paso, tal que:

$$\text{mín}\, f(x_k + \alpha p_k)$$

2.4.1. Curvas de Nivel y el vector Gradiente

El gradiente de una función f en un punto en particular es perperdicular a la curva de nivel de f en el mismo punto.

Prueba:

Consideremos que X_0 es el punto de interés. La curva de nivel que pasa a través de X_0 es una $X|f(X) = f(X_0)$. Consideremos la curva paramétrica $\gamma(t)$ es la curva de nivel que pasa a traves de X, por lo tanto podemos asumir que $\gamma(0) = X_0 = [x_1(t), x_2(t), \cdots, x_n(t)]^T$. Entonces tenemos

$$f(\gamma(0)) = f(X_0) = C$$

Ahora vamos a calcular la derivada de la función con respecto al parámetro t utilizando la regla de la cadena

$$\begin{aligned}
\frac{df(x_1(t), x_2(t), \cdots, x_n(t))}{dt} &= \frac{\partial f(X)}{\partial x_1}\frac{dx_1}{dt} + \frac{\partial f(X)}{\partial x_2}\frac{dx_2}{dt} + \cdots + \frac{\partial f(X)}{\partial x_n}\frac{dx_n}{dt} = 0 \\
&= \left[\frac{\partial f(X)}{\partial x_1}, \frac{\partial f(X)}{\partial x_2}, \cdots \frac{\partial f(X)}{\partial x_n}\right] \begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \\ \vdots \\ \frac{dx_n}{dt} \end{bmatrix} = 0 \\
&= [\nabla f(X_0)]^T \gamma(X_0) = 0
\end{aligned}$$

De la formula anterior podemos ver que $f'(X(t))$ es el producto escalar del gradiente y la derivada de la curva de nivel. El producto escalar de vectores, que también puede ser escrito como $||\nabla f(x_0)||||y'(0)||cos(\theta)$ y la única posibilidad de que este sea cero es que el ángulo entre ambos vectores sea 90°. Por lo tanto la el angulo entre la tangente a la curva de nivel $\gamma'(X_0)$ forma un ángulo de 90° con la dirección del vector gradiente. En la figura 2.1 podemos ver las curvas de nivel en un punto y la dirección del menos gradiente.

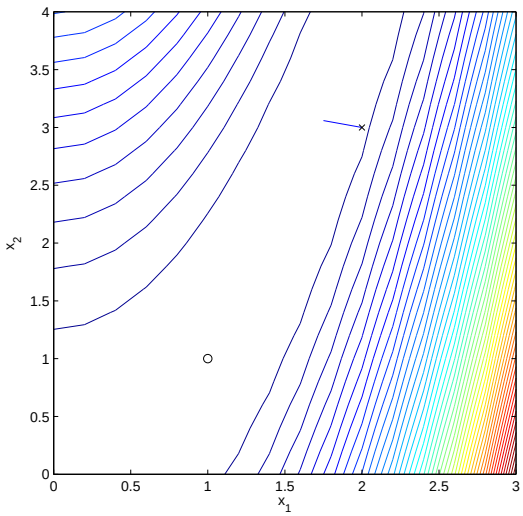


Figura 2.1: Curvas de nivel para la función de Rosenbrock

Ver [**?**]

Tarea 2

Producto cruz y Serie de Taylor

1.- Determinar el ángulo entre los vectores $a = \widehat{i} + 3\widehat{j} - 2\widehat{k}$

y $b = 3\widehat{k}$

2.- Realizar la aproximación en serie de Taylor de tercer grado, de la función $f(x) = e^x + cos(x)$ en el punto $a = 0$

2.5. Métodos de búsqueda en una dirección

Los métodos de búsqueda en una sola dirección calculan una dirección p_k y entonces deciden que tan lejos se deben mover en esta dirección. La iteración esta dada a por:

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

donde el escalar positivo α es llamada el tamaño de paso

De acuerdo a lo anterior la mejor distancia de descenso es:

$$p = -\frac{\nabla f}{\|\nabla f\|}$$

El algoritmo de búsqueda es 4:

2.5.1. Ejemplo

Este ejemplo muestra como se da la búsqueda en una direción. Así dada la función en dos dimensiones $f(x, y) = e^{-x^2-y^2}x$ cuya imagen se muestra en la figura 2.2 y considerando como punto inicial $X_0 = [-1, -1]$:
El vector gradiente esta dado por

$$g(x, y) = \begin{bmatrix} e^{-x^2-y^2} - 2e^{-x^2-y^2}x^2 \\ -2e^{-x^2-y^2}xy \end{bmatrix}$$

La dirección del gradiente es $d = \left[-\frac{1}{\sqrt{5}}, -\frac{2}{\sqrt{5}}\right]$ y la dirección contraria al gradiente es $p = \left[\frac{1}{\sqrt{5}}, \frac{2}{\sqrt{5}}\right]$

Algoritmo 4 Algoritmo de Búsqueda en la dirección del gradiente

BÚSQUEDA-UNIDIMENSIONAL-GRADIENTE$(x_0, \alpha, f(x), T)$

1 `Hacemos` $k \leftarrow 0$
2 `Calculamos el valor del gradiente` $\nabla f(x_k)$
3 `Calculamos` $p_k \leftarrow -\frac{\nabla f_k}{\|\nabla f_k\|}$
4 **repetir**
5 $\quad x_{k+1} = x_k + \alpha * p$
6 $\quad k \leftarrow k + 1$
7 **hasta** $f(x_{k+1}) < f(x_k)$
8 **regresar** x_{k+1}

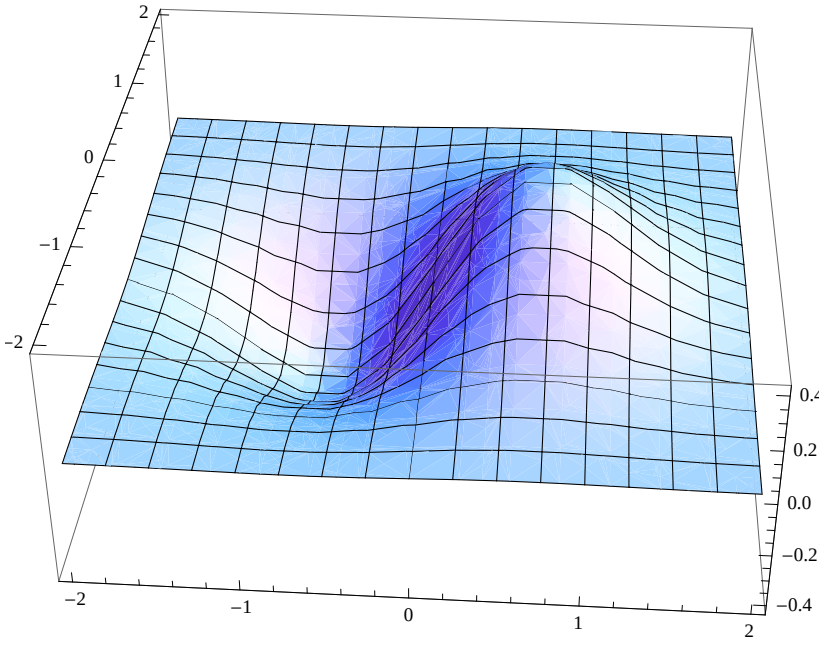


Figura 2.2: Función $f(x, y) = e^{-x^2-y^2}x$

La función a minimizar en la dirección del gradiente es

$$\phi(\alpha) = \left(-1 + \frac{\alpha}{\sqrt{5}}\right) e^{-\left[\left(-1+\frac{\alpha}{\sqrt{5}}\right)^2 + \left(-1+\frac{2\alpha}{\sqrt{5}}\right)^2\right]}$$

cuya imagen se muestra en la figura 2.3, en esta podemos verificar que el minimo $\phi(0.952194) = -0.403933$.

Los valores x y y se calculan como:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix} - 0.952194 \times \begin{bmatrix} -\frac{1}{\sqrt{5}} \\ -\frac{2}{\sqrt{5}} \end{bmatrix} = \begin{bmatrix} -0.574166 \\ -0.148331 \end{bmatrix}$$

Obviamente la función $f(-0.574166, -0.148331) = -0.403933$ y no es el mínimo global de la función. La función tiene un minimo global en $X = [-\frac{1}{\sqrt{2}}, 0]^T$ con $f(-\frac{1}{\sqrt{2}}, 0) = -0.428882$ y el valor calculado esta lejos de la solución, sin embargo, nos hemos movido hacia abajo.

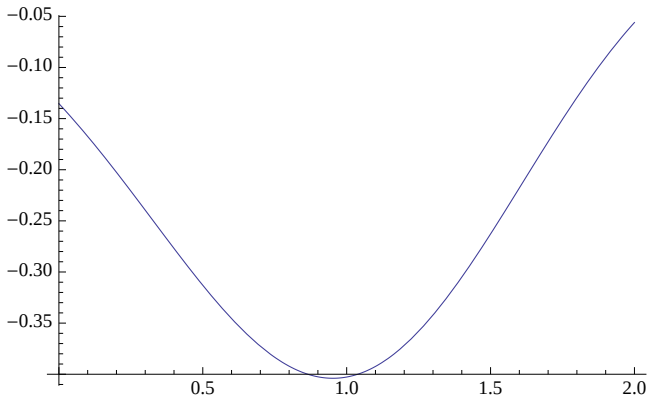


Figura 2.3: Función $\phi(X_0)$

2.5.2. Ejemplo

Consideremos la función $f : R^2 \to R$ dada por la ecuación 2.6, queremos calcular el mínimo de la función utilizando la dirección de gradiente. Dado punto de partida $x_0 = [2, 3]^T$ y un tamaño de paso $\alpha = 0.05$ calculamos el mínimo de la función.

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{2.6}$$

El gradiente de la función para cualquier valor de x es:

$$\nabla f(x) = \begin{bmatrix} -400(x_2 - x_1^2)x_1 + 2x_1 - 2 \\ 200x_2 - 200x_1^2 \end{bmatrix}$$

Si valuamos el gradiente en x_0

$$\nabla f(x) = \begin{bmatrix} 802 \\ -200 \end{bmatrix}$$

La dirección de búsqueda es

$$p = -\begin{bmatrix} 802/\sqrt{802^2 + 200^2} \\ -200/\sqrt{802^2 + 200^2} \end{bmatrix} = \begin{bmatrix} -0.97028 \\ +0.24197 \end{bmatrix}$$

En la siguiente tabla tenemos los resultados de la búsqueda lineal. Note que el mínimo de la función se localiza en el punto $x* = [1, 1]$ y el algoritmo no esta llegando a tal valor. øA que se debe esta situación?. En la figura 2.1 se presenta la solución calculada por este metodo. Con una x se senala el valor inicial y con una o el minimo global

k	$\alpha * k$	x_1	x_2	f
0	0	2.00000	3.00000	101.000000
1	0.05	1.951486	3.0120985	64.2986276
2	0.10	1.902972	3.024197	36.46884822
3	0.15	1.854458	3.0362955	16.94835566
4	0.20	1.805944	3.048394	5.188138435
5	0.25	1.75743	3.0604925	0.652479811
6	0.30	1.708916	3.072591	2.81895777

Tarea 3

Modificar el algoritmo Búsqueda en la dirección de gradiente

Dado el algoritmo 4, hacer la modificaciones necesaria para que en lugar de realizar búsqueda exhaustiva, el algoritmo utilice la razón dorada. Utilice los datos del ejemplo anterior

2.6. Escalamiento

El desempeño de un algoritmo dependerá principalmente, de cómo el problema es formulado. En optimización sin restricciones, un problema es llamado pobremente escalado si producen más cambios en una dirección que en otra. Un ejemplo simple es el propuesto por la función:

$$f(x) = ax_1^2 + x_2^2$$

Al calcular el gradiente de la función tenemos

$$\nabla f(x) = \begin{bmatrix} 2ax_1 \\ 2x_2 \end{bmatrix}$$

Aplicando el método de búsqueda en una dirección, podremos notar que, es mas sensible a cambios en x_1 que a cambios en x_2.

Ejemplo

Consideremos la función $f(x) = x_1^2 + x_2^2$, la cual podemos ver gráficamente en la figura 2.4.

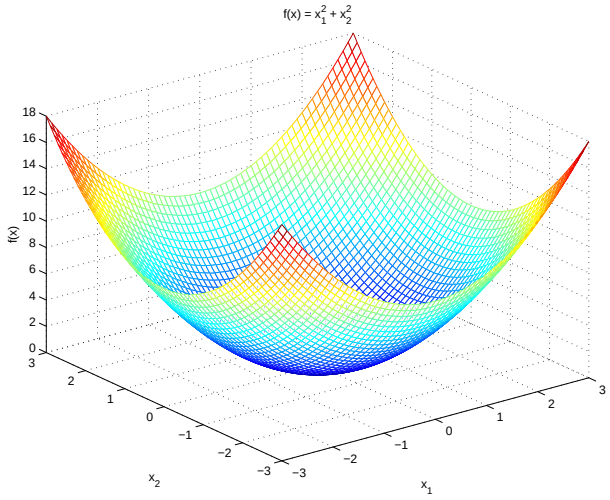


Figura 2.4: Función $f(x) = x_1^2 + x_2^2$

Con un punto inicial $x_0 = [1, 1]$ y calculando la dirección de búsqueda utilizando el gradiente, podemos ver la línea de búsqueda en la figura 2.5. Note que siguiendo la línea pasamos sobre el mínimo de la función.

Ahora si modificamos la función $f(x) = 2x_1^2 + x_2$, tendremos un escalamiento en la dirección x_1. Las curvas de nivel de esta se muestran en la figura 2.6 y podemos notar que la linea de búsqueda no pasará por el mínimo.

Ver [**?**]

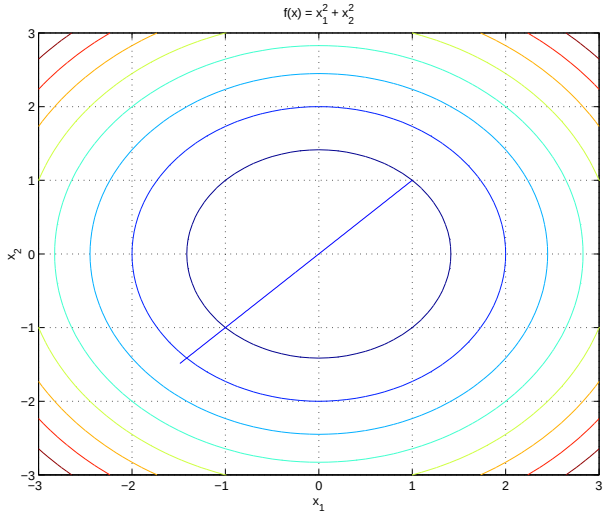


Figura 2.5: Curvas de nivel de la función $f(x) = x_1^2 + x_2^2$

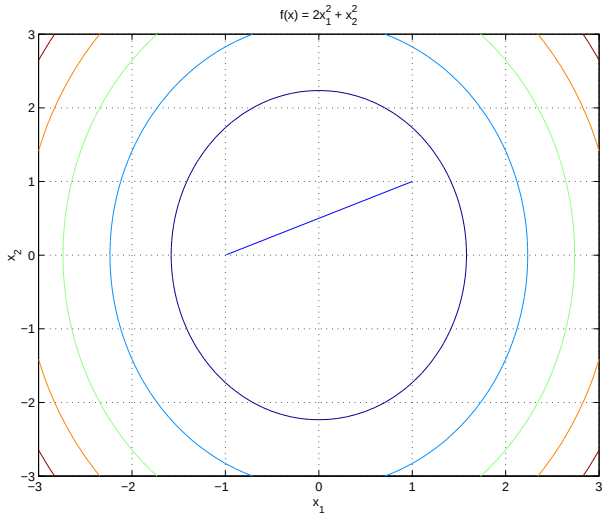


Figura 2.6: Curvas de nivel de la función $f(x) = 2x_1^2 + x_2^2$

2.7. Las condiciones de Wolfe

La popular e inexacta búsqueda lineal estipula que α_k debe dar suficiente decrecimiento en la función objetivo f, como medida de esto se tiene la siguiente desigualdad:

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T p_k$$

Si definimos $\phi(\alpha) = f(x_k + \alpha p_k)$ y $l(\alpha) = f(x_k) + c_1 \alpha \nabla f_k^T p_k$, podemos escribir

$$\phi(\alpha) \leq l(\alpha)$$

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

La condición suficiente de decaimiento es ilustrada en la figura 2.7. El lado derecho de la ecuación la llamaremos $l(\alpha)$. La función $l(\alpha)$ tiene pendiente negativa c, $\nabla {f_K}^T P_K$ pero dada $c_1 \in (0, 1)$ esta miente con respecto a que tan pequeña deben de ser los valores de α. En la práctica podemos decir que $c_1 = 10^{-4}$. La condición de Armijo no es suficiente

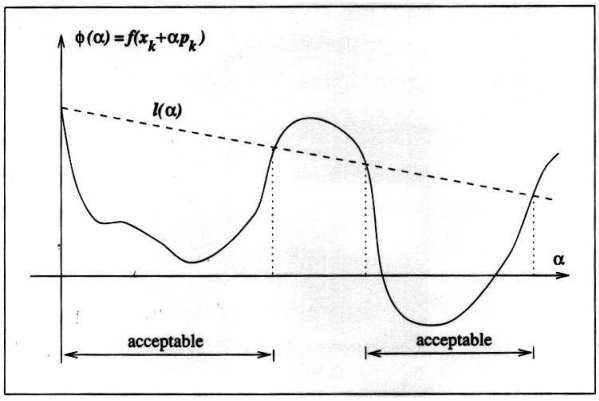


Figura 2.7: Condición de decrecimiento suficiente

por si misma para asegurar que el algoritmo tiene progresos razonables. Por otro lado introduciremos un segundo requerimiento llamada la condición de curvatura, la cual requiere que α_k satisfaga:

$$\nabla f(x_k + \alpha p_k)^T p_k \geq c_2 \nabla {f_k}^T p_k \tag{2.7}$$

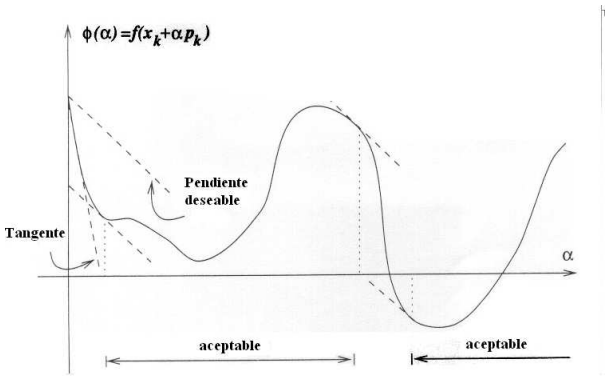


Figura 2.8: Condición de curvatura

donde: $c_2 \in (c_1, 1)$. Esta condición se observa en la figura 2.8.

Las condiciones de decrecimiento suficiente y pendiente son conocidas como las condiciones de Wolfe, las cuales están ilustradas en la figura 2.9 y se resumen en 2.8

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T p_k$$

$$\nabla f(x_k + \alpha p_k)^T p_k \geq c_2 \nabla {f_k}^T p_k \tag{2.8}$$

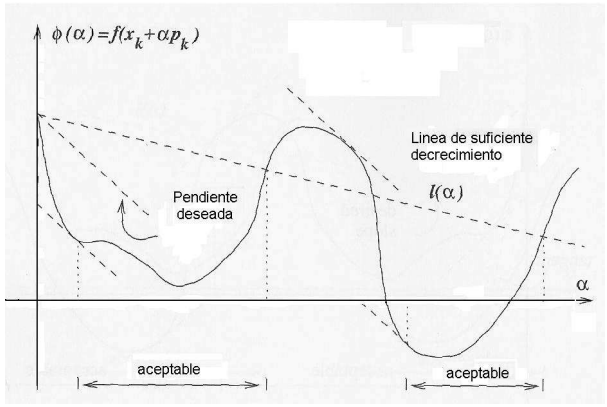


Figura 2.9: Condiciones de wolfe

Ver [**?**]

Las condiciones fuertes de wolfe están dada por las ecuaciones 2.9 y garantizan que tenemos las condiciones suficientes para tener un mínimo.

$$\begin{aligned} f(x_k + \alpha p_k) &\leq f(x_k) + c_1 \alpha \nabla f_k^T p_k \\ |\nabla f(x_k + \alpha p_k)^T p_k| &\leq c_2 |\nabla {f_k}^T p_k| \end{aligned} \tag{2.9}$$

2.7.1. Ejemplo

Para una función $f(x)$ tenemos una función $\Phi(\alpha) = cos(\alpha + \frac{\pi}{4})$, calcular la gráfica $\Phi(\alpha)$ donde se cumplen:

1. las condiciones de armijo,
2. las condiciones de curvatura,
3. las condiciones fuertes de curvatura y
4. las condiciones fuentes de wolfe

considere $c_1 = 10^{-4}$ y $c_2 = 1$

En la figura 2.10 se muestran las condiciones de wolfe para la función, en obscuro se muestra la región donde no se cumplen las condiciones. Las condiciones de Armijo, Curvatura, Curvartura fuerte y Wolfe fuerte se muestran en las figuras 2.10(a), 2.10(b), 2.10(c) y 2.10(d) respectivamente.

2.7.2. Ejemplo

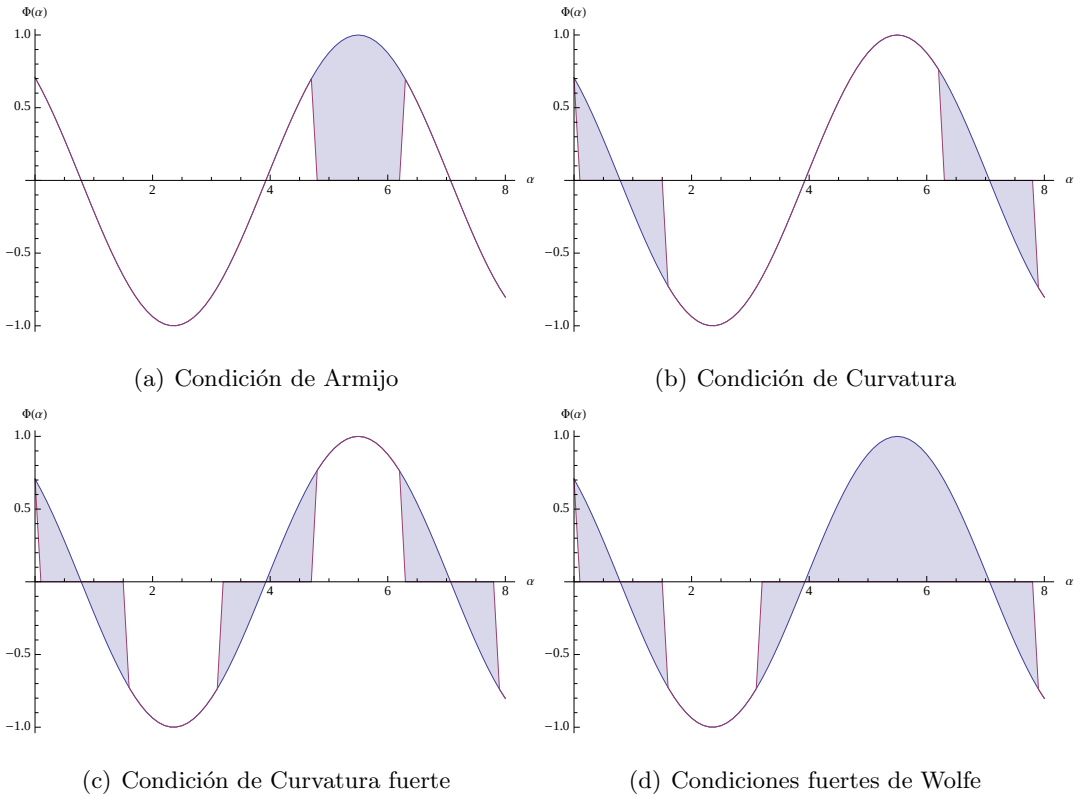
Para una función $f(x)$ dada como

$$f(x) = \sum_{i=1}^{10} \left[(x_i - y_i)^2 + 2.5(x_i - x_{i-1})^2 \right]$$

determinar el rango de valores en el cual se cumplen las condiciones de fuertes de Wolfe. Considere $c_1 = 10^{-4}$, $c_2 = 1.0$, $y = [1, 2, 3, 4, 5, 4, 3, 2, 1, 0]^T$ y $x_0 = y$

En la figura 2.11, se muestra el intervalo en el cual son validas las condiciones fuertes de wolfe. En obscuro se muestra la region donde ya no se cumplen dichas condiciones para la función dada.

Tarea 4



(a) Condición de Armijo

(b) Condición de Curvatura

(c) Condición de Curvatura fuerte

(d) Condiciones fuertes de Wolfe

Figura 2.10: Condiciones de Wolfe

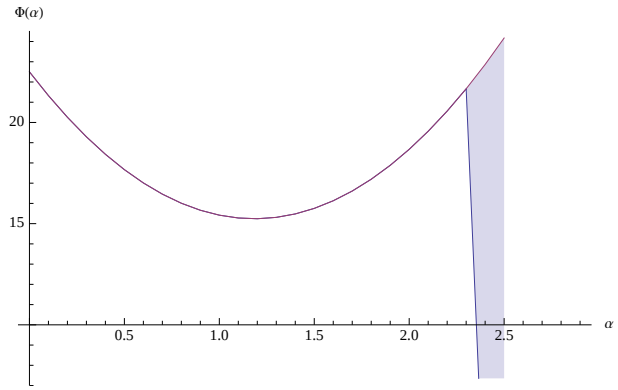


Figura 2.11: Condiciones fuertes de wolfe

Condiciones de Wolfe

Dada la función $100(x_2 - x_1^2)^2 + (1 - x_1^2)$, determinar según las condiciones de Wolfe, cuales

son los valores de α aceptables. Considere valores para $c_1 = 10^{-4}$ y $c_2 = 1.0$ y como valor inicial $x_0 = [2, 3]^T$

2.7.3. Algoritmo de Suficiente decaimiento y muestreo hacia atrás

La condición de Armijo (ecuación 2.7), también llamada condición de suficiente decaimiento es suficiente, no garantiza que el algoritmo seleccione una longitud de paso apropiadamente. Sin embargo si utilizamos el algoritmo de muestreo hacia atrás, la condición de Armijo será suficiente.

Algoritmo 5 Búsqueda-lineal-con muestreo-hacia-atras

BÚSQUEDA-LINEAL-CON MUESTREO-HACIA-ATRAS($\overline{\alpha}, \rho, c_1$)

1 `Dado un valor inicial` $\overline{\alpha} > 0$, `un factor de incremento` ρ `y un escalar` $c_1 \in (0, 1)$
2 `Hacemos` $\alpha \leftarrow \overline{\alpha}$
3 **repetir**
4 $\alpha \leftarrow \rho * \alpha$
5 **hasta** $f(x_k + \alpha * p_k) <= f(x_k) + c_1 \alpha \nabla f(x_k)^T p_k$
6 $\alpha_k \leftarrow \alpha$
7 **regresar** α_k

2.8. Algoritmos para seleccionar la longitud del paso

Ahora consideremos las técnicas para encontrar, de la mejor manera, el mínimo de la función $\phi(\alpha)$

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

2.8.1. Máximo descenso

Una manera precisa de calcular α, es utilizar las condiciones de Wolfe, sin embargo una aproximación cuadrática es una alternativa aproximada y rápida.

Si f es una función convexa cuadrática $f(x) = \frac{1}{2}x^T Ax + B^T x + c$ podemos calcular de manera exacta el valor de la longitud de paso utilizando la serie de Taylor:

$$f(x_k + \alpha_k p_k) = f(x_k) + \alpha_k {p_k}^T \nabla f(x_k) + \frac{1}{2}{\alpha_k}^2 {p_k}^T \nabla^2 f(x_k) p_k$$

Dado que nuestra función depende solo de α, derivamos respecto a ella e igualamos a cero:

$${p_k}^T \nabla f(x_k) + \alpha_k {p_k}^T \nabla^2 f(x_k) p_k = 0$$

despejando, obtenemos:

$$\alpha_k = \frac{-p_k^T \nabla f(x_k)}{{p_k}^T \nabla^2 f(x_k) p_k} \tag{2.10}$$

Utilizando esto, tenemos que una iteración de máximo descenso, esta dada por :

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

$$x_{k+1} = x_k - \left[\frac{{p_k}^T \nabla f(x_k)}{{p_k}^T \nabla^2 f(x_k) p_k}\right] p_k$$

El el caso que p_k sea un vector unitario con dirección opuesta al gradiente, el algoritmo se llamará, Máximo Descenso de Gradiente.

$$x_{k+1} = x_k - \left[\frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_k)^T \nabla^2 f(x_k) \nabla f(x_k)}\right] \nabla f(x_k)$$

2.8.2. Interpolación

Suponga que tenemos un valor α_0, tal que cumple con la regla de Armijo (eq2.7):

$$\phi(\alpha_0) \leq \phi(0) + c_1 \alpha_0 \phi'(0)$$

donde $\phi(\alpha_0) = f(x_k + \alpha p_k)$ y $\phi'(0) = \nabla^T(f_k) p_k$

De ser así, tenemos que el valor que minimiza la función $\phi(\alpha)$, se encuentra en el intervalo $[\alpha_0, \alpha_1]$. Así pues hacemos una aproximación cuadrática de $\phi_q(\alpha)$ dada por:

$$\phi_q(\alpha) = A(\alpha - \alpha_0)^2 + B(\alpha - \alpha_0) + C \tag{2.11}$$

Dado que conocemos $\phi(\alpha_0)$, $\phi'(\alpha_0)$ tenemos la información suficiente para calcular los coeficientes de la ecuación cuadrática 2.11.

Para determinar C, valuamos la función 2.11, en α_0

$$\phi_q(\alpha_0) = C = \phi(\alpha_0)$$

Para calcular B, derivamos la función 2.11 y la valuamos en α_0

$$\phi'_q(\alpha_0) = 2A(\alpha_0 - \alpha_0) + B = \phi'(\alpha_0)$$

$$\therefore B = \phi'(\alpha_0) = \nabla^T(f_k + \alpha_0 p_k)p_k$$

Finalmente el valor de A lo calculamos sustituyendo B y C en la ecuación 2.11

$$\phi_q(\alpha_1) = A(\alpha_1 - \alpha_0)^2 + \phi'(\alpha_0)(\alpha_1 - \alpha_0) + \phi(\alpha_0)$$

$$A = \frac{\phi_q(\alpha_1) - \phi'(\alpha_0)(\alpha_1 - \alpha_0) - \phi(\alpha_0)}{(\alpha_1 - \alpha_0)^2}$$

El valor de α que minimiza 2.11 se calcula haciendo

$$\phi'(\alpha) = 2 * A(\alpha - \alpha_0) + B = 0$$

$$\alpha = \alpha_0 - \frac{B}{2A}$$

$$\alpha = \alpha_0 - \frac{\phi'(\alpha_0)(\alpha_1 - \alpha_0)^2}{2(\phi(\alpha_1) - \phi'(\alpha_0)(\alpha_1 - \alpha_0) - \phi(\alpha_0))} \tag{2.12}$$

2.8.3. Método de Newton

Consideremos que la función $\phi : \mathbb{R} \to \mathbb{R}$ es de clase dos. Siguendo con la idea de reemplazar en la vecindad del punto α_k de la función ϕ por una aproximación cuadrática $\phi(\alpha)$ dada por

$$\phi(\alpha_{k+1}) = \phi(\alpha_k) + \phi'(\alpha_k)(\alpha_{k+1} - \alpha_k) + \frac{1}{2}\phi''(\alpha_k)(\alpha_{k+1} - \alpha_k)^2 \tag{2.13}$$

Comenzaremos por recordar que $\phi(\alpha_k) = f(x_0 + \alpha_k p_0)$ y procederemos a hacer la expansión en serie de Taylor para $f(x_{k+1})$

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

Dado un punto inicial x_0 y que la dirección de búsqueda es p_0, suponemos que tenemos diferentes valores de x_k modificando la variable α_k como $x_{k+1} = x_0 + p_0\alpha_{k+1}$. En base a esto podemos decir que

$$x_{k+1} - x_k = (x_0 + p_0\alpha_{k+1}) - (x_0 + p_0\alpha_k) = p_0(\alpha_{k+1} - \alpha_k)$$

$$f(x_0+p_0\alpha_{k+1}) = f(x_0+p_0\alpha_k)+p_0^T\nabla f(x_0+p_0\alpha_k)(\alpha_{k+1}-\alpha_k)+\frac{1}{2}p_0^T\nabla^2 f(x_0+\alpha_k p_0)p_0(\alpha_{k+1}-\alpha_k)^2 \tag{2.14}$$

Podemos notar que (2.13) y (2.14) son equivalentes si

$$\begin{aligned}
\phi(\alpha_{k+1}) &= f(x_0 + p_0\alpha_{k+1}) \\
\phi(\alpha_k) &= f(x_0 + p_0\alpha_k) \\
\phi'(\alpha_k) &= p_0^T\nabla f(x_0 + p_0\alpha_k) \\
\phi''(\alpha_k) &= p_0^T\nabla^2 f(x_0 + \alpha_k p_0)p_0
\end{aligned}$$

El mínimo de la ecuación (2.13) lo podemos calcular como

$$\phi'(\alpha_k) + \phi''(\alpha_k)(\alpha_{k+1} - \alpha_k) = 0$$

despejando para α_{k+1} tenemos

$$\begin{aligned}
\alpha_{k+1} &= \alpha_k - \frac{\phi'(\alpha_k)}{\phi''(\alpha_k)} \\
\alpha_{k+1} &= \alpha_k - \frac{p_0^T\nabla f(x_0 + p_0\alpha_k)}{p_0^T\nabla^2 f(x_0 + \alpha_k p_0)p_0}
\end{aligned} \tag{2.15}$$

El algoritmos de Newton para búsqueda del α de máximo descenso se muestra en 6.

2.8.4. Ejemplo

Para la función de Ronsenbrock determinar el valor del paso optimo α^* para un punto inicial $x_0 = [2, 3]$ en un rango $\alpha \in [0, 0.3]$, utilizando máximo descenso, inteporlación y Newton de máximo descenso. La función de Ronsenbrock es

$$f(x) = 100 * (x_2 - x_1^2)^2 + (1 - x_1)^2$$

Algoritmo 6 Algoritmo de Newton Unidimensional

NEWTON α MÁXIMO DESCENSO$(f(x), x_0)$

1 `Dado una función diferenciable dos veces` $f(x)$

2 `una dirección de búsqueda` p_0 `y un valor inicial` x_0

3 **repetir**

4 $\alpha_{k+1} = \alpha_k - \frac{p_0^T \nabla f(x_0 + p_0 \alpha_k)}{p_0^T \nabla^2 f(x_0 + \alpha_k p_0) p_0}$

5 $k = k + 1$

6 **hasta** $\|p_0^T \nabla f(x_0 + \alpha_k)\| \leq \varepsilon$

7 **regresar** α_{k+1}

El gradiente de la función para cualquier valor de x es:

$$\nabla f(x) = \begin{bmatrix} -400(x_2 - x_1^2)x_1 + 2x_1 - 2 \\ 200x_2 - 200x_1^2 \end{bmatrix}$$

El hessiano es

$$\nabla^2 f(x) = \begin{array}{|c|c|} \hline 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ \hline -400x_1 & 200 \\ \hline \end{array}$$

Si valuamos el gradiente y el hessiano en $x_0 = [2, 3]^T$

$$\nabla f(2,3) = \begin{bmatrix} 802 \\ -200 \end{bmatrix}$$

$$\nabla^2 f(2,3) = \begin{bmatrix} 3602 & -800 \\ -800 & 200 \end{bmatrix}$$

La dirección de búsqueda es

$$p = - \begin{bmatrix} 802/\sqrt{802^2 + 200^2} \\ -200/\sqrt{802^2 + 200^2} \end{bmatrix} = \begin{bmatrix} -0.97028 \\ +0.24197 \end{bmatrix}$$

Máximo descenso

Para el calculo utilizamos la ecuación 2.10

$$\alpha_k = \frac{-\left[-0.97028, 0.24197\right] \begin{bmatrix} 802 \\ -200 \end{bmatrix}}{\left[-0.97028, 0.24197\right] \begin{bmatrix} 3602 & -800 \\ -800 & 200 \end{bmatrix} \begin{bmatrix} -0.97028 \\ +0.24197 \end{bmatrix}} = 0.2188$$

Interpolación cuadrática

De acuerdo con la ecuación 2.12 necesitamos calcular $\phi(0)$, $\phi'(0)$ y dado $\alpha_0 = 0$ $\alpha_0 = 0.3$. De la definición tenemos que

$$\phi(\alpha_k) = f(x_k + \alpha_k p_k)$$

$$\phi(\alpha_0) = \phi(0) = f(2,3) = 101$$

$$\phi'(\alpha_k) = p_k^T \nabla f(x_k + \alpha_k p_k)$$

$$\phi'(\alpha_0) = \phi'(0) = p_k^T \nabla f(x_k) = \left[-0.97028, 0.24197\right] \begin{bmatrix} 802 \\ -200 \end{bmatrix} = -826.5586$$

$$\phi(\alpha_1) = \phi(0.3) = 2.8189$$

El valor final para el paso es

$$\alpha_k = 0 - \frac{(-826.5586) * (0.3 - 0)^2}{2 * [2.8189 - 101 - (-826.5586) * (0.3 - 0)]} = 0.2483$$

Newton

La actualización del método de newton es

$$\alpha_{k+1} = \alpha_k - \frac{\phi'(\alpha_k)}{\phi''(\alpha_k)}$$

Las iteraciones para este método son:

$$\begin{aligned}
\alpha_1 &= 0.000000 - \frac{802.000000}{3602.0} = 0.218756 \\
\alpha_2 &= 0.218756 - \frac{103.904780}{2616.063257} = 0.257093 \\
\alpha_3 &= 0.257093 - \frac{3.044904}{2454.412016} = 0.258247 \\
\alpha_4 &= 0.258247 - \frac{0.102357}{2449.595032} = 0.258248 \\
\alpha_5 &= 0.258248 - \frac{0.099721}{2449.590714} = 0.258248
\end{aligned}$$

Ejemplo

Para la función $f(x,y) = xe^{-x^2-y^2}$, un punto de coordenadas $x_0 = [-0.5, -0.5]$ y una dirección $\theta = 30^0$ determinar:

1. La derivada direccional en la dirección dada,
2. el mínimo en la dirección dada utilizando interpolación cuadrática y
3. el mínimo utilizando Razón Dorada.

El vector gradiente para esta función es:

$$\nabla f(x,y) = \begin{bmatrix} (1-2x^2)e^{(-x^2-y^2)} \\ -2xye^{(-x^2-y^2)} \end{bmatrix}$$

El Hessiano es

$$\nabla^2 f(x,y) = \begin{bmatrix} (-6x+4x^3)e^{(-x^2-y^2)} & (-2y+4x^2y)e^{(-x^2-y^2)} \\ (-2y+4x^2y)e^{(-x^2-y^2)} & (-2x+4xy^2)e^{(-x^2-y^2)} \end{bmatrix}$$

El vector unitario en la dirección dada es $u_{30} = [0.866, 0.5]^T$, el gradiente valuado en el punto dado es $\nabla f(-0.5,-0.5) = [0.303265, -0.303265]^T$ y la derivada direccional

$$D_{u_{30}} = [0.303265, -0.303265] \begin{bmatrix} 0.866 \\ 0.5 \end{bmatrix} = 0.11102$$

Con esto podemos definir $l(\alpha) = f(x) + \alpha D_{u_{30}} = -0.303265 + 0.11102\alpha$ y $\phi(\alpha) = f(x + \alpha D_{u_{30}})$. Suponiendo $\alpha_0 = 0.1$ tenemos:

$$x_1 = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} + 0.1 * \begin{bmatrix} 0.866 \\ 0.5 \end{bmatrix} = \begin{bmatrix} -0.4134 \\ -0.45 \end{bmatrix}$$

Utilizando el algoritmo de interpolación cuadrática tenemos que

$$\alpha = \alpha_0 - \frac{\phi'(\alpha_0)(\alpha_1 - \alpha_0^2)}{2(\phi(\alpha_1) - \phi'(\alpha_0)(\alpha_1 - \alpha_0) - \phi(\alpha_0))}$$

$$\alpha = 0 - \frac{0.11102 * (0.1 - 0)^2}{2(-0.28458) - 0.11102 * (0.1 - 0) - 0.303265)}$$

$$\alpha = -0.073154$$

Con el algoritmo de razón dorada tenemos que $\alpha = 8.0123461723868 * 10^{-16}$.

Podemos notar que el valor *alpha* interpolado es negativo y el de razón dorada es casi cero. Esta condición se debe a que en la dirección dada la curvatura es negativa y por lo tanto en la dirección de u_{30^0} hay un máximo, tal como se puede confirmar en la figura 2.12

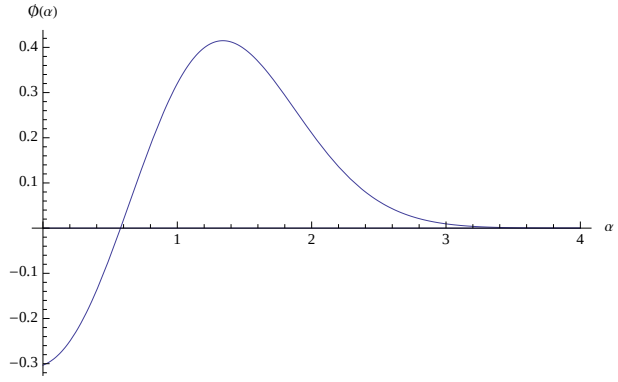


Figura 2.12: Función $\phi(\alpha)$ para la función $xe^{-x^2-y^2}$, en la dirección u_{30}

2.9. Método de descenso de gradiente

El método del descenso de gradiente es un algoritmo de búsqueda lineal en la dirección de gradiente. Los pasos de este algoritmo son:

El valor de α_k, para el algoritmo 7, puede ser calculado utilizando:

Algoritmo 7 Método de descenso de Gradiente

DESCENSO_DE_GRADIENTE($f(x), x_0$)

1 `Dado una función diferenciable` $f(x)$ `y un valor inicial` x_0
2 `Hacemos` $k \leftarrow 0$
3 **repetir**
4 `Evaluar la dirección de búsqueda` $p_k = -\frac{\nabla f(x_k)}{|\nabla f(x_k)|}$
5 `Calcular el valor de` α_k
6 `Hacer la actualización` $x_{k+1} = x_k + \alpha_k p_k$
7 $k \leftarrow k + 1$
8 **hasta** $\|\nabla f(x_k)\| \geq \varepsilon$
9 **regresar** x_k

- Búsqueda exhaustiva (eq. 1.1)
- Razón dorada (sección 2)
- Búsqueda lineal con muestreo hacia atrás (sección 2.7.3).
- Máximo descenso (eq. 2.10)
- Interpolación cuadrática (eq. 2.12)
- Método de Newton de Alfa de máximo descenso (2.15)

2.9.1. Ejemplo

Dada la función $f(x) = \sin(10x_1^2 + x_2^2)$ y un punto inicial $x_0 = [1, 2]^T$, calcular el mínimo utilizando el algoritmo de descenso de gradiente y calculando el paso óptimo con:

1. Búsqueda lineal con muestreo hacia atrás y refinando con Interpolación cuadrática
2. Búsqueda lineal con muestreo hacia atrás y razón dorada

3. Máximo descenso

En todos los casos calcular el ángulo entre direcciones consecutivas utilizando la ecuación $\cos\theta = p_{k+1}^T * p_k$

2.9.2. Convergencia del Método de descenso de gradiente

La principal desventaja del método de descenso de gradiente es el hecho de que la convergencia puede ser muy lenta para cierto tipo de funciones o valores iniciales. En general direcciones de búsqueda consecutivas están a 90 grados tal como se muestra en la figura 2.13.

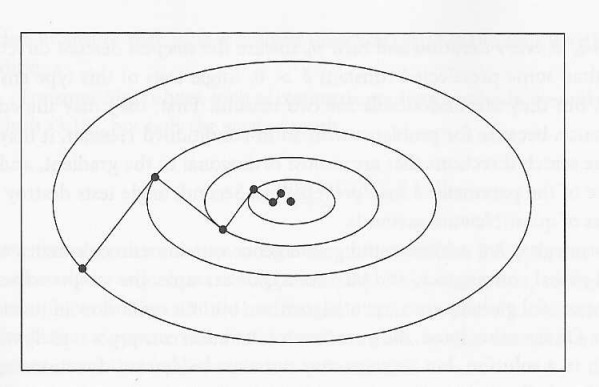


Figura 2.13: Direcciones de búsqueda para descenso de gradiente

Prueba

En los métodos de descenso de gradiente queremos calcular el valor α_k que minimiza

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

por lo tanto tenemos que calcular la derivada, de esta, respecto a α_k e igualar a cero

$$\frac{\mathrm{d}\phi(\alpha_k)}{\mathrm{d}\alpha_k} = {p_k}^T \nabla f(x_k + \alpha_k p_k) = {p_k}^T p_{k+1} = 0$$

De la definición del producto escalar de vectores, sabemos que:

$$\|p_k\| \, \|p_{k+1}\| \cos\theta = 0$$

la única manera de que esto se cumpla es que $\theta = 90^O$. Por lo tanto, las direcciones de búsqueda se encuentran a 90 grados.

Tarea 5

Comparación de los métodos de búsqueda

Implementar el algoritmo de descenso de gradiente 7 y comparar su desempeño con las diferentes estrategias de búsqueda lineal para calcular el α_k

2.9.3. Mejora del Método de descenso de gradiente

Para eliminar la característica de zigzag (ver figura 2.14) de los métodos de descenso de gradiente cuando se aplica a una función mal condicionada el siguiente método de aceleración propuesto por Forsythe and Luenberger. En cada paso k, almacenamos (comenzando en x_k) m estados del método de descenso de gradiente, el cual producirá un punto al que llamaremos y^k. El punto x_{k+1} es entonces calculado como la minimización en la dirección $d_k = y^k - x^k$ en lugar de p_k (dirección opuesta al gradiente).

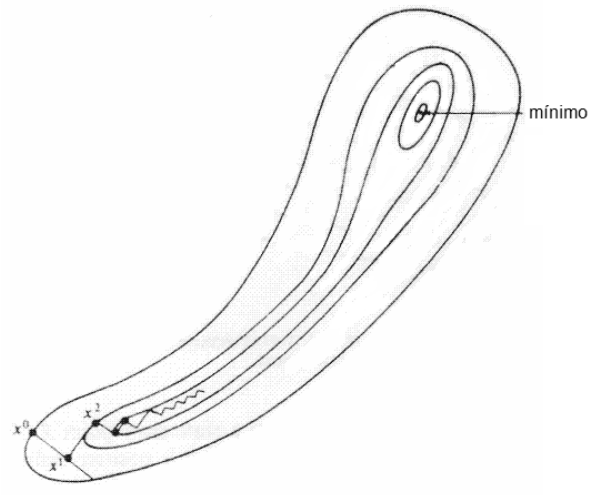


Figura 2.14: Efecto de zigzag creado por el método de descenso de gradiente

Este algoritmo queda como se muestra en 8:

- Dado un x_k y m estados,

Algoritmo 8 Mejora del Método de descenso de Gradiente

FORSYTE($f(x), x_0, m$)

1 `Dado una función diferenciable` $f(x)$ `y un valor inicial` x_0
2 `Hacemos` $k \leftarrow 0$
3 **repetir**
4 `Calcular los m estados haciendo`
5 x_{k+1} `= Descenso_de_Gradiente(`$f(x)$`,` x_k`)`
6 x_{k+2} `= Descenso_de_Gradiente(`$f(x)$`,` x_{k+1}`)`
7 ...
8 x_{k+m} `= Descenso_de_Gradiente(`$f(x)$`,` x_{k+m-1}`)`
9 `hacemos` $y_k = x_{k+m}$
10 `la nueva dirección de búsqueda es` $d_k \leftarrow y_k - x_k$
11 `Calcular el valor de` α_k `que minimiza` $f(x_k + \alpha_k d_k)$
12 `Hacer la actualización` $x_{k+1} = x_k + \alpha_k d_k$
13 $k \leftarrow k + 1$
14 **hasta** $\|\nabla f(x_k)\| \geq \varepsilon$
15 **regresar** x_k

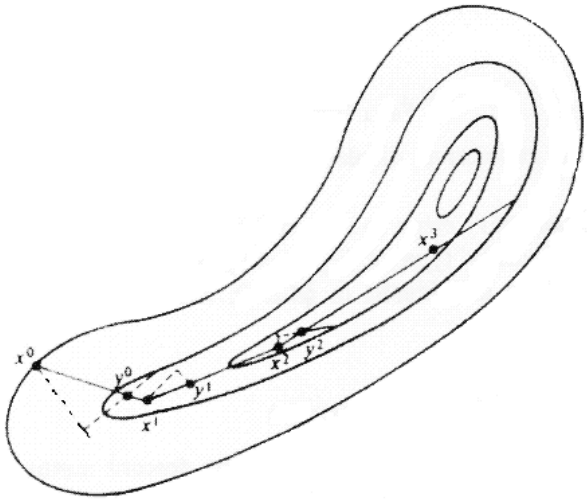


Figura 2.15: Algoritmo de Forsyte de dos pasos

- Calculamos los valores temporales de x_{k+1}, x_{k+2}, ..., x_{k+m} utilizando el algoritmo de descenso de gradiente.
- hacemos $y_k = x_{k+m}$
- la nueva dirección de búsqueda es $d_k = y_k - x_k$
- Buscamos el valor de α_k que minimiza $f(x_k + \alpha_k d_k)$
- Hacemos la actualización $x_{k+1} = x_k + \alpha_k d_k$

El la figura 2.15, podemos ver el desempeño del algoritmo de Forsyte y Luenberger de dos pasos para la función mostrada en la figura 2.14. Note la mejora en el desempeño del algoritmo.

2.9.4. Ejemplo

Implementar la mejora al método de descenso de gradiente propuesta por Forsythe y Luenberger. Los resultados de las iteraciones de búsqueda lineal con muestreo hacia atrás refinando con Razón Dorada y Forsythe y Luenberger (para $m = 3$), utilizando un valor de $\varepsilon = 1 \times 10^{-6}$.

k	x_k	y_k	$f(x,y)$
1	-0.809017	-0.190982	-0.405384
2	-0.677837	-0.059803	-0.426608
3	-0.717333	-0.020307	-0.428615
4	-0.703751	-0.006725	-0.428852
5	-0.708231	-0.002246	-0.428878
6	-0.706732	-7.483517E-4	-0.428881
7	-0.707231	-2.495093E-4	-0.428881
8	-0.707065	-8.315781E-5	-0.428881
9	-0.707120	-2.772337E-5	-0.428881
10	-0.707102	-9.237318E-6	-0.42881
11	-0.707108	-3.081763E-6	-0.428881
12	-0.707106	-1.023345E-6	-0.428881
13	-0.707106	-3.402927E-7	-0.428881
14	-0.707106	-1.130393E-7	-0.428881

Tabla 2.1: solución para la función $f(x) = xe^{(-x^2-y^2)}$ utilizando descenso de gradiente

k	x_k	y_k	$f(x,y)$
1	-0.703257	0.0031191	-0.428865
2	-0.707106	-3.7149863E-7	-0.428881

Tabla 2.2: solución para la función $f(x) = xe^{(-x^2-y^2)}$ utilizando descenso de gradiente utilizando la mejora de Forsyte–Luenberger

La función a minimizar es $f(x,y) = xe^{-x^2-y^2}$, con un punto inicial $x_0 = -0.5$, $y_0 = -0.5$

Los resultados para la Búsqueda lineal con muestreo hacia atrás refinando con Razón Dorada se muestran en la tabla 2.1 y para el algoritmo mejorado de Forsythe y Luenberger en la tabla 2.2. Note que el algoritmo de Forsythe y Luenberger converge en dos iteraciones para considerando la misma precisión que el descenso de gradiente normal.

Tarea 6

Algoritmo de Forsyte

Replicar la solución del ejemplo anterior (2.9.4)utilizando el algoritmo de descenso de gradiente 7 y de Forsyte 8.

2.10. Descenso de gradiente con búsqueda lineal basada en condiciones de Wolfe

Una especificación formal de la búsqueda lineal es la siguiente. Nos referiremos a la ecuación 2.7 como la condición de suficiente decrecimiento y a la ecuación 2.8 como la condición de curvatura. El algoritmo de búsqueda lineal determina cual ancho de paso α^*, satisface las condición de Wolfe.

Algoritmo 9 Algoritmo de Búsqueda lineal basado en condiciones de Wolfe

BÚSQUEDA_LINEAL$(f(x), x_0)$

1 `Dado una función diferenciable` $f(x)$ `y un valor inicial` x_0
2 `Hacer` $\alpha_0 = 0$, `seleccionar` $\delta\alpha > 0$ `y un` α_{max}
3 **para** $i \leftarrow 1$ `hasta` MAX
4 {
5 `Evaluar` $\phi(\alpha_i)$
6 **si** $\phi(\alpha_i) > \phi(0) + c_1 * \alpha_i\phi'(0)$ `O` $\phi(\alpha_i) \geq \phi(\alpha_{i-1})$
7 **entonces** `hacer` $\alpha^* \leftarrow RanzonDorada(\alpha_{i-1}, \alpha_i)$ `y finalizar`
8 `Evalúa` $\phi'(\alpha_i)$
9 **si** $\mid \phi'(\alpha_i) \mid > -c_2\phi'(0)$
10 **entonces** $\alpha^* \leftarrow \alpha_i$ `y termina`
11 **si** $\phi'(\alpha_i) \geq 0$
12 **entonces** $\alpha^* = RazonDorada(\alpha_{i-1}, \alpha_i)$ `y finalizar`
13 `Seleccionar` $\alpha_{i+1} = \alpha_i + \delta\alpha$
14 `hacer` $i \leftarrow i + 1$
15 }
16 **regresar** α^*

2.10.1. Ejemplo

Una comparacion es necesaria, para evaluar las condiciones Wolfe. En este caso utilizaremos la funcion de Ronsebrock dada como $f(x) = 100 * (x_2 - x_1^2)^2 + (1 - x_1)^2$. Para que sea interesante veamos el comportamiento cuando queremos calcular un minimo utilizando $f(x)$ directamente y un máximo con una función alterna $g(x) = -f(x)$.

En la figura 2.16 y la tabla 2.3 podemos ver el comportamiento de los algoritmos de maximo descenso utilizando tamano de paso de maximo descenso y tamaño de paso con condiciones de wolfe para un punto inicial $x_0 = [2, 3]^T$. Note que ambos algoritmos tienen un desempeno muy similar cuando minimizamos la funcion $f(x)$. En la tabla 2.3 se presenta en la primer columna el α de máximo descenso, el la segunda columna el valor de la función, el la tercer columna el α con condiciones de wolfe y en la última columna el valor de la función.

Tabla 2.3: Comparacion de iteraciones entre tamano de paso α de MD y Wolfe para minimizar $f(x)$

k	MD		Wolfe	
	α	$f(x)$	α	$f(x)$
0	0.0000	101.00	0.0000	101.0000
1	0.2187	2.6682	0.2582	0.5620
2	0.0383	0.5641	0.0750	0.5473
3	0.0011	0.5623	0.0032	0.5314
4	0.0094	0.5604	5.2786E-4	0.5313
5	0.0012	0.5584	5.2786E-4	0.5311
6	0.0106	0.5562	5.2786E-4	0.5310
7	0.0013	0.5540	5.2786E-4	0.5309
8	0.0123	0.5514	5.2786E-4	0.5308
9	0.0013	0.5489	5.2786E-4	0.5306

En el caso de la figura 2.17 y la tabla 2.4 es muy diferente. Aqui calcular el tamaño de paso con máximo descenso lleva al algoritmo a un máximo mientras que calcular el tamano de paso con condiciones de wolfe va al mínimo.

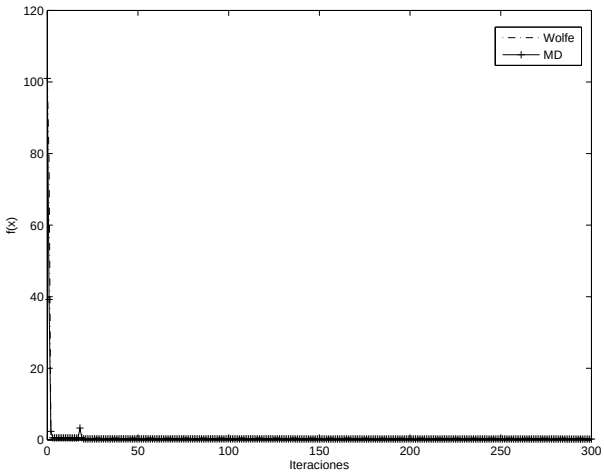


Figura 2.16: Comportamiento de Maximo descenso y Maximo descenso con condiciones de wolfe

Tabla 2.4: Comparacion de iteraciones entre tamano de paso utilizando α de MD y Wolfe para minimizar $-f(x)$

k	MD		Wolfe	
	α	$f(x)$	α	$f(x)$
0	0.0000	-101.0000	0.0000	-101.0000
1	-0.2187	-2.6682	0.0010	-101.8284
2	-0.0383	-0.5641	0.0010	-102.6606
3	-0.0011	-0.5623	0.0010	-103.4967
4	-0.0094	-0.5604	0.0010	-104.336
5	-0.0012	-0.5584	0.0010	-105.180
6	-0.0106	-0.5562	0.0010	-106.027
7	-0.0013	-0.5540	0.0010	-106.878
8	-0.0123	-0.5514	0.0010	-107.733
9	-0.0013	-0.5489	0.0010	-108.592

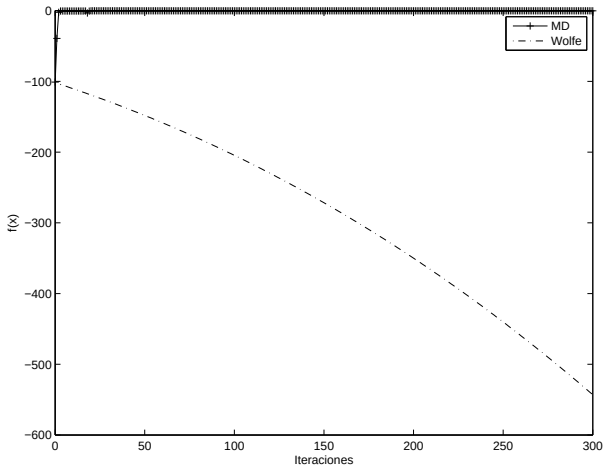


Figura 2.17: Comportamiento de Maximo descenso y Maximo descenso con condiciones de wolfe

Capítulo 3

Método del gradiente conjugado

El método del gradiente conjugado es un método iterativo para solucionar un sistema lineal de ecuaciones de la forma $Ax = b$; donde A es una matriz definida positiva. Este problema es equivalente al siguiente problema de minimización

$$\phi(x) = \frac{1}{2}x^T Ax - b^T x$$

3.1. Método de direcciones conjugadas

Una de las propiedades más importantes del método de gradiente es su habilidad de generar direcciones conjugadas de una manera muy económica. Un conjunto de vectores $\{p_0, p_1, \ldots p_l\}$ es llamado conjugado respecto a una matriz definida positiva A, si :

$$p_i{}^T A p_j = 0 \qquad \forall i \neq j$$

La importancia de las direcciones conjugadas radica en el hecho de que podemos minimizar $\phi(\cdot)$ en n pasos por la sucesiva minimización a lo largo de direcciones individuales en un conjunto conjugado. Para verificar esto, consideremos el siguiente método de direcciones conjugadas.

Dado un punto inicial $x_0 \in R^n$ y un conjunto de direcciones $\{p_0, p_1, \ldots, p_{n-1}\}$. Consideraremos la siguiente secuencia generadora $\{x_K\}$

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

donde α_k es el minimizador de la función cuadrática $\phi(\cdot)$ a lo largo de $x_k + \alpha p_k$ dada explícitamente por

$$\alpha_k = -\frac{r_k^T p_k}{{p_k}^T A p_k}$$

con $r_k = \nabla\phi(x) = Ax - b$

Teorema

Para cualquier $x_0 \in R^n$ la secuencia $\{x_k\}$ generada por el algoritmo de direcciones conjugadas converge a la solución x^* de un sistema lineal en no más de n pasos.

$$x^* - x_0 = \sigma_0 p_0 + \sigma_1 p_1 + \ldots + \sigma_{n-1} p_{n-1}$$

para algún escalar σ_k. Pre–multiplicando la expresión por ${p_k}^T A$ y utilizando la propiedad del conjugado tenemos:

$$\sigma_k = \frac{{p_k}^T A(x^* - x_0)}{{p_k}^T A_{p_k}}$$

Ahora estableceremos el resultado mostrando que este coeficiente coincide con la longitud de paso α_k.

Si x_k es generado por la sucesión $x_{k+1} = x_k + \alpha_k P_k$ tenemos:

$$x_k = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \ldots + \alpha_{k-1} p_{k-1}$$

Pre multiplicando ésta expresión por ${p_k}^T A$ y utilizando la propiedad del conjugado tenemos:

$${p_k}^T A(x_k - x_0) = 0$$

Adicionalmente

$${p_k}^T A(x^* - x_k) = {p_k}^T A(\alpha_{k+1} p_{k+1} + \alpha_{k+1} p_{k+1} + \ldots + \alpha_{n-1} p_{n-1})$$

tenemos que ${p_k}^T A(x^* - x_k) = {p_k}^T A(x^* - x_k)$ y por lo tanto:

$${p_k}^T A(x^* - x_0) = {p_k}^T A(x^* - x_k) = {p_k}^T (b - Ax_k) = -{p_k}^T r_k$$

comparando ambas relaciones tenemos que $\alpha_k = \sigma_k$.

Hay una interpretación simple de las direcciones conjugadas. Si la matriz A es diagonal los contornos de la función $\phi(\cdot)$ son elipses cuyos ejes están alineados con los ejes

coordenados. Podemos encontrar el mínimo de ésta función con una búsqueda lineal a lo largo de los ejes coordenados.

Cuando A no es una diagonal, sus contornos son elípticos, pero ellos no están alineados en la dirección de los ejes coordenados. La estrategia de minimizaciones sucesivas a lo largo de estas direcciones no da una solución en n iteraciones.

En la figura 3.1, podemos ver como se ejemplifica el uso de las direcciones conjugadas.

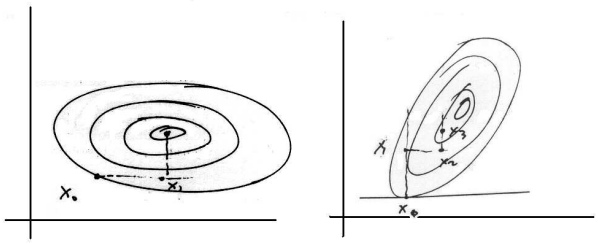


Figura 3.1: Ejemplo de como trabajan las direcciones conjugadas

3.2. El método del gradiente conjugado para funciones cuadráticas

Asumiremos que minimizaremos una función cuadrática de la forma:

$$\phi(x) = \frac{1}{2}x^T Ax + bx + c \tag{3.1}$$

La idea del método es construir progresivamente direcciones $p_0, p_1, \ldots, p_K$, las cuales son conjugadas respecto a la matriz A de la forma cuadrática. En cada estado K la dirección P_K es obtenida por la combinación lineal del gradiente $-\nabla q(x^K)$ en cada posición x_K y las direcciones previas $p_0, p_1, \ldots, p_{K-1}$.

Llamemos $q_k = \nabla q(x_k)$ el gradiente de la función q en x_k^l, El método toma la siguiente forma que se muestra en 10.

El algoritmo de gradiente conjugado, requiere de la siguiente sucesión para calcular las direcciones conjugadas $p_{k+1} = -r_{k+1} + \beta_{k+1}p_k$ con un parámetro $\beta_{k+1} = \frac{{r_{k+1}}^T Ap_k}{p_k^T Ap_k}$, pero debemos mostrar que estas expresiones crean direcciones conjugadas.

Algoritmo 10 Algoritmo de Gradiente Conjugado preliminar

GRADIENTE CONJUGADO PRELIMINAR$(f(x), x_0)$

1 `Dado una función diferenciable` $f(x)$ `y un valor inicial` x_0
2 `Calcular el gradiente` $r_0 = \nabla f(x_0) = Ax_0 - b$
3 `Hacer hacer` $p_0 = -r_0$ `y poner` $k = 0$.
4 **repetir**
5 `Calcular` $\alpha_k = -\frac{r_k^T p_k}{p_k{}^T A p_k}$
6 `Hacer la actualización` $x_{k+1} = x_k + \alpha_k p_k$
7 `Calcular el valor del gradiente` $r_{k+1} = Ax_{k+1} - b$
8 `determinar` $\beta_{k+1} = \frac{r_{k+1}{}^T A p_k}{p_k{}^T A p_k}$
9 `Calcular la nueva dirección con` $p_{k+1} = -r_{k+1} + \beta_{k+1} p_k$
10 `Actualizar` $k = k + 1$
11 **hasta** $\|\nabla f(x_k)\| \leq \varepsilon$
12 **regresar** x_k

Prueba

Por definición tenemos que las direcciones conjugadas cumplen con 3.2:

$$p_{k+1}{}^T A p_k = 0 \tag{3.2}$$

y tenemos que:

$$p_{k+1} = -r_{k+1} + \beta_{k+1} p_k \tag{3.3}$$

Sustituyendo 3.3 en 3.2:

$$(-r_{k+1} + \beta_{k+1} p_k)^T A p_k = 0$$

$$(-r_{k+1}{}^T + \beta_{k+1} p_k{}^T) A p_k = 0$$

$$-r_{k+1}{}^T A p_k + \beta_{k+1} p_k{}^T A p_k = 0$$

$$\beta_{k+1} = \frac{r_{k+1}{}^T A p_k}{p_k{}^T A p_k}$$

Con lo cual se muestra que la sucesión si produce direcciones conjugadas.

Una expresión equivalente para el ancho de paso es:

$$\alpha_k = \frac{r_k{}^T r_k}{p_k{}^T A p_k}$$

mostrar que es equivalente a:

$$\alpha_k = -\frac{r_k{}^T p_k}{p_k{}^T A p_k}$$

Prueba:

Tenemos que $p_k = -r_k + \beta_k p_{k-1}$ podemos escribir:

$$\alpha_k = -\frac{r_k{}^T p_k}{p_k{}^T A p_k}$$

$$\alpha_k = -\frac{r_k{}^T (-r_k + \beta_k p_{k-1})}{p_k{}^T A p_k}$$

$$\alpha_k = \frac{r_k{}^T r_k}{p_k{}^T A p_k} - \beta_k \frac{r_k{}^T p_{k-1}}{p_k{}^T A p_k}$$

Dado que $(p_0, p_1, \cdots, p_{k-1})$ son mutuamente conjugados x_k es el mínimo de $\phi(x)$ en la dirección p_{k-1} así que:

$$r_k{}^T p_{k-1} = 0$$

Con lo cual

$$\alpha_k = \frac{{r_k}^T r_k}{{p_k}^T A p_k}$$

Una expresión equivalente para β_{k+1} es:

$$\beta_{k+1} = \frac{{r_{k+1}}^T r_{k+1}}{{r_k}^T r_k} = \frac{{r_{k+1}}^T A p_k}{{p_k}^T A p_k}$$

Recordemos que

$$r_k = A x_k - b$$

Note que:

$$r_{k+1} - r_k = A\left(x_{k+1} - x_k\right) = \alpha_k A p_k \tag{3.4}$$

Si multiplicamos por ${r_{k+1}}^T$ la ecuación 3.4 y despejamos, tenemos:

$${r_{k+1}}^T A p_k = \frac{1}{\alpha_k} {r_{k+1}}^T \left(r_{k+1} - r_k\right)$$

Sustituyendo el valor de α_k

$${r_{k+1}}^T A p_k = \left(\frac{{p_k}^T A p_k}{{r_k}^T r_k}\right) {r_{k+1}}^T \left(r_{k+1} - r_k\right)$$

reordenando términos tenemos

$$\frac{{r_{k+1}}^T A p_k}{p_k^T A p_k} = \frac{{r_{k+1}}^T \left[r_{k+1} - r_k\right]}{{r_k}^T r_k}$$

Lo cual finalmente demuestra que

$$\beta_{k+1} = \frac{{r_{k+1}}^T r_{k+1}}{{r_k}^T r_k}$$

3.2.1. Algoritmo de GC

El algoritmo 11 presenta los detalles para la implementación del Método de Gradiente Conjugado.

Algoritmo 11 Algoritmo de Gradiente Conjugado

GRADIENTE CONJUGADO$(f(x), x_0)$

1 `Dado una función diferenciable` $f(x)$ `y un valor inicial` x_0
2 `Calcular el gradiente` $r_0 = \nabla f(x_0) = Ax_0 - b$
3 `Hacer hacer` $p_0 = -r_0$ `y poner` $k = 0$.
4 **repetir**
5 `Calcular` $\alpha_k = \frac{r_k^T r_k}{p_k{}^T A p_k}$
6 `Hacer la actualización` $x_{k+1} = x_k + \alpha_k p_k$
7 `Calcular el valor del gradiente` $r_{k+1} = r_k + \alpha_k A p_k$
8 `determinar` $\beta_{k+1} = \frac{r_{k+1}{}^T r_{k+1}}{r_k{}^T r_k}$
9 `Calcular la nueva dirección con` $p_{k+1} = -r_{k+1} + \beta_{k+1} p_k$
10 `Actualizar` $k = k + 1$
11 **hasta** $\|r_k\| \leq \varepsilon$
12 **regresar** x_k

3.2.2. Ejemplo

Muestre que la siguiente sistema de ecuaciones tiene una matriz definida positiva. Determine la solución del sistema de ecuaciones, utilizando gradiente conjugado.

$$\begin{pmatrix} 3 & -1 & -2 \\ -1 & 4 & -3 \\ -2 & -3 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 3 \end{pmatrix}$$

Para determinar si se trata de una matriz definida positiva, calculamos su determinante

$$\det \begin{pmatrix} 3 & -1 & -2 \\ -1 & 4 & -3 \\ -2 & -3 & 6 \end{pmatrix} = 11$$

$\therefore$ tenemos una matriz definida positiva y se puede aplicar el método de Gradiente conjugado.

La solución aplicando el proceso iterativo es:

Iteración 1

$r_0 = [-1, 0, -3]^T$

$p_0 = [1, 0, 3]^T$

$r_0^T r_0 = 10$

$Ap_0 = [-3, -10, 16]^T$

$\alpha_0 = 0.2222$

$x_1 = [0.2222, 0, 0.6667]^T$

$r_1 = [1.6667, -2.2222, 0.5556]^T$

$r_1^T r_1 = 8.0247$

$\beta_1 = 8.0247/10 = 0.80247$

$p_1 = [2.4691, 2.2222, 1.8519]^T$

Iteración 2

$Ap_2 = [1.4815, 0.8642, -0.4938]^T$

$\alpha_2 = 1.7206$

$x_2 = [4.4706, 3.8235, 3.8529]^T$

$r_2 = [0.8824, -0.7353, -0.2944]^T$

$r_2^T r_2 = 1.4057$

$\beta_2 = 1.4057/10 = 0.1752$

$p_2 = [-0.4498, 1.1246, 0.6185]^T$

Iteración 3

$Ap_3 = [-3.7111, 3.0926, 1.2370]^T$

$\alpha_3 = 0.2378$

$x_3 = [4.3636, 4.0909, 4]^T$

$r_3 = [0.0111, 0.0555, -0.1443] \times 10^{-14}$

$r_3^T r_3 = 2.4036 \times 10^{-30}$

$\beta_3 = 1.7099 \times 10^{-30}$

$p_3 = [0.0111, 0.0555, -0.1443] \times 10^{-14}$

Finalmente la solución es:

$x = [4.3636, 4.0909, 4]$

3.3. Precondicionamiento

Podemos acelerar el método del gradiente conjugado con una transformación C sobre los valores de la matriz A. la clave de este proceso, el cual es conocido como precondicionamiento, es cambiar x a $\hat{x}$ via una matriz no singular C tal que

$$\hat{x} = Cx \therefore x = C^{-1}\hat{x}$$

La función cuadrática es transformada a

$$\phi(\hat{x}) = \frac{1}{2}\hat{x}^T(C^{-T}AC^{-1})\hat{x} - (C^{-T}b)^T\hat{x}$$

Si aplicamos una estrategia de minimización tenemos

$$(C^{-T}AC^{-1})\hat{x} = C^{-T}b$$

Tomando en cuenta que tenemos una ecuación de segundo grado la cual minimizamos en la dirección opuesta al gradiente tenemos que $f(x) = \frac{1}{2}x^TAx - b^Tx$, el gradiente esta dado por $\nabla f(x) = Ax - b$. Considerando que pre-multiplicamos por una matriz M^{-1} nuestra función $f(x)$ el resultado no se ve afectado en el mínimo y si $M^{-1} \approx A$, la solución la podemos calcular en una sola iteración de Gradiente conjugado .

$$\begin{aligned} \hat{f}(x) &= M^{-1}(\frac{1}{2}x^T Ax - b^T x) \\ \nabla\hat{f}(x) &= M^{-1}(Ax - b) \end{aligned}$$

La manera de calcular el $y_k = \nabla\hat{f}(x_k)$ será simplemente resolviendo el sistema de ecuaciones $Mr_k = y_k$ donde $r_k = \nabla f(x_k) = Ax_k - b$

Calculo de las direcciones:

Para hacer las actualizaciones de las direcciones haremos una combinación lineal entre el residuo precondicionado y_{k+1} y la dirección anterior p_k, así la nueva dirección la calcularemos como

$$p_{k+1} = -y_{k+1} + \beta_{k+1}p_k \tag{3.5}$$

Tamaño de Paso:

El tamaño de paso para el método de Gradiente Conjugado Precondicionado para una función cuadrática $\hat{\phi}(\alpha) = M^{-1}(\frac{1}{2}(x_k + \alpha_k p_k)^T A(x_k + \alpha_k p_k) - b^T(x_k + \alpha_k p_k))$ es

$$\alpha_k = -\frac{r_k^T p_k}{{p_k}^T A p_k}$$

sustituimos el valor de (3.5) en el tamaño de paso α_k obtenemos

$$\begin{aligned} \alpha_k &= -\frac{r_k^T(-y_k + \beta_k p_{k-1})}{{p_k}^T A p_k} \\ \alpha_k &= \frac{r_k^T y_k}{p_k^T A p_k} - \beta_k \frac{r_k^T p_{k-1}}{p_k^T A p_k} \end{aligned}$$

Dado que $r_k^T p_{k-1} = 0$ tenemos que el tamaño de paso es:

$$\alpha_k = \frac{r_k^T y_k}{{p_k}^T A p_k}$$

Cálculo del gradiente:

Nuestro nuevo vector de gradiente precondicionado y_k esta definido como $y_k = M^{-1}r_k$, para el calculo de r_k hacemos

$$r_{k+1} = Ax_{k+1} - b = A(x_k + \alpha_k p_k) - b = (Ax_k - b) + \alpha_k A p_k$$

finalmente tenemos que

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

$$M y_{k+1} = r_k + \alpha_k A p_k$$

Cálculo del tamaño de paso para las direcciones:

A partir de la ecuación 3.5 y asumiendo que las direcciones están conjugadas podemos escribir

$$(-y_{k+1} + \beta_{k+1} p_k)^T A p_k = 0$$

$${y_{k+1}}^T A p_k + \beta_{k+1} {p_k}^T A p_k = 0$$

despejando tenemos

$$\beta_{k+1} = \frac{{y_{k+1}}^T A p_k}{{p_k}^T A p_k}$$

A partir de la ecuación $r_{k+1} - r_k = \alpha_k A p_k$, si multiplicamos por ${y_{k+1}}^T$ y despejamos, tenemos:

$${y_{k+1}}^T A p_k = \frac{1}{\alpha_k} {y_{k+1}}^T (r_{k+1} - r_k)$$

Sustituyendo el valor de α_k

$${y_{k+1}}^T A p_k = \left(\frac{{p_k}^T A p_k}{{r_k}^T y_k} \right) {y_{k+1}}^T (r_{k+1} - r_k)$$

reordenando términos tenemos

$$\frac{{y_{k+1}}^T A p_k}{p_k^T A p_k} = \frac{{y_{k+1}}^T [r_{k+1} - r_k]}{{r_k}^T y_k} = \frac{r_{k+1}^T y_{k+1}}{r_k^T y_k} - \frac{y_{k+1}^T r_k}{r_k^T y_k}$$

y dado que $y_{k+1}^T r_k = M^{-T} r_{k+1}^T r_k = 0$ finalmente tenemos

$$\beta_{k+1} = \frac{{r_{k+1}}^T y_{k+1}}{{r_k}^T y_k}$$

El método de gradiente conjugado utilizando precondicionado se presenta en el algoritmo 12

Algoritmo 12 Algoritmo de Gradiente Conjugado Precondicionado

GRADIENTE CONJUGADO PRECONDICIONADO$(f(x), x_0)$

1 Dado una función diferenciable $f(x)$, un valor inicial x_0
2 y una matriz de precondicionamiento M
3 Calcular el gradiente $r_0 = \nabla f(x_0) = Ax_0 - b$
4 Resolver $My_0 = r_0$ para y_0
5 Hacer hacer $p_0 = -y_0$ y $k = 0$.
6 **repetir**
7 Calcular $\alpha_k = \frac{r_k^T y_k}{{p_k}^T A p_k}$
8 Hacer la actualización $x_{k+1} = x_k + \alpha_k p_k$
9 Calcular el valor del gradiente $r_{k+1} = r_k + \alpha_k A p_k$
10 Resolver para y_{k+1}, el sistema de ecuaciones $My_{k+1} = r_{k+1}$
11 determinar $\beta_{k+1} = \frac{{r_{k+1}}^T y_{k+1}}{{r_k}^T y_k}$
12 Calcular la nueva dirección con $p_{k+1} = -y_{k+1} + \beta_{k+1} p_k$
13 Actualizar $k = k + 1$
14 **hasta** $\|r_k\| \leq \varepsilon$
15 **regresar** x_k

Observe que en el caso del que $C = I$ el algoritmo 12 es equivalente al método del Gradiente Conjugado. Por esta razón es importante calcular una matriz C que acerque a la solucón final, reduciendo las iteraciones y/o tiempo necesario para que el algoritmo de Gradiente conjugado converga

3.3.1. Precondicionadores Prácticos

El algoritmo de Cholesky, se basa en el hecho de que una matriz simétrica y definida positiva se puede factoriza en dos matrices L tal que

$$A = LL^T$$

donde los factores triangulares de A son L y L^T.
La formula para calcular la descomposicón de Cholesky en el caso de los elementos fuera de la diagonal es:

$$l_{k,i} = \frac{a_{k,i} - \sum_{j=1}^{i-1} l_{i,j} l_{k,j}}{l_{i,i}} \tag{3.6}$$

$$\forall k = 0, 1, 2, \cdots, N-1 \tag{3.7}$$

$$\forall i = 0, 1, 2, \cdots, k-1$$

y para los elementos en la diagonal

$$l_{k,k} = \sqrt{a_{k,k} - \sum_{j=1}^{k-1} l_{k,j}^2} \tag{3.8}$$

El código Java para realizar esta implementación se muestra a continuación

```
public Matriz Cholesky3() {
  int i, j, k, n, s;
  double fact, suma = 0;

  if (this.nren != this.ncol || this == null) {
    System.out.println("Matriz mal condicionada");
    return null;
  }
  n = this.nren;
  for (k = 0; k < n; k++) {
    for (i = 0; i <= k - 1; i++) {
      suma = 0;
      for (j = 0; j <= i - 1; j++)
```

```
          suma += this.datos[i][j] * this.datos[k][j];

        this.datos[k][i] = (this.datos[k][i] - suma) / this.datos[i][i];
      }
      suma = 0;
      for (j = 0; j <= k - 1; j++)
        suma += this.datos[k][j] * this.datos[k][j];
      this.datos[k][k] = Math.sqrt(this.datos[k][k] - suma);
    }
    return this;
  }
```

Choleski incompleto es probablemente la estrategia de precondicionamiento mas efectiva. La idea básica es muy simple. Seguiremos el procedimiento de Choleski, pero en lugar de calcular el valor exacto de L que satisface $A = LL^T$, calcularemos un valor de $\hat{L}$ que es tan dispersa como A. Tendremos entonces $A \approx \hat{L}\hat{L}^T$ y seleccionando $C = \hat{L}^T$, obtendremos $A = \hat{L}\hat{L}^T$ y

$$C^T A C^{-1} = \hat{L}^{-1} A \hat{L}^{-T} \approx I$$

En el caso de el Cholesky incompleto aplicaremos las formulas (3.7) y (3.8) solo en el caso que el $a_{i,j} \neq 0$. El código Java para esta implementación es:

```
public static void CholeskyI(double A[][]) {
    int i, j, k, n, s;
    double fact, suma = 0;
    n = A.length;

    for (i = 0; i < n; i++) {
        for (j = 0; j <= i - 1; j++) {
            if (A[i][j] != 0) {
                System.out.print("A"+i+"," + j +"= ");
                suma = 0;
                for (k = 0; k <= j - 1; k++) {
                    suma += A[i][k] * A[j][k];
                    if(A[i][k]*A[j][k] != 0) System.out.print ("A"+i+"," + k + " * A" +j +","+k  + " ");
                }
                A[i][j] = (A[i][j] - suma) / A[j][j];
                System.out.println(" ");
            }
        }
        suma = 0;
        for (k = 0; k <= i - 1; k++) {
            suma += A[i][k] * A[i][k];
            if(A[i][k]*A[i][k] != 0) System.out.println("si");
        }
        A[i][i] = Math.sqrt(A[i][i] - suma);
    }

    for (j = 0; j < n; j++) {
        for (k = 1 + j; k < n; k++)
            A[j][k] = 0;
    }
}
```

La solución del sistema factorizado $LL^T x = b$ lo realizaremos en dos pasos, para ello hacemos $L^T x = y$. El el primer paso resolvemos $Ly = b$ para y y en el segundo paso resolvemos el sistema $Lx = y$ para x. La solución de estos sistemas la haremos utilizando Sustitución hacia adelante (3.9) y sustitución hacia atras (3.10)

$$y_i = \frac{b_i - \sum_{k=0}^{k=i-1} l_{i,k} y_i}{l_{i,i}} \qquad i = 0, 1, \cdots, N-1 \tag{3.9}$$

$$x_i = \frac{y_i - \sum_{k=0}^{k=i-1} l_{i,k} y_i}{l_{i,i}} \qquad i = N-1, \cdots, 1, 0 \tag{3.10}$$

3.3.2. Ejemplo

Dado el sistema de de ecuaciones $Ax = b$ calcular la solución utilizando

$$\begin{bmatrix} 3 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 3 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 3 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 2 \\ 1 \\ 2 \\ 3 \\ 2 \end{bmatrix}$$

a) Utilizando la factorizacón de Choleky

La matriz L factorizada queda como

$$L = \begin{bmatrix} 1.7321 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ -0.5774 & 1.9149 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & -0.5222 & 1.9306 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & -0.5180 & 1.6528 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ -0.5774 & -0.1741 & -0.0471 & -0.0148 & 1.6229 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & -0.5222 & -0.1413 & -0.0443 & -0.6767 & 1.8021 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & -0.5180 & -0.1623 & -0.0165 & -0.6057 & 1.8271 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & -0.6050 & -0.0055 & -0.0169 & -0.6067 & 1.5052 \end{bmatrix}$$

Aplicando sustitución hacia atras solucionamos el sistema $Ly = b$, para y

$$y = [0.5774, 1.2185, 1.8835, 1.8004, 1.0233, 2.0391, 3.0211, 3.297]^T$$

y finalmente resolvemos el sistema $L^T x = y$ para x

$$x = [1.4762, 1.9524, 2.381, 2.1905, 1.4762, 1.9524, 2.381, 2.1905]^T$$

b) La solución utilizando Cholesky incompleto.

Primero calculamos la factorización incompleta de Cholesky dada como

$$\widehat{L} = \begin{bmatrix} 1.7321 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -0.5774 & 1.9149 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & -0.5222 & 1.9306 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -0.518 & 1.6528 & 0.0 & 0.0 & 0.0 & 0.0 \\ -0.5774 & 0.0 & 0.0 & 0.0 & 1.633 & 0.0 & 0.0 & 0.0 \\ 0.0 & -0.5222 & 0.0 & 0.0 & -0.6124 & 1.8309 & 0.0 & 0.0 \\ 0.0 & 0.0 & -0.518 & 0.0 & 0.0 & -0.5462 & 1.8529 & 0.0 \\ 0.0 & 0.0 & 0.0 & -0.605 & 0.0 & 0.0 & -0.5397 & 1.5306 \end{bmatrix}$$

Aplicando sustitución hacia atras solucionamos el sistema $\widehat{L}y = b$ para y

$$y = [0.5774, 1.2185, 1.8835, 1.8004, 0.8165, 1.7130, 2.6505, 2.9529]^T$$

y finalmente resolvemos el sistema $\widehat{L}^T x = y$ para x

$$x = [1.2235, 1.5969, 1.9919, 1.7955, 1.0737, 1.5299, 1.9923, 1.9293]^T$$

Note que la solución es próxima a la solución del inciso a).

c) Calcular la solución utilizando el método de Gradiente Conjugado con un vector inicial $X^{(0)} = [10, 0, 0, 0, 0, 0, 0, 0]^T$. La solución en 7 iteraciones utilizando el método de GC fue

$$X_{GC} = [1.4762, 1.9524, 2.3810, 2.1905, 1.4762, 1.9524, 2.3810, 2.1905]^T$$

d) Calcular la solución utilizando el método de Gradiente Conjugado precondicionado con un vector inicial $X^{(0)} = [10, 0, 0, 0, 0, 0, 0, 0]^T$.

Utilizando GC precondicionado con Cholesky incompleto, en 4 iteraciones fue:

$$X_{GC-Pre} = [1.4762, 1.9524, 2.3810, 2.1905, 1.4762, 1.9524, 2.3810, 2.1905]^T$$

e) Haga una comparación entre el desempeño del GC y el GC precondicionado.

En la tabla 3.1 y en la figura 3.3 podemos ver un comparativo del desempeño de estos dos algoritmos para los datos del problema. Note como el método de Gradiente conjugado precondicionado alcanza el mínimo en menos iteraciones.

Tabla 3.1: Comparación entre el gradiente Conjugado y el Gradiente Conjugado Precondicionado

iteracion	G Conj	G Conj. Pre
0	290.0	290.0
1	23.10551046561298	-4.518689830799406
2	-0.5750827715810338	-0.024727932744468717
3	-2.2694980820399095	0.003288775934358057
4	-1.5700877025774922	3.622004717129812E-5
5	-0.9214604310248227	5.986190387829993E-8
6	-0.10251958033978781	-5.208278253121534E-12
7	7.105427357601002E-15	0.0
8	7.105427357601002E-15	0.0

ver ejemplo_regula.java

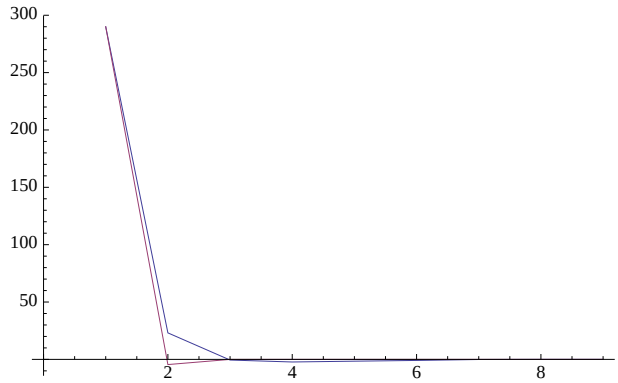


Figura 3.2: Comparación entre el método de GC y GC precondicionado.

3.4. Manejo de Dispersidad

En algunas ocaciones el minimizar una función, dado el número de dimensiones, implica que no es posible ni siquiera reservar una cantidad de memoria para almacenar el Hessiano. Sin embargo dado el número de ceros en el Hessiano es preferible no reservar una cantidad de memoria y trabajar con alguna técnica para el manejo de dispersidad.

Consideremos el caso de minimizar la función dada por

$$E(f) = \sum_{i=0}^{i=N-1} \sum_{j=0}^{j=N-1} \left[(f_{i,j} - g_{i,j})^2 + \lambda (f_{i,j} - f_{i-1,j})^2 + \lambda (f_{i,j} - f_{i,j-1})^2 \right]$$

Con un valor de $\lambda = 5$ y una senãl bidimensional g dada

$$g = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

El sistema lineal por resolver es $Af = b$ con

$$A = \begin{bmatrix} 11.0 & -5.0 & 0.0 & -5.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -5.0 & 16.0 & -5.0 & 0.0 & -5.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & -5.0 & 11.0 & 0.0 & 0.0 & -5.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -5.0 & 0.0 & 0.0 & 16.0 & -5.0 & 0.0 & -5.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & -5.0 & 0.0 & -5.0 & 21.0 & -5.0 & 0.0 & -5.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -5.0 & 0.0 & -5.0 & 16.0 & 0.0 & 0.0 & -5.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & -5.0 & 0.0 & 0.0 & 16.0 & -5.0 & 0.0 & -5.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & -5.0 & 0.0 & -5.0 & 21.0 & -5.0 & 0.0 & -5.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -5.0 & 0.0 & -5.0 & 16.0 & 0.0 & 0.0 & -5.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -5.0 & 0.0 & 0.0 & 11.0 & -5.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -5.0 & 0.0 & -5.0 & 16.0 & -5.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -5.0 & 0.0 & -5.0 & 11.0 \end{bmatrix}$$

y $b = [0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0]^T$. La solución es:

$$f = [0.1475, 0.1595, 0.1475, 0.165, 0.2155, 0.165, 0.165, 0.2155, 0.165, 0.1475, 0.1595, 0.1475]^T$$

Para este sistema podemos notar que muchos de los valores son cero.

La factorización de la matriz A es:

$$L = \begin{bmatrix} 3.3166 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -1.5074 & 3.705 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & -1.3494 & 3.0296 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -1.5074 & 0.0 & 0.0 & 3.705 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & -1.3494 & 0.0 & -1.3494 & 4.1662 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.6502 & 0.0 & -1.2 & 3.4403 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & -1.3494 & 0.0 & 0.0 & 3.7654 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & -1.2 & 0.0 & -1.3277 & 4.2185 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -1.4532 & 0.0 & -1.1851 & 3.5331 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -1.3277 & 0.0 & 0.0 & 3.0392 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -1.1851 & 0.0 & -1.645 & 3.4479 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -1.415 & 0.0 & -1.45 & 2.6257 \end{bmatrix}$$

La formula para calcular la descomposicón de Cholesky sobre los elementos fuera de la diagonal es

$$l_{k,i} = \frac{a_{k,i} - \sum_{j=1}^{i-1} l_{i,j} l_{k,j}}{l_{i,i}}$$

Podemos notar, dado que es una matriz bandada que los productos $l_{i,j} l_{k,j} = 0$ siempre son cero y los elementos $a_{k,i} = -\lambda$ por tal razón la ecuación para modificar los elementos fuera de las diagonales se puede expresar como

$$l_{k,i} = \frac{-\lambda}{l_{i,i}}$$

Dado la simplicidad de los elementos fuera de la diagonal, no los almacenaremos y en su lugar realizaremos este simple cálculo y simplimente almacenaremos los datos de la diagonal en una matriz

$$D = \begin{bmatrix} 11.0 & 16.0 & 16.0 & 11.0 \\ 16.0 & 21.0 & 21.0 & 16.0 \\ 16.0 & 21.0 & 21.0 & 16.0 \\ 11.0 & 16.0 & 16.0 & 11.0 \end{bmatrix}$$

Los elementos en la diagonal son actualizados por la ecuación :

$$l_{i,i} = \sqrt{a_{i,i} - \sum_{k=0}^{i-1} l_{i,k}^2}$$

Sustituyendo la expresion para los elementos fuera de la diagonal

$$l_{i,i} = \sqrt{a_{i,i} - \sum_{k=0}^{i-1} \frac{\lambda^2}{l_{k,k}^2}}$$

En nuestra representación matricial

$$D_{i,j} = \sqrt{D_{i,j} - \frac{\lambda^2}{D_{i-1,j}^2} - \frac{\lambda^2}{D_{i,j-1}^2}}$$

Aplicando la ecuaciones anteriores, los elementos de la diagonal factorizados son

$$LD = \begin{bmatrix} 3.3166 & 3.705 & 3.0296 \\ 3.705 & 4.1662 & 3.4403 \\ 3.7654 & 4.2185 & 3.5331 \\ 3.0392 & 3.4479 & 2.6257 \end{bmatrix}$$

Para realizar la solución del sistema tenemos que calcular la sustitución hacia atras y hacia adelante.

3.4.1. Sustitución hacia adelante

Como primer paso debemos resolver el sistema $Ly = b$ utilizando sustitución hacia adelante. La ecuación para llevar a cabo al sustitución hacia adelante esta dada por

$$y_i = \frac{b_i - \sum_{k=0}^{k=i-1} l_{i,k} y_i}{l_{i,i}}$$
$$i = 0, 1, 2, \cdots, N-1$$

Para nuestro sistema de ecuaciones tenemos:

$$y_{i,j} = \frac{b_{i,j} + \lambda * y_{i-1,j}/D_{i-1,j} + \lambda y_{i,j-1}/D_{i,j-1}}{D_{i,j}}$$

La implementación en Java queda como:

```
static public void SAdelante(double A[][], double x[][], double b[][], double lambda, int nr, int nc)
   {
     int i, j;
     double suma;

     for(i=0; i<nr; i++)
       for(j=0; j<nc; j++)
       {
         suma = 0;
         if(i-1 >= 0) suma += -lambda * x[i - 1][j] / A[i - 1][j];
         if(j-1 >= 0) suma += -lambda * x[i][j - 1] / A[i][j - 1];

         x[i][j] = (b[i][j] - suma)/A[i][j];
       }
   }
```

3.4.2. Sustitución hacia atras

Como segundo paso debemos resolver el sistema $L^T x = y$ utilizando sustitución hacia atras. La ecuación para llevar a cabo al sustitución hacia adelante esta dada por

$$x_k = \frac{y_k - \sum_{i=0}^{i=k-1} l_{k,i} x_i}{l_{k,k}}$$

$$k = N-1, N-2, \cdots, 2, 1, 0$$

Para nuestro sistema de ecuaciones tenemos:

$$x_{i,j} = \frac{b_{i,j} + \lambda * y_{i+1,j}/D_{i+1,j} + \lambda y_{i,j+1}/D_{i,j+1}}{D_{i,j}}$$

La implementación en Java queda como:

```
static public void SAtras(double A[][], double x[][], double b[][], double lambda, int nr, int nc)
{
int i, j;
double suma;

for(i=nr-1; i>=0; i--)
  for(j=nc-1; j>=0; j--)
  {
    suma = 0;
    if(i+1 < nr) suma += -lambda*x[i+1][j]/A[i][j];
    if(j+1 < nc) suma += -lambda*x[i][j+1]/A[i][j];
    x[i][j] = (b[i][j] - suma)/A[i][j];
  }
}
```

El resultado depués de aplicar el procedimiento completo es:

$$f = \begin{bmatrix} 0.0457 & 0.0467 & 0.0267 \\ 0.0538 & 0.1015 & 0.0491 \\ 0.0462 & 0.1033 & 0.0588 \\ 0.0278 & 0.0515 & 0.0501 \end{bmatrix}$$

3.4.3. Ejemplo de Filtrado de Imágenes

Como ejemplo considere que para una imágen queremos encontra un imagen filtrada. Para esto utilizaremos la función de energía

$$E(f) = \sum_{i=0}^{i=N-1} \sum_{j=0}^{j=N-1} \left[(f_{i,j} - g_{i,j})^2 + \lambda (f_{i,j} - f_{i-1,j})^2 + \lambda (f_{i,j} - f_{i,j-1})^2 \right]$$

En el caso de una imagen pequenã las dimensiones pueden ser 256×256. Considerando que es el número de variables de nuestro sistema tendremos que el Hessiano tiene que ser una matriz de dobles de tamaño $256^2 \times 256^2$, dando un total de memoria de 128 Giga Bytes. Una cantindad de memoria muy alta si consideramos que solamente necesitamos almacenar menos $0.007\,\%$ de variables diferentes de cero.

Los resultados aplicando Gradiente conjudado y Gradiente conjugado se muestran en la siguiente figura:

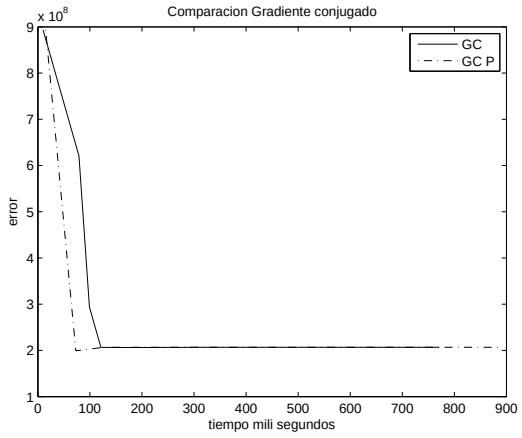


Figura 3.3: Comparación entre el método de GC y GC precondicionado para el caso del filtrado de imagenes.

Tarea 7

Características de precondicionamiento

Para el ejemplo anterior:

1. Comparar la solución para los métodos de descenso de gradiente, Gradiente Conjugado y Gradiente conjugado precondicionado
2. Mostrar en una gráfica el número de iteraciones de cada uno
3. Comparar la matriz de precondicionamiento calculada con Choleski Incompleto y la matriz Inversa.

ver (regulariza.java)

3.5. Gradiente conjugado no lineal

Hemos notado que le método de G.C., puede ser visto como un algoritmo de minimización de funciones cuadráticas convexas. Es natural preguntar que pasa cuando no tenemos una función no lineal.

Flecher y Reaves mostraron que una extensión de este es posible haciendo cambios simples al algoritmo de G.C. (ver [**?**, **?**]

Primero la longitud de paso

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}$$

debe ser reemplazada por una búsqueda en una dirección.

Segundo la manera de calcular el residuo debe ser reemplazada por el cálculo del gradiente.

Los cambios dan el siguiente algoritmo:

Para calcular α debemos considerar las condiciones de Wolfe

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k$$

$$|\nabla f(x_k + \alpha_k p_k)^T p_k| \leq c_2 |\nabla f_k^T p_k|$$

Algoritmo 13 Algoritmo de Gradiente Conjugado no lineal

GRADIENTE CONJUGADO NO LINEAL$(f(x), x_0)$

1 Dado una función diferenciable $f(x)$ y un valor inicial x_0
2 Hacer hacer $p_0 = -\nabla f(x_0)$ y poner $k = 0$.
3 **repetir**
4 Calcular α_k utilizando búsqueda lineal
5 Hacer la actualización $x_{k+1} = x_k + \alpha_k p_k$
6 Calcular el valor del gradiente $\nabla f(x_{k+1})$
7 determinar $\beta_{k+1}^{FR} = \frac{\nabla f(x_{k+1})^T \nabla f(x_{k+1})}{\nabla f(x_k)^T \nabla f(x_k)}$
8 Calcular la nueva dirección con $p_{k+1} = -\nabla f(x_{k+1}) + \beta_{k+1}^{FR} p_k$
9 Actualizar $k = k + 1$
10 **hasta** $\|\nabla f(x_k)\| \leq \varepsilon$
11 **regresar** x_k

3.5.1. El método de Polak - Ribiere

Existen muchas variantes del método de Flecher-Reeves que difieren principalmente en la selección del parámetro β_k. La más importante de estas variantes propuesta por Polak y Ribiere, define este parámetro como:

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

Un hecho sorprendente acerca del algoritmo PR-GR es que las condiciones de Wolfe no garantizan que p_k es siempre descendente en la dirección p_k. Si nosotros definimos el parámetro β como:

$$\beta_{k+1}^{+} = max\left\{\beta_{k+1}^{PR}, 0\right\}$$

Da lugar a un algoritmo que llamaremos PR+, entonces una simple adaptación de las condiciones de Wolfe encierra que la propiedad descendente se mantenga.

3.5.2. Ejemplo

Implementar el algoritmo de GC no lineal de Fletcher–Reeves FR y la mejora de Polak–Ribiere para la función $f(x) = x_1 * e^{(-x_1^2 - x_2^2)}$ comparar gráficamente los resultados. Considere un punto inicial $x = [-0.5, -0.5]^T$

Las soluciones utilizando los tres métodos es $x = [-0.707107, 0.000000]^T$. En la siguiente tabla, se muestra como el gradiente tiende a cero para los tres métodos.

k	DG	GC-PR+	GC-FR
0	0.42888	0.42888	0.42888
1	0.21898	0.21898	0.21898
2	0.07216	0.05840	0.05840
3	0.02462	0.00881	0.01740
4	0.00816	0.00005	0.00929
5	0.00273	0.00000	0.00181
6	0.00091	0.00000	0.00086
7	0.00030	0.00000	0.00029
8	0.00010	0.00000	0.00007
9	0.00003	0.00000	0.00004
10	0.00001	0.00000	0.00001
11	0.00000	0.00000	0.00000

note que para este ejemplo, el método de GC-PR+ tiene un mejor desempeño.

Capítulo 4

Métodos de Newton

4.1. Método de Newton

Consideremos que la función f es de clase dos, es decir que la segunda derivada puede ser calculada. La idea consiste en reemplazar en la vecindad del punto x^k de la función f por una aproximación cuadrática $q(x)$ dada por

$$q(x) = f(x_k) + \nabla f^T(x_k)(x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k)$$

llamaremos a x_{k+1} el mínimo de $q(x)$. Para calcularlo, es necesario que la matriz $\nabla^2 f(x_k)$ sea una matriz definida positiva. La función $q(x)$ es entonces estrictamente convexa y tiene un mínimo único en x_{k+1} dado por

$$\nabla_q(x_{k+1}) = 0$$

Esto da lugar sistema lineal de ecuaciones:

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

La solución para x_{k+1} es el mínimo, lo cual da lugar a la recurrencia

$$x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) \tag{4.1}$$

Comparando las ecuación 4.1 con la sucesión para el descenso de gradiente, podemos obsérvese que la dirección y el tamaño de paso es obtenido de manera óptima.

Algoritmo 14 Algoritmo de Newton

NEWTON$(f(x), x_0)$

1 `Dado una función diferenciable dos veces` $f(x)$ `y un valor inicial` x_0
2 **repetir**
3 $\quad x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} * \nabla f(x_k)$
4 $\quad k = k + 1$
5 **hasta** $\|\nabla f(x_k)\| \leq \varepsilon$
6 **regresar** x_k

Una propiedad importante de este método es que converge en un solo paso cuando se tiene una función estrictamente cuadrática.

Demostración:

Dada una función cuadrática

$$f(x) = \frac{1}{2}x^T Ax + x^T b + c$$

el gradiente lo podemos calcular como $\nabla f(x) = Ax + b$ y el Hessiano es $\nabla^2 f(x) = A$. Si sustituimos en la ecuación 4.1 tenemos

$$= x_k - A^{-1}(Ax_k + b)$$

$$x_{k+1} = x_k - x_k - A^{-1}b$$

Lo que finalmente da como resultado

$$x_{k+1} = -A^{-1}b$$

4.1.1. Ejemplo

Dada la función $f(x) = 10x_1^2 + x_2^2$ y un punto inicial $x_0 = [1, 2]^T$, calcular el mínimo utilizando elalgoritmo de Newton.

Para esta función el gradiente y el Hessiano es:

$$\nabla f(x) = \begin{array}{|c|}\hline 20x_1 \\ \hline 2x_2 \\ \hline \end{array}$$

y el Hessiano es:

$$A(x) = \begin{array}{|c|c|}\hline 20 & 0 \\ \hline 0 & 2 \\ \hline \end{array}$$

Aplicando la ecuación 4.1 tenemos

$$x_{k+1} = \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline \end{array} - \begin{array}{|c|c|}\hline 20 & 0 \\ \hline 0 & 2 \\ \hline \end{array}^{-1} \begin{array}{|c|}\hline 20 \\ \hline 4 \\ \hline \end{array} = \begin{array}{|c|}\hline 0 \\ \hline 0 \\ \hline \end{array}$$

4.1.2. Ejemplo

Minimizar es $f(x) = x_1 e^{-x_1^2 - x_2^2}$, con un punto inicial $x_1^{(0)} = -0.5$, $x_2^{(1)} = -0.1$ aplicando el Método de Newton.

k	Newton	DG	GC-PR+	GC-FR
0	0.39316	0.39316	0.39316	0.39316
1	0.02003	0.05006	0.05006	0.05006
2	0.00002	0.00930	0.00289	0.00289
3	0.00000	0.00091	0.00010	0.00019
4	0.00000	0.00017	0.00000	0.00002
5	0.00000	0.00002	0.00000	0.00000

Note que el mejor desempeño lo tiene el método de Newton para este problema.

4.2. Problemas de convergencia del Método de Newton

Si uno desea aplicar el método a una función arbitraria encontraremos algunas dificultades, esencialmente debido al hecho de que no encontraremos convergencia global. Si el punto inicial x^0 esta muy lejos de x^* el método no podrá converger.

4.2.1. Ejemplo

Minimizar es $f(x) = x_1 e^{-x_1^2 - x_2^2}$, con un punto inicial $x_1^{(0)} = -0.5$, $x_2^{(0)} = -0.5$ aplicando el Método de Newton.

k	Newton	DG	GC-PR+	GC-FR
0	0.42888	0.42888	0.42888	0.42888
1	0.30262	0.21898	0.21898	0.21898
2	0.04768	0.07216	0.05840	0.05840
3	0.01579	0.02462	0.00881	0.01740
4	0.00542	0.00816	0.00005	0.00929
5	0.00189	0.00273	0.00000	0.00181
6	0.00067	0.00091	0.00000	0.00086
7	0.00024	0.00030	0.00000	0.00029
8	0.00008	0.00010	0.00000	0.00007
9	0.00003	0.00003	0.00000	0.00004
10	0.00001	0.00001	0.00000	0.00001
11	0.00000	0.00000	0.00000	0.00000

Podemos ver que todos los algoritmos convergen, pero el caso de Newton la solución a la que llega es $x_{(20)} = [-2.33182, 4.39075]^T$, mientras que la solución con los otros métodos fue $x_{(20)} = [-0.70711, 0.00000]^T$. Esta condición se debe a que el punto inicial para el método de Newton esta muy lejos y la aproximación cuadrática, resulta in apropiada. En la figura 4.1 se muestra la función y se puede ver que esta tiene solamente un mínimo.

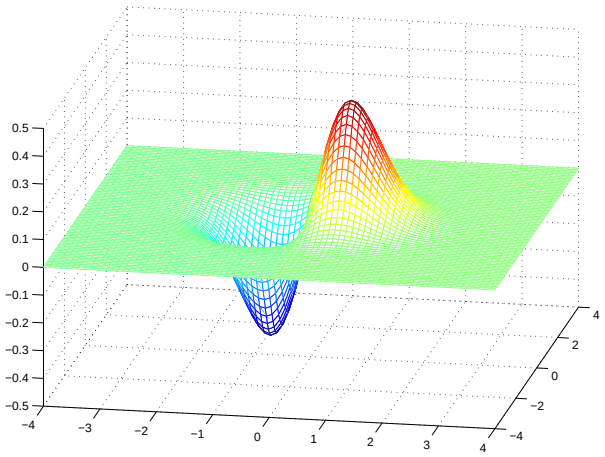


Figura 4.1: Función $f(x) = x_1 e^{-x_1^2 - x_2^2}$.

Para comenzar, la aproximación de $f(x)$ dada por $q(x)$ es solamente valida a la vecindad de x_k, el tamaño de paso puede ser controlado a través de una formula iterativa de tipo:

$$x_{k+1} = x_k - \lambda_k \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$$

donde λ_k es un escalar, seleccionado por ejemplo, de tal manera que $||x_{k+1} - x_k||$ no sea muy largo. También podemos seleccionarlo tal que λ_k minimice $\phi(\lambda_k) = f(x_k + \lambda_k d_k)$ en la dirección de:

$$d_k = -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$$

4.2.2. Ejemplo

Minimizar $f(x) = (x-3)^4$ utilizando el método de Newton y la modificación planteada utilizando un valor $\lambda_k = 2.9$. Considere un punto inicial $x_0 = 2$.

k	Newton	Newton con $\lambda = 2.9$
0	2.00000	2.00000
1	2.33333	2.96667
2	2.55556	2.99889
3	2.70370	2.99996
4	2.80247	3.00000
5	2.86831	3.00000
6	2.91221	3.00000
7	2.94147	3.00000
8	2.96098	3.00000
9	2.97399	3.00000
10	2.98266	3.00000
11	2.98844	3.00000

4.3. Método de Newton Modificado

Una alternativa propuesta por Ralston y Rabinowitz (1978) es la de definir una nueva función $u(x)$, que es el cociente del gradiente de $g(x)$ y su derivada (en una dimensión)

$$u(x) = \frac{g(x)}{g'(x)}$$

Se puede mostrar que esta función tiene las mismas raíces que $g(x)$ y que la multiplicidad de raíces no afectará. La formulación del método de Newton es:

$$x_{k+1} = x_k - \frac{u(x)}{u'(x)}$$

La derivada de u(x) es:

$$u'(x) = \frac{g'(x)g'(x) - g(x)g''(x)}{[g'(x)]^2}$$

Sustituyendo esta, tenemos la formulación final del Método de Newton modificado.

$$x_{k+1} = x_k - \frac{g'(x)g(x)}{g'(x)g'(x) - g(x)g''(x)}$$

Para el ejemplo anterior tenemos que con un valor inicial $x_0 = 2$ la solución es:

$$x_{k+1} = 2 - \frac{12 * (-4)}{12 * 12 - (-4) * (-24)} = 3$$

Note que la solución se da en una sola iteración.

4.4. Método de Broyden's o secante

Para este método consideraremos que los valores de x serán calculados utilizando:

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

donde $\nabla f(x_k)$ es el gradiente, pero en lugar de calcular A_k como la matriz Hessiana realizaremos una aproximación con secantes. Así pues:

$$A_k = \frac{\nabla f(x_k) - \nabla f(x_{k-1})}{x_k - x_{k-1}} \tag{4.2}$$

La restricción de la secante la podemos dar como

$$A_k(x_k - x_{k-1}) = \nabla f(x_k) - \nabla f(x_{k-1})$$

$$A_k s_k = y_k$$

donde $s_k = x_k - x_{k-1}$ y $y_k = \nabla f(x_k) - \nabla f(x_{k-1})$

Para determinar una forma iterativa de calcular A_k, calculamos para la figura 4.2, la ecuación de la linea recta que pasa en el punto P_0

$$l_0(x) = \nabla f(x_0) + A_0(x - x_0)$$

de igual manera en el punto P_1

$$l_1(x) = \nabla f(x_1) + A_1(x - x_1)$$

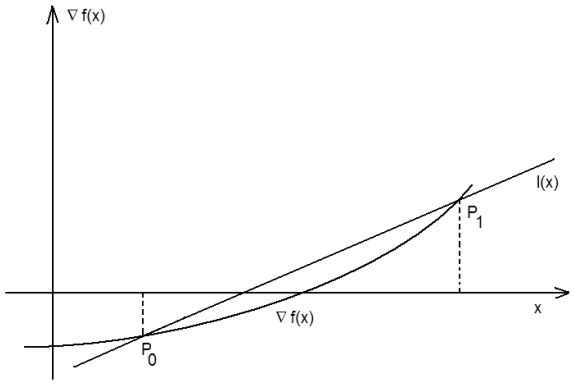


Figura 4.2: Gradiente con una aproximación de secantes

Es deseable que las dos líneas sean iguales y que con esto se cumpla que esta linea es secante. Así :

$$l_1(x) - l_0(x) = 0$$

sustituyendo los valores respectivos

$$\nabla f(x_1) + A_1(x - x_1) - \nabla f(x_0) - A_0(x - x_0) = 0$$

re agrupando términos tenemos

$$\nabla f(x_1) - \nabla f(x_0) + A_1(x - x_1) - A_0(x - x_0) = 0$$

Aplicando la definición de secante dada por la ecuación 4.2 tenemos:

$$\nabla f(x_1) - \nabla f(x_0) = A_1(x_1 - x_0)$$

sustituyendo

$$A_1(x_1 - x_0) + A_1(x - x_1) - A_0(x - x_0) = (A_1 - A_0)(x - x_0)$$

suponiendo $x = x_1$ tenemos

$$(A_1 - A_0)(x_1 - x_0) = A_1(x_1 - x_0) + A_1(x_1 - x_1) - A_0(x_1 - x_0)$$

$$(A_1 - A_0)s_1 = A_1(x_1 - x_0) - A_0(x_1 - x_0)$$

$$(A_1 - A_0)s_1 = y_1 - A_0(x_1 - x_0)$$

$$(A_1 - A_0)s_1 = y_1 - A_0 s_1$$

por lo tanto

$$A_1 = A_0 + \frac{(y_1 - A_0 s_1)s_1^T}{s_1^T s_1}$$

En general, para cualquier iteración, podemos calcular la pendiente del hiperplano de la secante utilizando la ecuación

$$A_{k+1} = A_k + \frac{(y_{k+1} - A_k s_{k+1})s_{k+1}^T}{s_{k+1}^T s_{k+1}} \tag{4.3}$$

4.4.1. Algoritmo

El algoritmo para este método es 15:

Algoritmo 15 Algoritmo de Broyden

BROYDEN($f(x), x_0$)

1 `Dado una función diferenciable` $f(x)$ `y un valor inicial` x_0
2 `Hacer` $k = 0$ `y` $A_0 = \nabla^2 f(x_0)$
3 **repetir**
4 `Resolver` $A_k s_{k+1} = -\nabla f(x_k)$ `para` s_{k+1}
5 $x_{k+1} = x_k + s_{k+1}$
6 $y_{k+1} = \nabla f(x_{k+1}) - \nabla f(x_k)$
7 $A_{k+1} = A_k + \frac{(y_{k+1} - A_k s_{k+1})s_{k+1}^T}{s_{k+1}^T s_{k+1}}$
8 $k = k + 1$
9 **hasta** $\|\nabla f(x_k)\| \leq \varepsilon$
10 **regresar** x_k

4.4.2. Ejemplo

Dado

$$\nabla f(x) = \begin{bmatrix} x_1 + x_2 & -3 \\ x_1^2 + x_2^2 & -9 \end{bmatrix}$$

la cual tiene raíces $(0,3)^T$ y $(3,0)^T$. Dado $x_0 = (1,5)^T$ aplicando el método de la secante calcular la solución y comparar con el método de Newton.

Primer iteración

$$A_0 = \nabla^2(x_0) = \begin{bmatrix} 1 & 1 \\ 2 & 10 \end{bmatrix}$$

el gradiente es

$$\nabla f(x_0) = \begin{bmatrix} 3 \\ 17 \end{bmatrix}$$

$$s_1 = -A_0^{-1}\nabla f(x_0) = \begin{bmatrix} -1.625 \\ -1.375 \end{bmatrix}$$

$$x_1 = x_0 + s_1 = \begin{bmatrix} -0.625 \\ 3.625 \end{bmatrix}$$

$$\nabla f(x_1) = \begin{bmatrix} 0 \\ 4.53125 \end{bmatrix}$$

La actualización de A es:

$$A_1 = A_0 + \begin{bmatrix} 0 & 0 \\ -1.625 & -1.375 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0.375 & 8.625 \end{bmatrix}$$

Segunda Iteración

$$s_2 = -A^{-1}\nabla f(x_1) = \begin{bmatrix} 0.549 \\ -0.549 \end{bmatrix}$$

$$x_2 = x_1 + s_2 = \begin{bmatrix} -0.076 \\ -3.076 \end{bmatrix}$$

$$\nabla f(x_1) = \begin{bmatrix} 0 \\ 0.466 \end{bmatrix}$$

$$A_2 = A_1 + \begin{bmatrix} 1 & 1 \\ -0.799 & 8.201 \end{bmatrix}$$

4.5. Método de Secante con actualización BFGS

Al resolver un problema de minimización sin restricciones por algún Quasi algoritmo de Newton, se tienen muchas mas ventajas y estabilidad numérica si se tiene un pseudo hessiano simétrico y definido positivo. En este caso utilizaremos la actualización del Hessiano propuesto por Broyden, Fletcher, Goldfarb and Shanno en 1970. El método se aplica de manera similar, pero el calculo de la matriz Hessiana se hace de la siguiente forma

$$A_{k+1} = A_k + \frac{y_{k+1}y_{k+1}^T}{y_{k+1}^T s_{k+1}} - \frac{A_k s_{k+1} s_{k+1}^T A_k}{s_{k+1}^T A_k s_{k+1}}$$

Algoritmo 16 Algoritmo BFGS

BFGS$(f(x), x_0)$

1 `Dado una función diferenciable` $f(x)$ `y un valor inicial` x_0
2 `Hacer` $k = 0$ `y` $A_0 = \nabla^2 f(x_0)$
3 **repetir**
4 `Resolver` $A_k s_{k+1} = -\nabla f(x_k)$ `para` s_{k+1}
5 $x_{k+1} = x_k + s_{k+1}$
6 $y_{k+1} = \nabla f(x_{k+1}) - \nabla f(x_k)$
7 $A_{k+1} = A_k + \frac{y_{k+1}y_{k+1}^T}{y_{k+1}^T s_{k+1}} - \frac{A_k s_{k+1} s_{k+1}^T A_k}{s_{k+1}^T A_k s_{k+1}}$
8 $k = k + 1$
9 **hasta** $\|\nabla f(x_k)\| \leq \varepsilon$
10 **regresar** x_k

4.6. Mínimos cuadrados no lineales

El problema de mínimos cuadrados puede ser escrito como

$$F(x) = \sum_{i=1}^{m} f_i^2(x)$$

$$F(x) = f^T(x) f(x)$$

y nuestro objetivo es minimizar la función $F(x)$ respecto de la variable x. Aplicando un esquema de Newton, debemos estimar el gradiente y Hessiano. Para obtener una expresión para el vector gradiente, las primeras derivadas parciales respecto a x_j esta dada por

$$g_j = 2 \sum_{i=1}^{m} f_i \frac{\partial f_i}{\partial x_j}$$

Si definimos la matriz Jacobiana J como

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

entonces podemos ver que el vector gradiente puede ser escrito como

$$g(x) = 2J^T(x) f(x)$$

Diferenciando nuevamente g_j, con respecto a x_k nos da el kj-ésimo elemento de la matriz hessiana

$$G_{kj} = 2 \sum_{i=1}^{m} \left\{ \frac{\partial f_i}{\partial x_k} \frac{\partial f_i}{\partial x_j} + f_i \frac{\partial^2 f_i}{\partial x_k \partial x_j} \right\}$$

entonces la matriz hessiana de $F(x)$ puede reescribirse como

$$G(x) = 2J^T(x) J(x) + 2 \sum_{i=1}^{m} f_i(x) T_i(x)$$

definiendo $S(x) = \sum_{i=1}^{m} f_i(x) T_i(x)$ tenemos

$$G(x) = 2J^T(x) J(x) + 2S(x)$$

La actualización de Newton queda como

$$(J^T(x_k) J(x_k) + S(x_k)) \delta_k = -J^T(x_k) f_k$$

$$x_{k+1} = x_k + \delta_k$$

4.6.1. Método de Gauss-Newton

Despreciando el termino S_k de la actualización de Newton tenemos la definición del método de Newton

$$(J^T(x_k)J(x_k))\delta_k = -J^T(x_k)f_k$$

$$x_{k+1} = x_k + \delta_k$$

Lo que resulta interesante de este método, es que el hecho de despreciar la parte de segundas derivadas del Hessiano, tenemos una mejora en la convergencia del algoritmo ya que $J^T(x_k)J(x_k)$ es una matriz definida positiva. Para mostrar esto podemos hacer $y = J(x_k)z$ podemos ver que

$$z^T J^T(x_k)J(x_k)z = y^T y \geq 0$$

Algoritmo 17 Algoritmo de Gauss Newton

GAUSS NEWTON$(f(x), x_0)$

1 `Dado una función diferenciable` $f(x)$ `y un valor inicial` x_0
2 `Hacer` $k = 0$
3 **repetir**
4 $\quad H(x_k) = J(x_k)^T J(x_k)$
5 $\quad$ `Resolver` $H(x_k)s_k = -\nabla f(x_k)$
6 $\quad x_{k+1} = x_k + s_k$
7 $\quad k = k + 1$
8 **hasta** $\|\nabla f(x_k)\| \leq \varepsilon$
9 **regresar** x_k

4.6.2. Método de Levenberg-Marquardt

El método de Levenberg-Marquardt incorpora una técnica para resolver el problema relacionado con la singularidad de la matriz $J^T(x_k)J(x_k)$ y es un algoritmo efectivo

para problema de residuos pequeños. La actualización en este caso esta dada por

$$(J^T(x_k)J(x_k) + \mu_k I)\delta_k = -J^T(x_k)f_k$$

Una estrategia para seleccionar μ_k, debe ser dada. El siguiente algoritmo proponer como realizar este procedimiento

Algoritmo 18 Algoritmo de Levenberg-Marquardt

LEVENBERG-MARQUARDT$(f(x), x_0)$

```
1  Dado una función diferenciable f(x) y un valor inicial x_0
2  Hacer μ_0 = 0.001, ν = 10
3  repetir
4          Poner μ_k = μ_k / ν
5          repetir
6                  Resolver (J^T(x_k)J(x_k) + μ_k I)δ_k = −J^T(x_k)f_k
7                  x_{k+1} = x_k + δ_k
8                  si F(x_{k+1}) > F(x_k)
9                     entonces μ_k = μ_k ν
10             hasta F(x_{k+1}) < F(x_k)
11         Hacer μ_{k+1} = μ_k
12         k = k + 1
13    hasta ‖∇f(x_k)‖ ≤ ε
14 regresar x_k
```

Capítulo 5

Algoritmos Genéticos

5.1. Búsqueda aleatoria

Este método evalúa en forma repetida la función mediante la selección aleatoria de valores de la variable independiente. Si un número suficiente de muestras se lleva a cabo, el óptimo será eventualmente localizado. (Para detalles ver [**?**])

Dada una función $f(x)$ en el dominio acotado $x_2 \in [x_2^{min}, x_2^{max}]$ y $x_2 \in [x_2^{min}, x_2^{max}]$, se genera un número aleatorio α entre 0 y 1 y los valores de x_1 y de x_2 en el rango se calculan como:

$$x_1 = x_1^{min} + (x_1^{max} - x_1^{min}) * \alpha$$

$$x_2 = x_2^{min} + (x_2^{max} - x_2^{min}) * \alpha$$

y en general para cualquier sistema de ecuaciones tenemos:

$$x_i = x_i^{min} + (x_i^{max} - x_i^{min}) * \alpha$$

Ejemplo

Encontrar el máximo de la función $f(x) = x_2 - x_1 - 2x_1^2 - 2x_1x_2 - x_2^2$ en el dominio acotado $x \in [-2, 2]$ y $y \in [1, 3]$.

k	x_1	x_2	$f(x,y)$
0	0.00000	0.00000	0.00000
1	-0.84276	1.19252	1.20270
2	-0.84276	1.19252	1.20270
3	-0.84276	1.19252	1.20270
4	-0.84276	1.19252	1.20270
5	-0.84276	1.19252	1.20270
6	-0.84276	1.19252	1.20270
7	-0.84276	1.19252	1.20270
8	-0.84276	1.19252	1.20270
9	-0.84276	1.19252	1.20270
167	-0.92622	1.28268	1.22395
183	-1.11212	1.49897	1.22463
465	-1.04210	1.66243	1.23375
524	-0.96660	1.36514	1.23859
626	-0.97539	1.48462	1.24931
999	-0.97539	1.48462	1.24931

La implementación de este algoritmo de búsqueda en Java es:

```
void Busqueda_Aleatoria()
{
  int i;
  double r, x, y, max = 0, maxx = 0, maxy = 0, fx;
  double xl = -2, xu = 2, yl = 1, yu = 3;
  Random azar = new Random();

  for(i=0; i<1000; i++)
  {
    r = azar.nextDouble();
    x = xl + (xu - xl)*r;
    r = azar.nextDouble();
    y = yl + (yu - yl)*r;
    fx = f(x,y);

    if(fx > max)
    {
      max = fx;
      maxx = x;
      maxy = y;
    }

    System.out.println(i + " " +maxx + " " + maxy + " " + max);
  }
}

double f(double x, double y)
{
  return y - x - 2*x*x - 2*x*y - y*y;
}
```

5.2. Problema del Agente Viajero

El problema del agente viajero o TSP como se le conoce principalmente en la literatura, consiste en un agente de ventas que tiene que visitas n ciudades, comenzando y terminando en una misma ciudad, visitando solamente una vez cada ciudad, excepto el origen y realizar el recorrido mínimo, este recorrido puede estar en función del tiempo o de la distancia, es decir, recorrer el mínimo de kilómetros o realizar el menor tiempo posible.

El problema del agente viajero se puede modelar fácilmente como un grafo completo dirigido, en donde los vértices del grafo son las ciudades y las aristas son los caminos, dichas aristas deben tener un peso, y este peso representa la distancia que hay entre dos vértices que están conectados por medio de dicha arista. Una solución del problema del agente viajero, se puede representar como una secuencia de n+1 ciudades, en donde se comienza y se termina con la misma ciudad: Solución = 0, 1, 2, 3, 0

La anterior es una de las posibles soluciones que hay para una instancia del problema con 4 ciudades, el cual se muestra en la figura 5.1

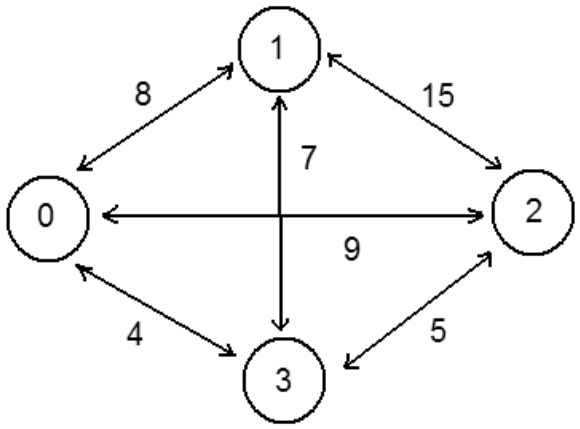


Figura 5.1: Problema del agente Viajero

5.2.1. Simulated Annealing

Las técnicas de recocido simulado "simulated annealing", tienen su origen en los procesos heurísticos que intentaban simular el comportamiento de un grupo de átomos

inicialmente en equilibrio a una temperatura t y que eran expuestos a enfriamiento. Un enfriamiento rápido llevaba a bloquear el sistema a un estado de alta energía que corresponde a un desorden, mientras que un enfriamiento lento o recocido, llevaba al sistema a un estado ordenado y con baja energía. Tal comportamiento se intenta reproducir por los programas de recocido, los cuales intentan simular el comportamiento que tiene un sistema de átomos en movimiento con energía E y temperatura t: en el sistema se elige un átomo de manera aleatoria y se le aplica un desplazamiento aleatorio. Considerando el cambio de energía dE, si dE fuera negativo entonces se acepta el desplazamiento y a la energía E se le resta la diferencia dE. Si, en cambio, dE fuera positivo, se evalúa, según la ley de Boltzmann, la probabilidad de que el cambio dE tenga lugar: $Prob(dE) = e^{-dE/kt}$, donde k es la constante de Boltzmann y t es la temperatura del sistema. La probabilidad $Prob(dE)$ se compara con el valor de una variable aleatoria x uniformemente distribuida en $[0, 1]$. El desplazamiento se acepta si y sólo si x no supera a $Prob(dE)$. Esta iteración básica se repite tantas veces como queda establecido por la ley de Boltzmann hasta que el sistema alcance un estado más estable a una temperatura t. En ese entonces t se multiplica por una constante de decaimiento en el intervalo [0,1] y la sucesión de iteraciones se repite. El valor minimal de la energía se decrementa con t así como la probabilidad de aceptar los desplazamientos. Cuando t es suficientemente pequeño, no se acepta más desplazamientos aleatorios y el sistema se congela en un estado casi estable que corresponde a un mínimo local de E. Esta heurística para problemas de optimización tiene las ventajas siguientes:

1. La heurística no se detiene al encontrar el primer óptimo local, sino que con cierta probabilidad positiva, procede a subsecuentes exploraciones del espacio de búsqueda.

2. La heurística exhibe a altas temperaturas un comportamiento similar a la técnica de divide y vencerás.

3. Bajo aseveraciones razonables, la heurística converge (quizás muy lentamente) a un óptimo global.

```
1. i = i0  //Valor inicial
2. T = T0  //Temperatura inicial
3. K = K0  //Máximo de Iteraciones.
4. while (condición de Paro)
5.  while (k < K && a < A)
```

```
6.      generar j en N(i)    //Vecindad de i
7.      if (f(j) - f(i) < 0)
8.          i = j
9.          a = a + 1
10.     else
11.         generar un numero r al azar (pseudo-random number)
12.         if (r < exp [(f(i) - f(j))/T])
13.             i = j
14.             a = a + 1
15.     k = k + 1
16. if (a = A)
17.     T =  alfa*T
18. else (k = K)
19.     T =  nu*T
20. K =  ro*K
21. k = 0
22. a = 0
23. mostrar i, c(i)
```

5.3. Algoritmos Genéticos

Los algoritmos genéticos, son algoritmos probabilistas en los cuales se introducen heurísticas basadas en la teoría de la evolución de las especies. Los pasos que se deben seguir para implementar un algoritmo genético son:

1. Definir parámetros, costo y una función de costo.
2. Crear aleatoriamente una población inicial.
3. Evaluar una función de costo.
4. Seleccionar parejas aleatoriamente
5. Reproducir
6. Mutar aleatoriamente algunos miembros de la población
7. Si la función de costo sigue decreciendo regresar a 3 sino terminar.
8. Parar

Las aplicaciones de los algoritmos genéticos son muy variadas, pero básicamente podemos definir los algoritmos genéticos con codificación binaria y codificación real.

5.4. Algoritmos Genéticos con codificación binaria

En este caso el algoritmo genético trabaja con un conjunto finito de valores. En este caso los algoritmos genéticos son ideales para minimizar funciones que solo pueden tomar un conjunto reducido de valores.

5.4.1. Representación de parámetros

Si el parámetro es continuo entonces este debe ser cuantizado. Las formular para hacer la codificación binaria y la decodificación de $n-esimo$ parámetro, p_n, son dados como

codificación: Para este paso se hace un escalamiento del intervalo $[p_{lo}, p_{hi}]$ al intervalo $[0, 1]$ utilizando

$$p_{norm} = \frac{p_n - p_{lo}}{p_{hi} - p_{lo}}$$

Posteriormente se hace una codificación binaria del número resultante. En java la codificación la hacemos como:

```
static public int[] codificar(double P, int k)
{
  double Pnorm = (P - Rangos[k][0])/(Rangos[k][1] - Rangos[k][0]);
  double suma;
  int gene[] = new int[Ngene];
  int i, aux;

  for(i=0; i<Ngene; i++)
  {
    Pnorm *= 2.0;
    aux = (int) Pnorm;
    gene[i] = aux;
    Pnorm -= aux;
  }
  return gene;
}
```

Decodificación: primeramente hacemos una conversión de binario a decimal y el valor del parámetro lo calculamos utilizando la ecuación

$$q_n = p_{quant}(p_{hi} - p_{lo}) + p_{lo}$$

La decodificación utilizando Java es

```
static public double decodificar(int gene[], int k)
{
  int i, n = gene.length;
  double suma = 0, fac = 0.5, valor;
```

```
  for(i=0; i<n; i++)
  {
    suma += gene[i]*fac;
    fac /=2.0;
  }

  valor = suma*(Rangos[k][1] - Rangos[k][0]) + Rangos[k][0];
  return valor;
}
```

Una vez realizada la codificación, un ejemplo de un cromosoma con $N_{gene} = 3$ y un número de parámetros $N_{par} = 4$ es:

001	111	101	010

5.4.2. Población inicial

El algoritmo genético comienza con una población inicial de tamaño N_{ipop}, la cual es una matriz de tamaño $N_{ipop} \times N_{bits}$ donde N_{bits} es el total de bits para representar los N_{par} de tamaño N_{gene}

```
static public void genera_poblacion()
{
  int i, j;
  int k = 0;

  for (i = 0; i < Npop; i++) {
    k = 0;
    for (j = 0; j < Npar; j++) {
      GeneraCromosoma(i, j);
    }
    Costos[i] = funcion(Cromosomas[i]);
  }
  ordena();
  Npop /= 2.0;
}

static public void GeneraCromosoma(int i, int j)
{
  int k;

  for(k=0; k<Ngene; k++)
    Cromosomas[i][j][k] = (int) (r.nextDouble() + 0.5);
}
```

Selección natural

Normalmente la población inicial es muy larga y es deseable reducirla eliminando algunos elementos de la población, para ello se aplica una rutina de ordenamiento que nos permitirá seleccionar a los mas aptos (una vez evaluada una función de costo). Así para una población inicial, reduciremos la población a la mitad de la población inicial $N_{pop} =$

$N_{ipop}/2$. La población mantendrá este tamaño durante todo el proceso iterativo. De esta población de tamaño N_{pop}, definiremos una parte que serán los mejores elementos N_{good} y los peores elementos N_{bad}. Los primeros, los N_{good} serán los candidatos a formar parejas y tener descendencia, los segundos serán reemplazados por la descendencia de los N_{good} elementos.

5.4.3. Ejemplo

Consideremos como ejemplo la función continua $f(x) = x_1 seno(4x_1) + 1.1x_2 seno(2x_2)$, para la cual deseamos calcular una población inicial en el rango $[0, 10]$ para ambas variables utilizando un cromosoma de 20 bits y una población inicial de 24 individuos. En la figura 5.2, se muestra la función utilizada en este ejemplo.

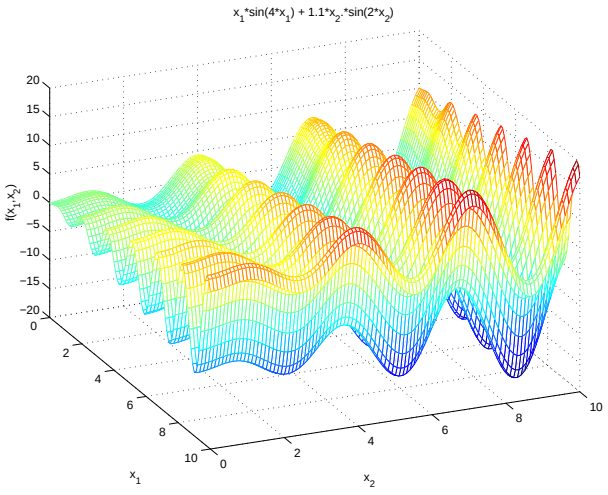


Figura 5.2: función $f(x) = x_1 seno(4x_1) + 1.1x_2 seno(2x_2)$

Ind	$Gene_1$	$Gene_2$	x_1	x_2	$f(x_1, x_2)$
0	1110001101	1000010010	8.876953125	5.17578125	-11.775107946435504
1	1110010101	0010111000	8.955078125	1.796875	-9.397234102008252
2	1001101100	1001100100	6.0546875	5.9765625	-8.57861714158948
3	1011110101	0011000111	7.392578125	1.943359375	-8.564664578124413
4	1001001000	0011000101	5.703125	1.923828125	-5.548160097898145
5	1110010010	1111010101	8.92578125	9.580078125	-4.910272831483166
6	0100101011	0111110110	2.919921875	4.90234375	-4.262645021922337
7	0111000010	0000000100	4.39453125	0.0390625	-4.195725821467517
8	0010101111	1000010111	1.708984375	5.224609375	-4.013162754305918
9	0100101000	0111101111	2.890625	4.833984375	-3.7188337466840142
10	0000000000	0011100111	0.0	2.255859375	-2.4316506543930565
11	1101011001	1110010100	8.369140625	8.9453125	-0.6696799445115991
12	1111001011	0111111011	9.482421875	4.951171875	-0.3361225990105128
13	1000001110	1001011110	5.13671875	5.91796875	0.7523936350592972
14	1101011001	1000111100	8.369140625	5.5859375	1.3357129488105146
15	1101100000	1000001110	8.4375	5.13671875	1.8566966237160756
16	0101000001	0101100110	3.134765625	3.49609375	2.4182539844896622
17	1111001010	0001000001	9.47265625	0.634765625	2.4698636738114845
18	0010111101	0101100000	1.845703125	3.4375	3.754030385201311
19	1010110011	0011010010	6.748046875	2.05078125	4.620931822460583
20	1010011101	0001001100	6.533203125	0.7421875	6.311111856904211
21	1010100011	1111001001	6.591796875	9.462890625	7.015104449242521
22	0000110000	1011001001	0.46875	6.962890625	7.935922596234852
23	1101100110	0110101110	8.49609375	4.19921875	8.55848738648271

5.4.4. Apareamiento

Existen diferentes formas de seleccionar a las parejas que tendrán descendencia. Es importante en el proceso de apareamiento que las parejas se selecciones de manera aleatoria. Algunas alternativas son

1. Apareamiento de arriba a abajo. En este esquema se hacen parejas con los $Ngood$ individuos de la siguiente forma $[1, 2]$, $[3, 4]$, $[5, 6]$, ... $[N_{good} - 1, N_{good}]$

2. Apareamiento aleatorio. En este caso se generan parejas aleatoria en el rango $[1, N_{good}]$

3. Apareamiento aleatorio pesado. En este caso se hace un aparemiento donde la probabilidad de apareamiento de cada uno de los individuos no es uniforme. Esta probabilidad dependerá del costo asociado a cada individuo, así el mas apto será el que tenga mayor

probabilidad de apareamiento. Esta se calcula utilizando la ecuación

$$p_i = \frac{c_i}{\sum_{i=1}^{Ngood} c_i}$$

donde

$$c_i = Costo_i - Costo_{Ngood+1}$$

El mejor esquema de apareamiento es el aleatorio pesado. Para nuestro ejemplo, las probabilidades pesadas son

i	$Costo_i$	$Costo_i - Costo_{N_{good}+1}$	p_i	$\pi_i = \sum p_i$
0	-11.7751	-7.5125	0.3238	0.3238
1	-9.3972	-5.1346	0.2213	0.5452
2	-8.5786	-4.3160	0.1860	0.7312
3	-8.5647	-4.3020	0.1854	0.9167
4	-5.5482	-1.2855	0.0554	0.9721
5	-4.9103	-0.6476	0.0279	1.0000

Para calcular las parejas se genera un número aleatorio α entre cero y uno. Se elige al $i - esimo$ individuo tal que $\alpha \leq \pi_i$.

Descendencia

Una vez calculadas las parejas, pasamos al proceso de generar descendencia. Para cada una de las parejas se selecciona un bit en el cual se llevará a cabo el proceso de cruzamiento. Esta posición se calcula de manera aleatoria.

Supongamos que aleatoriamente se generaron las parejas $[(3,2),(0,3),(2,0)]$ y un cruzamiento aleatorio en el bit 7, 6 y 10 respectivamente se tienen las siguientes parejas e hijos resultantes

	Ind	*Cromosoma*	x_1	x_2	$f(x_1, x_2)$
padre	3	$\overbrace{10111101}\overline{010011000111}$	7.392578125	1.943359375	-8.564664578124413
madre	2	$\overline{10011011}\overbrace{001001100100}$	6.0546875	5.9765625	-8.57861714158948
$hijo_1$	6	$\overbrace{10111101}\overbrace{001001100100}$	7.3828125	5.9765625	-10.805733576930972
$hijo_2$	7	$\overline{10011011010011000111}$	6.064453125	1.943359375	-6.103949823195019
padre	0	$\overbrace{1110001}\overline{1011000010010}$	8.876953125	5.17578125	-11.775107946435504
madre	3	$\overline{1011110}\overbrace{1010011000111}$	7.392578125	1.943359375	-8.564664578124413
$hijo_1$	8	$\overbrace{1110001}\overbrace{1010011000111}$	8.876953125	1.943359375	-8.67166082376904
$hijo_2$	9	$\overline{10111101011000010010}$	7.392578125	5.17578125	-11.668111700790877
padre	2	$\overbrace{1001101100}\overline{1001100100}$	6.0546875	5.9765625	-8.57861714158948
madre	0	$\overline{1110001101}\overbrace{1000010010}$	8.876953125	5.17578125	-11.775107946435504
$hijo_1$	10	$\overbrace{1001101100}\overbrace{1000010010}$	6.0546875	5.17578125	-9.347934290467709
$hijo_2$	11	$\overline{11100011011001100100}$	8.876953125	5.9765625	-11.005790797557275

5.4.5. Mutación

Este procedimiento, se realiza totalmente, de manera aleatoria. Normalmente el número de individuos a mutar va del 1 al 20 %, aunque este valor pudiera cambiar. Es deseable no mutar al mejor individuo de la población, por lo cual, se generan números aleatorios en el rango $[2, N_{good}]$ además se debe decir cual es el bit que se va a cambiar.

Vamos a suponer que se mutan el bit 18 del cromosoma 4 y el bit 3 del cromosoma 2 tenemos

Ind	*Cromosoma*	x_1	x_2	$f(x_1, x_2)$
0	11100011011000010010	8.876953125	5.17578125	-11.775107946435504
1	11100101010010111000	8.955078125	1.796875	-9.397234102008252
2	$100\overbrace{0}1011001001100100$	5.4296875	5.9765625	-2.3228094178933736
3	10111101010011000111	7.392578125	1.943359375	-8.564664578124413
4	$100100100000110001\overbrace{1}1$	5.703125	1.943359375	-5.6245656618150175
5	11100100101111010101	8.92578125	9.580078125	-4.910272831483166
6	10111101001001100100	7.3828125	5.9765625	-10.805733576930972
7	10011011010011000111	6.064453125	1.943359375	-6.103949823195019
8	11100011010011000111	8.876953125	1.943359375	-8.67166082376904
9	10111101011000010010	7.392578125	5.17578125	-11.668111700790877
10	10001011001000010010	5.4296875	5.17578125	-3.0921265667716025
11	11100011011001100100	8.876953125	5.9765625	-11.005790797557275

Este procedimiento se repite hasta convergencia, para el ejemplo de la función

$f(x) = x_1 seno(4x_1) + 1.1x_2 seno(2x_2)$, tenemos en la figura , el procedimiento de convergencia.

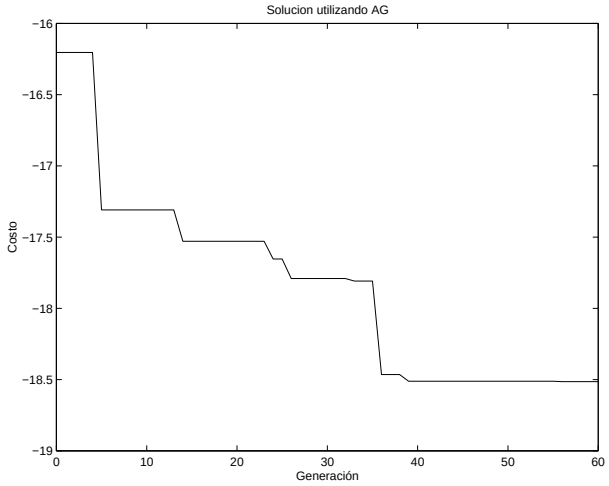


Figura 5.3: Solución usando AG para la función $f(x) = x_1 seno(4x_1) + 1.1x_2 seno(2x_2)$

5.5. AG con codificación Real

Dada una función $f : R^n$ la solución encontrada por los algoritmos Genéticos con codificación binaria, no darán una solución con la precisión necesaria ya que esta limitada al conjunto de valores que el proceso de cuantización puede ver. Esto se puede mejorar a medida que se aumenta el número de bits sin embargo el tiempo se ejecución se incrementa.

Los pasos que seguiremos serán los mismos, la diferencia radica en como determinar los valores.

5.5.1. Población Inicial

Para la población inicial generaremos aleatoriamente una población de tamaño N_{ipop} donde cada cromosoma tiene la forma

$$C = \boxed{c_1}\boxed{c_2}\boxed{\cdots}\boxed{c_j}\boxed{\cdots}\boxed{c_{N_{par}-1}}\boxed{c_{N_{par}}}$$

y los parámetros se calculan

$$c_j = (c_{hi} - c_{lo})\alpha + c_{lo}$$

donde α es un número aleatorio entre cero y uno. Note que calculados de esta manera se tendrán número aleatorios en el rango de cada uno de los parámetros que representan la función.

Selección Natural

Una vez generada la población inicial, se selecciona el 50 % de los individuos con menor costo. Esta será la población inicial y el tamaño se conservará durante el proceso de convergencia del algoritmo.

5.5.2. Apareamiento

Para el proceso de apareamiento consideraremos los mismos casos que en los AG con codificación binaria, esto debido a que para la selección de las parejas interviene solamente el costo y no la representación de los cromosomas.

5.5.3. Cruzamiento

El procedimiento de cruzamiento es llevado a cabo después de seleccionar aleatoriamente a las parejas.

Dado un par de padres p y m, el procedimiento de apareo, genera dos descendientes usando una pareja de cromosomas de acuerdo con

$$C^{(m)} = \boxed{c_1^{(m)} \mid c_2^{(m)} \mid \cdots \mid c_i^{(m)} \mid \cdots \mid c_{N_{par}}^{(m)}}$$

$$C^{(p)} = \boxed{c_1^{(p)} \mid c_2^{(p)} \mid \cdots \mid c_i^{(p)} \mid \cdots \mid c_{N_{par}}^{(p)}}$$

Entonces a punto de cruzamiento $j-th$ es seleccionado aleatoriamente en el intervalo $[1, N_{gen}]$ y los parámetros alrededor del gene j son intercambiados como muestra la expresión

$$offspring^{(1)} = \boxed{c_1^{(m)} \mid c_2^{(m)} \mid \cdots \mid c_j^{(new_1)} \mid \cdots \mid c_{N_{par}-1}^{(p)} \mid c_{N_{par}}^{(p)}}$$

$$offspring^{(2)} = \boxed{c_1^{(p)} \mid c_2^{(p)} \mid \cdots \mid c_j^{(new_2)} \mid \cdots \mid c_{N_{par}-1}^{(m)} \mid c_{N_{par}}^{(m)}}$$

El gene en la posición $j-th$ es combinado usando la siguiente ecuación

$$c_j^{(new_1)} = c_j^{(m)} - \beta(c_j^{(m)} - c_j^{(p)})$$

$$c_j^{(new_2)} = c_j^{(p)} + \beta(c_j^{(m)} - c_j^{(p)})$$

donde β es un número aleatoria entre cero y uno.

5.5.4. Mutación

Para este procedimiento es necesario seleccionar un porcentaje de mutaciones. Recuerde que la mutación es un procedimiento que permite búsquedas en lugares poco convencionales.

Para el proceso de mutación seleccionaremos aleatoriamente un número entre 2 y $Npop$, nunca seleccionaremos al individuo mejor adaptado para mutar. Posteriormente se selecciona aleatoriamente el parámetro del cromosoma que se alterara (entre 1 y $Npar$) y se calcula utilizando:

$$c_j = (c_{hi} - c_{lo})\alpha + c_{lo}$$

En la figura 5.4 podemos ver el desempeño del algoritmo genético para la función $f(x) = x_1 seno(4x_1) + 1.1x_2 seno(2x_2)$ planteada en el ejemplo. Note el mejor desempeño en este caso de la representación real (ver figura 5.3).

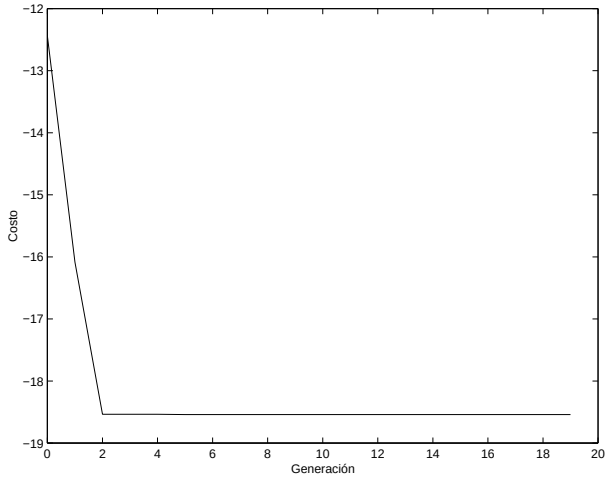


Figura 5.4: Solución usando AG para la función $f(x) = x_1 seno(4x_1) + 1.1x_2 seno(2x_2)$

Capítulo 6

Optimización con Restricciones

6.1. Teoría de la optimización con restricciones

La formulación general del problema, de optimización de una función con restricciones puede escribirse como:

$$\begin{aligned} &\text{mín} && f(x) && x \in \Re \\ &s.a \\ &&& C_i(x) = 0 : i \in \mathcal{E} \\ &&& C_i(x) \geq 0 : i \in \mathcal{I} \end{aligned} \tag{6.1}$$

donde $f(x)$ es la función objetivo, C_i son las restricciones de igualdad y desigualdad, i es el índice en el conjunto de números los cuales indican la cantidad de restricciones de igualdad $\mathcal{E}$ y desigualdad $\mathcal{I}$, y x es un vector de variables.

La región de factibilidad es aquella que contiene elementos (x) tales que cumplen con las restricciones C_i. En un lenguaje matemático se pueden expresar de la siguiente manera.

$$\Omega = \{x | C_i(x) = 0, i \in \mathcal{E}; C_i(x) \geq 0, i \in \mathcal{I}\}$$

de manera compacta podemos escribir 6.1 como:

$$\min_{x \in \Omega} f(x)$$

6.2. Una simple restricción de desigualdad

En los problemas de optimización sin restricciones, el problema de minimización es sencillo, solo deben encontrar los ceros del vector gradiente $\nabla f(x) = 0$ tal que la matriz hessina $H(x)$ seá definida positiva. El problema de minimización restringida es un poco más complejo, e iniciaremos por analizar dicho problema considerando restricciones de igualdad.

A manera de introducción consideraremos el siguiente ejemplo;

$$\begin{aligned} min\ f(x) &= x_1 + x_2 \\ s.a &\quad x_1^2 + x_2^2 - 2 = 0 \end{aligned}$$

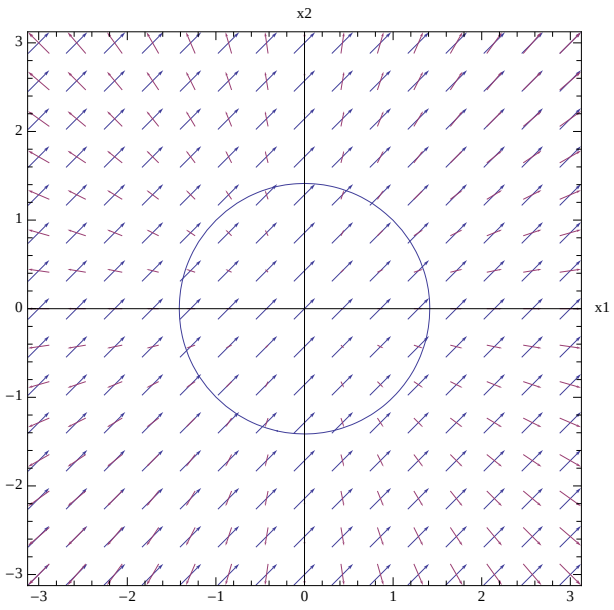


Figura 6.1: Función lineal con restricción cuadrática

Se puede observar que $f(x)$ es un plano recto, por lo tanto su mínimo será $x_{min} = \infty$. Sin embargo, al considerar la restricción, la cual es un circulo con centro en el origen y radio $\sqrt{2}$, el mínimo para este problema ya no se encuentra en $x_{min} = \infty$. Entonces,

la solución del problema estará acota a valores de x que pertenezcan a la única restrición $C_1 = x_1^2 + x_2^2 - 2$ (perímetro de la circunferencia). La solución obvia es x^* es $[-1, -1]$, ya que este valor es el único que cumple con C_1 y a la vez no existe otro valor sobre $f(x)$ que minimice $f(x)$ y satisfaga $C(x)$.

De la figura 6.1 se puede observar que existen un número infinito de valores de x tales que $C(x) = 0$, sin embargo estos no minimizan $f(x)$, por lo tanto no son mejores que $[-1, -1]$.

Ahora, observemos como se encuentran direccionados los gradientes de $f(x)$ y $c(x)$; el gradiente de la función es un vector $\nabla f(x) = [1, 1]^T$ y el gradiente de la única restricción es $\nabla C(x) = [2x_1, 2x_2]^T$

De la figura 6.1, se puede observar que x^* se encuentra en el punto en el que el gradiente de la función ∇f y el gradiente de las restricciones ∇C están alineados y pero con sentidos opuesta. Observe que en el óptimo $x^* = [-1, -1]^T$, la dirección del gradiente $\nabla f(x^*) = [1, 1]^T$ tiene la misma dirección que el vector gradiente de la restricciones $\nabla C(x^*) = [-2, -2]^T$, es decir:

$$\nabla f(x^*) = \lambda_1 \nabla C(x^*) \tag{6.2}$$

donde λ_1 es una constante de proporcionalidad. De esta ecuaciones podemos concluir que el gradiente de la función y el gradiente de las restricciones son vectores paralelos. En nuestro caso el valor de λ_1, puede ser calculado sustituyendo los valores del gradiente de la función y el gradiente de la restricción

$$\begin{aligned} \begin{bmatrix} 1 \\ 1 \end{bmatrix} &= \lambda \begin{bmatrix} -2 \\ -2 \end{bmatrix} \\ \lambda &= -\frac{1}{2} \end{aligned}$$

lo cual da como resultado $\lambda = -\frac{1}{2}$. Otro punto que cumple con dicha condición es $x = [1, 1]$, sin embargo es obvio que este punto no minimiza $f(x)$. Es necesario mostrar de manera formal este hecho

Otro ejemplo que podemos ver es el caso de la siguiente función cuadrática con una restricción lineal.

$$\begin{aligned} min\ f(x) &= 2x_1^2 + 3x_2^2 \\ s.a &\quad 3x_1 + x_2 - 3 = 0 \end{aligned}$$

El mínimo en este caso es $f(x^*) = 1.86207$ con $x^* = [0.931034, 0.206897]$, podemos notar que en el óptimo los gradientes de la función y la restricción tienen la misma dirección tal como se muestra en la figura 6.2.

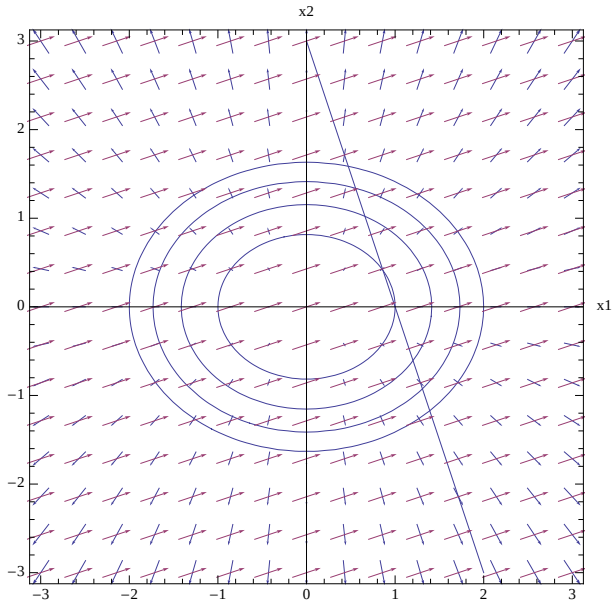


Figura 6.2: Función cuadrática con restricción lineal

El gradiente de la función valuado en el óptimo es

$$\nabla f(x^*) = \begin{bmatrix} 4x_1^* \\ 6x_2^* \end{bmatrix} = \begin{bmatrix} 4 \times 0.931034 \\ 6 \times 0.206897 \end{bmatrix}$$

El gradiente de la restricción queda como

$$\nabla C(x^*) = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

Con esto podemos notar que $\nabla f(x^*) = \lambda \nabla C(x^*)$ con $\lambda = 1.241382$

$$\begin{bmatrix} 3.724136 \\ 1.241382 \end{bmatrix} = 1.241382 \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

Un último ejemplo que podemos ver es el caso de de una función cuadrática con una restricción cuadratica dada por.

$$\begin{aligned} min\ f(x) &= 2x_1^2 + 3x_2^2 \\ s.a &\quad x_1^2 + x_2^2 = 3 \end{aligned}$$

Para este caso tenemos 2 mínimos localizados en $[\sqrt{3}, 0]$ y $[-\sqrt{3}, 0]$ con $f(x^*) = 6$, en la figura 6.3 podemos ver como en este punto los gradientes de la restricción y de la función son paralelos.

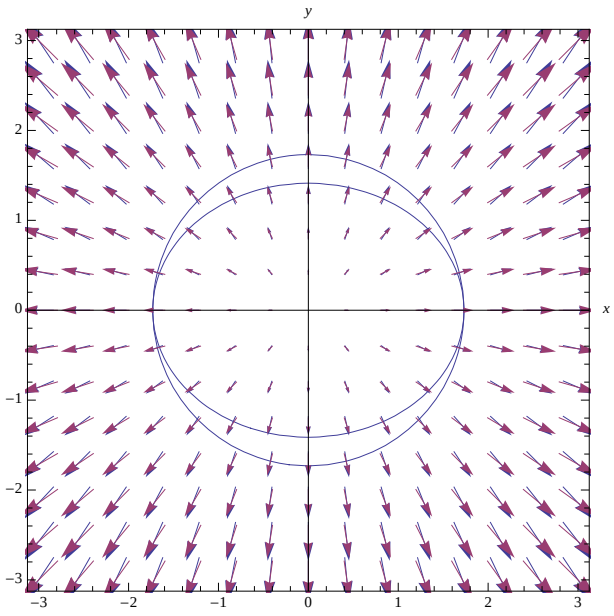


Figura 6.3: Función cuadrática con restricción cuadrática

El gradiente de la función valuado en el óptimo es

$$\nabla f(x^*) = \begin{bmatrix} 4x_1^* \\ 6x_2^* \end{bmatrix} = \begin{bmatrix} 4 \times 1.73205 \\ 6 \times 0 \end{bmatrix}$$

El gradiente de la restricción queda como

$$\nabla C(x^*) = \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} == \begin{bmatrix} 2 \times 1.73205 \\ 2 \times 0 \end{bmatrix}$$

Con esto podemos notar que $\nabla f(x^*) = \lambda \nabla C(x^*)$ con $\lambda = 1.241382$

$$\begin{bmatrix} 3.724136 \\ 1.241382 \end{bmatrix} = 1.241382 \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

6.3. Multiplicadores de Lagrange

El método de los multiplicadores de Lagrange es una herramienta útil para encontrar el mínimo o máximo, para una función $f(x)$ dada, sujeta a una a una restricción $C(x) = 0$. El método puede ser también usado en casos de mas de dos variables o mas de una restricción. Note que solamente para este método se aplican las restricciones de igualdad.

Restricción de Igualdad

La forma general del problema es

$$\begin{aligned} \text{mín} \quad & f(x) \\ s.a \quad & \\ & c_i(x) = 0 \end{aligned}$$

El método de multiplicadores de Lagrange se basa en una simple observación de que el gradiente de la restricción es paralelo al gradiente de la función, en el punto de solución eq. 6.1.

Prueba

Podemos derivar la expresión 6.2 aplicando la Serie de Taylor sobre la restricción de la función objetivo. Para mantener la restricción $C_1(x)$, si minimizamos en una dirección d, cualquiera se debe mantener que $c_1(x + d) = 0$ esto significa que

$$\begin{aligned} 0 &= c_1(x + d) \approx c_1(x) + \nabla c_1(x)^T d \\ 0 &= \nabla c_1(x)^T d \end{aligned} \tag{6.3}$$

Similarmente, una dirección apropiada que produzca un decrecimiento en $f(x)$, es

$$\begin{aligned} 0 &\geq f(x+d) - f(x) \\ &\approx f(x) + \nabla f(x)^T d - f(x) \end{aligned}$$

con lo cual tenemos

$$\nabla f(x)^T d < 0 \tag{6.4}$$

Existe un sinfín de números que cumplen con ambas restricciones (ecuaciones 6.3 y 6.4) y solamente existe una dirección que no satisface ambas restricciones. En el mínimo tenemos que $\nabla f(x)^T d = 0$ y debe de cumplirse que $\nabla c_1(x)^T d = 0$, por lo tanto en el mínimo el gradiente de la función debe ser paralelo al gradiente de la restricción

$$\nabla_x f(x) = \lambda \nabla_x C(x) \tag{6.5}$$

La ecuación 6.5 indica que si los ∇f y ∇C están alineados en un punto x, x es x^*, recuerde que esta condición es necesaria pero no suficiente, λ es denominado Multiplicador de Lagrange. La ecuación 6.5 da lugar a la llamada función Lagrangiana;

$$\mathcal{L} = f(x) - \lambda C(x) \tag{6.6}$$

de la ecuación 6.6 podemos ver que el gradiente de esta $\nabla\mathcal{L}$ debe ser igual a cero para cumplir con la ecuación 6.5.

6.3.1. Ejemplo

En este ejemplo se muestra una función de dos variables con una sencilla restricción de igualdad dada como, la cual se muestra en la figura 6.1:

$$\begin{aligned} \text{mín} \quad & f(X) = x_1 + x_2 \\ s.a. \quad & x_1^2 + x_2^2 - 2 = 0 \end{aligned}$$

La función Lagrangiana para este ejemplo es

$$\mathcal{L}(x, \lambda) = x_1 + x_2 + \lambda(x_1^2 + x_2^2 - 2)$$

y el gradiente de esta función es

$$\nabla\mathcal{L}(x,\lambda) = \begin{bmatrix} 1+2\lambda x_1 \\ 1+2\lambda x_2 \\ x_1^2+x_2^2-2 \end{bmatrix}$$

La solución X^* de acuerdo con 6.1 es $[-1,1]^T$ con $\lambda = -0.5$ y podemos verificar que para estos valores $\nabla\mathcal{L} = 0$

6.4. Restricciones de desigualdad

Consideremos una simple modificación al ejemplo anterior. En este caso queremos minimizar

$$x_1 + x_2$$

$$s.a$$

$$2 - x_1^2 - x_2^2 \geq 0$$

la región de factibilidad es el interior de un circulo de radio $\sqrt{2}$, de acuerdo como se muestra en la figura 6.1. Para determinar la solución tenemos que cumplir la restricción, cualquier movimiento en dirección d que hagamos debe cumplir también con esta restricción

$$0 \leq c_1(x+d) \approx c_1(x) + \nabla c_1(x)^T d$$

por lo que la factibilidad de primer orden se mantiene si

$$c_1(x) + \nabla c_1(x)^T d \geq 0 \tag{6.7}$$

En este caso también deben existir una dirección que satisfaga esta restricción y $\nabla f(x)^T d \leq 0$. Consideremos los siguientes casos

Caso 1: Considere primero el caso en el cual x vive dentro de la circunferencia, por lo cual la se da de manera estricta la restricción $c_1(x) > 0$. En este caso, algún vector d satisface la condición 6.7, siempre que la longitud sea pequeña. En particular, siempre que $\nabla f(x^*) \neq 0$, podemos obtener una dirección d que satisface ambas restricciones 6.7 y 6.4

haciendo

$$d = -c_1(x)\frac{\nabla f(x)}{\|\nabla f(x)\|}$$

Caso 2: Consideremos ahora el caso en el cual x vive en la frontera de el circulo, por lo cual $c_1(x) = 0$. Las condiciones entonces son

$$\begin{aligned} \nabla f(x)^T d &< 0 \\ \nabla c_1(x)^T d &\geq 0 \end{aligned}$$

las cuales son posibles cuando los vectores gradientes son paralelos, así que

$$\nabla f(x)^T d = \lambda_1 \nabla c_1(x)$$

para algún $\lambda_1 > 0$

Las condiciones de optimalidad para ambos casos pueden ser sumarizados con la función lagrangiana. Así tenemos

$$\nabla_x \mathcal{L}(x^*, \lambda_1^*) = 0$$

para algún $\lambda_1 > 0$ y también requeriremos que

$$\lambda_1^* c_1(x^*) = 0$$

Esta condición es conocida como la condición complementaria. Note que en el caso I, cuando $c_1(x^*) > 0$ requiere que $\lambda_1^* = 0$, de ahí que $\nabla f(x^*) = 0$. En el caso II tenemos que λ_1^* tenga un valor positivo con lo cual se cumple que $\nabla f(x)^T d = \lambda_1 \nabla c_1(x)$.

6.5. Condiciones Necesarias de Primer Orden

Suponga que x^* es una solución de una función con restricciones. Entonces existe un multiplicador de Lagrange λ^*, con componentes $\lambda_i^*, i \in \mathcal{E} \bigcup \mathcal{I}$, tal que las siguientes

condiciones son satisfechas en el punto (x^*, λ^*)

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda_1^*) &= 0 \\ c_i(x^*) &= 0 \text{ para todo } i \in \mathcal{E} \\ c_i(x^*) &\geq 0 \text{ para todo } i \in \mathcal{I} \\ \lambda_i^* &\geq 0 \text{ para todo } i \in \mathcal{E} \\ \lambda_i^* c_i(x^*) &= 0 \text{para todo } i \in \mathcal{E} \bigcup \mathcal{I} \end{aligned}$$

Estas son conocidas como las condiciones de Karush-Kuhn-Tucker o KKT.

6.6. Programación Cuadrática

A los problemas de optimización, cuya función objetivo es cuadrática y sus restricciones son lineales, se le denominan problema de *Programación Cuadrático (QP)*. La formulación general del problema es el siguiente;

$$\text{mín } q(x) = \tfrac{1}{2} x^T G x + x^T d \tag{6.8}$$

$s.a$

$$a_i^T x = b_i,\ i \in \varepsilon \tag{6.9}$$

$$a_i^T x \geq b_i,\ i \in \mathrm{I} \tag{6.10}$$

donde G es una matriz simétrica de tamaño $n \times n$, y $\mathcal{E}$ e $\mathcal{I}$ son conjuntos de indices correspondientes a las restricciones de igualdad y desigualdad, respectivamente.

Los problemas de QP pueden ser resueltos, o puede mostrarse, que son infactibles, en un número finitos de iteraciones, el esfuerzo de solución depende de las características de la función objetivo y del número de restricciones de desigualdad.

Si G es positiva entonces el problema es denominado QP convexa

6.6.1. Restricciones de igualdad QP

Consideramos que el número de restricciones de igualdad es m, y que $m \geq 0$, escribimos el problema de QP como;

$$\begin{aligned} \text{mín} \quad & q(x) \stackrel{def}{=} \frac{1}{2}x^T Gx + x^T d \\ s.a. & \\ & Ax = b \end{aligned} \tag{6.11}$$

donde A es de $m \times n$ y es el Jacobiano de las restricciones de igualdad, definido por:

$$A = [a_i] \qquad \forall i \in \mathcal{E} \tag{6.12}$$

Consideraremos, que A es de rango completo (rango = m) y que las restricciones son consistentes. La solución de 6.11 requiere del cumplimiento de las condiciones de KKT, las cuales son:

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*) = 0 \\ C_i(x) = 0 \end{aligned} \tag{6.13}$$

Para expresar completamente la ecuación 6.11 en términos de KKT, formulamos la $\mathcal{L}$ del problema;

$$\mathcal{L} = \frac{1}{2}x^T Gx + x^T d - \lambda^T (Ax - b) \tag{6.14}$$

donde el $\nabla\mathcal{L}$ es;

$$\nabla_x : \mathcal{L}(x^*, \lambda^*) = Gx^* + d - A^T\lambda^* = 0 \tag{6.15}$$

Además, las restricciones de igualdad es la ecuación **??** en términos de KKT, son:

$$Ax^* - b = 0 \tag{6.16}$$

Por medio de 6.15 y 6.16 hemos expresado las condiciones de KKT y en x^* estas deben ser satisfecha simultáneamente, de tal manera que las podemos expresar como sigue.

$$\begin{bmatrix} G & -A^T \\ -A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} -d \\ -b \end{bmatrix} \tag{6.17}$$

Entonces el problema ha sido reducido a la solución de 6.17. La ecuación 6.17 pude reescribirla, para que esta sea usada para cuestiones computacionales. Se puede expresar x^* como;

$$x^* = x + p \tag{6.18}$$

donde x es un valor estimado de la solución y p es el incremento deseado. Sustituyendo 6.18 en 6.15 6.16 obtenemos:

$$\begin{bmatrix} G & -A^T \\ -A & 0 \end{bmatrix} \begin{bmatrix} p \\ \lambda^* \end{bmatrix} = \begin{bmatrix} -Gx - d \\ -b + Ax \end{bmatrix} \Rightarrow \begin{bmatrix} G & -A^T \\ -A & 0 \end{bmatrix} \begin{bmatrix} p \\ \lambda^* \end{bmatrix} = \begin{bmatrix} g \\ c \end{bmatrix} \tag{6.19}$$

Entonces el problema se ha reducido a solucionar la ecuación 6.11, 6.17 o 6.19 y la matriz $\begin{vmatrix} G & -A^T \\ -A & 0 \end{vmatrix}$ es denominada Matriz de Karush - Kum - Tucker.

6.7. Conjunto Activo

El problema a minimizar es el siguiente;

$$\begin{aligned} \text{mín } q(x) &= \frac{1}{2}x^T Gx + x^T d \\ s.a. &: a_i^T x = b \end{aligned}$$

El método comienza por hacer una estimación x_0 factible, y después se aseguran de que todas las subsecuentes iteraciones sigan siendo factibles. Se encuentra un paso a partir de la primera iteración resolviendo un sub-problema cuadrático en el cual un subconjunto 6.9, 6.10 se impongan como igualdades. Este subconjunto es nuestro *conjunto de trabajo* y es denotado como W_k en la iteración k. Este consiste en todas las igualdades existentes en $\mathcal{E}$ 6.9 junto con algunos pero no necesariamente todas las desigualdades.

Dada un x_k en una iteración y un conjunto de trabajo W_k, primero observamos si el x_k minimiza a q en el subespacio definido por el conjunto de trabajo. Si no, calculamos un p resolviendo un problema QP. Con lo cual definimos lo siguiente; $p = x - x_k$, y $g_k = Gx_k + d$, y sustituyendo x en 6.8 encontramos que;

$$q(x) = q(x_k + p) = \frac{1}{2}p^T Gp + g_k^T p + c,$$

Donde $c = \frac{1}{2}x_k^T G x_k^T + d^T x_k$ con esto nosotros podemos escribir el problema QP a resolver como;

$$\text{mín } \frac{1}{2}p^T Gp + g_k^T p \tag{6.20}$$

$$s.a: \qquad a_i^T p = 0, : \forall : i \in W_k \tag{6.21}$$

Suponga que tenemos un p_k que es diferente de cero. Necesitamos decidir como movernos a lo largo de esta dirección. Si $x_k + p_k$ es factible con respecto a todas las restricciones, nuestro conjunto $x_{k+1} = x_k + p_k$. Sino ocurre esto fijamos.

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

Donde el valor del parámetro α es elegido en un rango entre $[0, 1]$ para el cual se cumplen todas las restricciones. Siempre que $a_i^T p_k < 0$ para algún $i \notin W_k$, sin embargo, nosotros tendremos que $a_i^T(x_k + \alpha_k p_k) \geq b_i$ solo si

$$a_i^T x_k + a_i \alpha_k p_k \geq b_i$$
$$\alpha_k a_i^T p_k \geq b_i - a_i^T x_k$$

$$\alpha_k \leq \frac{b_i - a_i^T x_k}{a_i^T p_k}$$

Puesto que quisiéramos que el α_k fuera tan grande como sea posible dentro del rango $[0, 1]$ conforme la factibilidad de retención, tenemos la definición siguiente:

$$\alpha_k \overset{def}{=} \text{mín}\left(1, \min_{W_k, a_i^T p_k < 0} \frac{b_i - a_i^T x_k}{a_i^T p_k}\right) \tag{6.22}$$

Activamos aquellas restricciones para las cuales el mínimo 6.22 se cumplió (Si $\alpha_k = 1$ y no hay nuevas restricciones activas en el $x_k + \alpha_k p_k$ entonces no hay restricciones de bloqueo en esta iteración)

Si $\alpha_k < 1$, esto es, el paso a lo largo de p_k fue bloqueado por alguna restricción que no esta en W_k, donde un nuevo conjunto de trabajo W_{k+1} es construido agregando una de las restricciones de bloqueo a W_k.

Ahora examinamos las muestras de los multiplicadores que corresponden a las restricciones en el conjunto de trabajo, es decir, los índices $i \in W$. Si estos multiplicadores no son negativos, la cuarta condición de KKT ($\lambda_i^* \geq 0$, para toda $i \in \mathcal{I} \cap A(x^*)$) también está se satisface, Concluimos que x' es un punto de KKT para el problema original. Puesto que G es definida semi-positiva, podemos demostrar que el x' es un minimizar. Cuando G es definida positiva, x' se minimiza.

Si por otra parte unos de los multiplicadores λ_j, $j \in W \cap \mathcal{I}$, son negativos, la condición cuatro de KKT no se satisface y la función objetivo $q(\cdot)$ puede ser decrementada por la disminución de las restricciones. Después quitamos un índice j que corresponde a uno de los multiplicadores negativos del conjunto de trabajo y solucionamos un nuevo problema para la siguiente iteración.

6.7.1. Algoritmo Conjunto Activo

1 Calcular un punto factible inicial x_0;

2 Poner W_0 como el sub-conjunto activo de las restricciones en el x_0

3 for k=0,1,2,...

4 Resolver mín $\frac{1}{2}p_k^T G p_k + g_k^T p_k$ para encontrar p_k

5 Si $p_k = 0$ {

6 Calcular los Multiplicadores de Lagrange $\hat{\lambda}_i$ que satisfacen $\sum_{i=1} a_i \lambda_i = g = Gx + d$ donde el conjunto $\hat{W} = W_k$

7 Si $\hat{\lambda} \geq 0$ para toda $i \in$ W $\cap$ I;

8 Parar, con la solución $x^* = x_k$

9 sino

10 Asignar a $j =$ arg $min_{j \in W_k \cap \mathrm{I}} \hat{\lambda}_j$;

11 $x_{k+1} = x_k; W_{k+1} \longleftarrow W_k \backslash \{j\}$

}

12 sino ($p_k \neq 0$) {

13 Calculamos α_k para mín $\left(1, \underset{i \notin W_k, a_i^T p_k < 0}{\text{mín}} \frac{b_i - a_i^T x_k}{a_i^T p_k}\right)$

14 $x_{k+1} \longleftarrow x_k + \alpha_k p_k$

15 Si las restricciones están bloqueas

16 Se obteniendo W_{k+1} agregando una de las restricciones bloqueadas a W_{k+1}

18 sino

20 $W_{k+1} \longleftarrow W_k$;

}

21 fin (for)

6.7.2. Ejemplos

Utilizando el método del conjunto activo de la programación cuadrática, encuentre los valores de las incógnitas que minimizan la función objetivo dada para cada caso, considerando para ello, las ecuaciones de restricción a las cuales se sujetan dichas funciones.

ejemplo uno

Minimizar:

$$q(x) = (x_1 - 2)^2 + (x_2 - 2)^2$$

sujeto a:

$$\begin{aligned} x_1 + x_2 &\geq 4 \\ 2x_1 - x_2 &\geq 0 \end{aligned}$$

considere la aproximación inicial al vector solución: $x^{(0)} = [2, 4]^T$

Los resultados obtenidos del algoritmo del conjunto activo para el problema planteado se muestran en la tabla 6.1. Además, en la figura 6.4 se muestra gráficamente el proceso de solución.

6.7.3. ejemplo dos

Minimizar:

$$q(x) = (x_1 - 2)^2 + (x_2 - 2)^2$$

k	x		p		λ		Notas
0	2.0	4.0	-0.8	-1.6	0.0	-0.8	Activar λ_0
$\alpha = 0.8333$ $x^{(k+1)} = [1.3333, 2.6667]^T$							
1	1.3333	2.6667	0.0	0.0	0.4444	-0.8889	Sacar λ_1
2	1.3333	2.6667	0.6667	-0.6667	0.0	0.0	
$\alpha = 1.0$ $x^{(k+1)} = [2.0, 2.0]^T$							
3	2.0	2.0	0.0	0.0	0.0	0.0	
$F(x)_{min} = 0.0$ $x^{(min)} = [2.0, 2.0]^T$							

Tabla 6.1: Resultados del ***Método del Conjunto Activo*** para $q(x) = (x_1 - 2)^2 + (x_2 - 2)^2$; $x_0 = [2, 4]^T$.

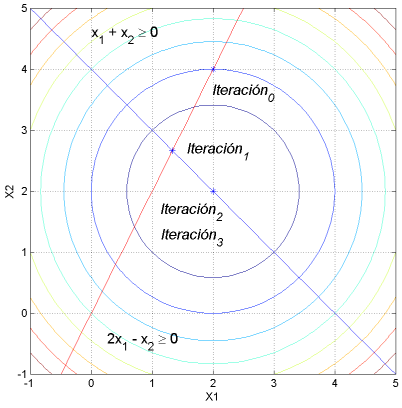


Figura 6.4: Solución del ***Método del Conjunto Activo*** para $q(x) = (x_1 - 2)^2 + (x_2 - 2)^2$; $x^{(0)} = [2, 4]^T$.

k	x		p		λ		Notas
0	4.0	3.0	-2.0	-1.0	0.0	0.0	Activar λ_0
$\alpha = 0.3333$ $x^{(k+1)} = [3.3333, 2.6667]^T$							
1	3.3333	2.6667	-0.3333	0.3333	2.0	0.0	
$\alpha = 1.0$ $x^{(k+1(} = [3.0, 3.0]^T$							
2	3.0	3.0	0.0	0.0	2.0	0.0	
$F(x^*) = 2.0$ $x^* = [3.0, 3.0]^T$							

Tabla 6.2: Resultados del ***Método del Conjunto Activo*** para $q(x) = (x_1 - 2)^2 + (x_2 - 2)^2$; $x^{(0)} = [4, 3]^T$.

sujeto a:

$$\begin{aligned} x_1 + x_2 &\geq 6 \\ 2x_1 - x_2 &\geq 0 \end{aligned}$$

considere la aproximación inicial al vector solución: $x^{(0)} = [4, 3]^T$

Los resultados obtenidos del algoritmo del conjunto activo para el problema planteado se muestran en la tabla 6.2. Además, en la figura 6.5 se muestra gráficamente el proceso de solución.

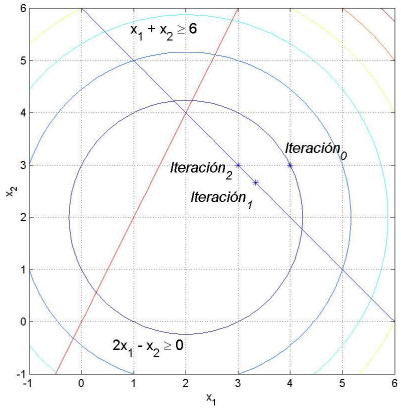


Figura 6.5: Solución del ***Método del Conjunto Activo*** para $q(X) = (x_1 - 2)^2 + (x_2 - 2)^2$; $x^{(0)} = [4, 3]^T$.

6.7.4. Ejercicio (del libro de Jorge Nocedal)

Minimizar la función:

$$q(x) = (x_1 - 1)^2 + (x_2 - 2.5)^2$$

sujeto a:

$$\begin{aligned} x_1 - 2x_2 + 2 &\geq 0 \\ -x_1 - 2x_2 + 6 &\geq 0 \\ -x_1 + 2x_2 + 2 &\geq 0 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

considere la aproximación inicial al vector solución: $x_0 = [1, 1]^T$

En este caso tenemos que el Hessiano G y el vector d es

$$G = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$d = \begin{bmatrix} -2 \\ -5 \end{bmatrix}$$

Las restricciones puden expresarse como

$$Ax \geq b$$

con

$$\begin{bmatrix} 1 & -2 \\ -1 & -2 \\ -1 & 2 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq \begin{bmatrix} -2 \\ -6 \\ -2 \\ 0 \\ 0 \end{bmatrix}$$

Iteración 1

Dado el punto inicial $x^{(0)} = [1, 1]^T$, el conjunto activo es $W^{(0)} = \{\}$ y tenemos que solucionar el sistema

$$\begin{bmatrix} 2.0 & 0.0 \\ 0.0 & 2.0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \begin{bmatrix} 0 \\ -3 \end{bmatrix}$$

La solucion es $p^{(0)} = [0.0, 1.5]^T$ y el cáculo de $Ap^{(0)} = [-3.0, -3.0, 3.0, 0.0, 1.5]^T$ por lo que las restricciones 1 y 2 se se activan. Para ello calculamos los valores de $\alpha^{(0)} = [0.33333, 1.0, *, *, *]^T$ y el valor ḿinimo es

$$\alpha^{(0)}_{min} = min\{1, [0.3333, 1.0, *, *, *]^T\} = 0.3333$$

En cálculo de las variables en la siguiente iteración queda como

$$x^{(1)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 0.33333 \times \begin{bmatrix} 0 \\ 1.5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1.5 \end{bmatrix}$$

El conjunto activo de restricciones queda $W^{(1)} = \{1\}$, dado que $x^{(1)}$ esta sobre la restricción 1.

Iteración 2

Dado el punto inicial $X^{(1)} = [1, .5]^T$, el conjunto activo es $W^{(1)} = \{1\}$ y tenemos que solucionar el sistema

$$\begin{bmatrix} 2.0 & 0.0 & -1.0 \\ 0.0 & 2.0 & 2 \\ -1.0 & 2.0 & 2 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \lambda_1 \end{bmatrix} \begin{bmatrix} 0 \\ -2 \\ 0 \end{bmatrix}$$

La solucion es $[p^{(2)}, \lambda_1]^T = [0.4, 0.2, 0.8]^T$ y el cáculo de $Ap^{(2)} = [0.0, -0.8, 0.0, 0.4, 0.2]^T$ por lo que la restriccion 2 se violan. Para ello calculamos los valores de $\alpha^{(2)} = [0, 2.5, *, *, *]^T$ y el valor ḿinimo es:

$$\alpha^{(2)}_{min} = min\{1, [0, 2.5, *, *, *]^T\} = 1.0$$

En cálculo de las variables en la siguiente iteración queda como:

$$x^{(1)} = \begin{bmatrix} 1 \\ 0.4 \end{bmatrix} + 1 \times \begin{bmatrix} 1.5 \\ 0.2 \end{bmatrix} = \begin{bmatrix} 1.4 \\ 1.7 \end{bmatrix}$$

El conjunto activo de restricciones queda $W^{(2)} = \{1\}$ dado que la restricción 2 no se viola.

Iteración 3

Dado el punto inicial $X^{(2)} = [1.4, 1.7]^T$, el conjunto activo es $W^{(2)} = \{1\}$ y tenemos que solucionar el sistema

$$\begin{bmatrix} 2.0 & 0.0 & -1.0 \\ 0.0 & 2.0 & 2 \\ -1.0 & 2.0 & 2 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \lambda_1 \end{bmatrix} \begin{bmatrix} 0.8 \\ -1.6 \\ 0 \end{bmatrix}$$

La solucion es $[p^{(3)}, \lambda_1]^T = [0.0, 0.0, 0.8]^T$ y el cáculo de $Ap^{(1)} = [0.0, 0.0, 0.0, 0.0, 0.0]^T$ por lo que no se viola restriccion alguna, por lo tanto la solución es:

$$x^{(3)} = \begin{bmatrix} 1.4 \\ 1.7 \end{bmatrix} + 1 \times \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 1.4 \\ 1.7 \end{bmatrix}$$

con esto se da por terminado el procedimiento siendo la solución $x^{(*)} = [1.4, 1.7]^T$.

6.7.5. Ejercicio (del libro de Jorge Nocedal)

Minimizar la función:

$$q(X) \quad = \quad (x_1 - 1)^2 + (x_2 - 2.5)^2$$

sujeto a:

$$\begin{aligned} x_1 - 2x_2 + 2 &\geq 0 \\ -x_1 - 2x_2 + 6 &\geq 0 \\ -x_1 + 2x_2 + 2 &\geq 0 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

considere la aproximación inicial al vector solución: $x_0 = [2, 0]^T$

Iteración 1

Dado el punto inicial $x^{(0)} = [2, 0]^T$, el conjunto activo es $W^{(0)} = \{3, 5\}$ en virtud de que el punto $x^{(0)}$ está en las intesección de ambas desigualdades. El sistema de ecuaciones a resolver es:

$$\begin{bmatrix} 2.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 2.0 & -2.0 & -1.0 \\ 1.0 & -2.0 & 0.0 & 0.0 \\ 0.0 & -1.0 & 0.0 & 0.0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \lambda_3 \\ \lambda_5 \end{bmatrix} = \begin{bmatrix} 2.0 \\ -5.0 \\ 0.0 \\ 0.0 \end{bmatrix}$$

La solucion es $[p^{(0)}, \lambda_3, \lambda_5] = [0.0, 0.0, -2.0, -1.0]^T$. En este caso tenemos valores de p en cero pero las *lambdas*'s están violando la condición de ser mayores de cero, esto se debe a que no estamos en el mínimo de la función. Por tal motivo tenemos que eliminar del conjunto activo la restricción con el valor de lambda menor.

$$\lambda_{min} = min[-2, -1] = -2$$

por lo tanto el conjunto activo queda $W^{(1)} = \{3, 5\} - \{3\} = \{5\}$ y $x^{(1)} = x^{(0)} = [2, 0]$

Iteración 2

Dado el punto $x^{(1)} = [2, .0]^T$ y el conjunto activo $W^{(1)} = \{5\}$, tenemos que solucionar el sistema de ecuaciones

$$\begin{bmatrix} 2.0 & 0.0 & 0.0 \\ 0.0 & 2.0 & -1.0 \\ 0.0 & -1.0 & 0.0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \lambda_5 \end{bmatrix} = \begin{bmatrix} 2.0 \\ -5.0 \\ 0.0 \end{bmatrix}$$

La solucion es $[p^{(2)}, \lambda_5]^T = [-1.0, 0.0, -5.0]^T$ y el cáculo de $Ap^{(2)} = [-1.0, 1.0, 1.0, -1.0, 0.0]^T$ por lo que la restriccion 1 y 4 se violan. Para ello calculamos los valores de $\alpha^{(2)} = [4, *, *, 2, *]$ y el valor ḿinimo es:

$$\alpha^{(2)}_{min} = min\{1, [4, *, *, 2, *]^T\} = 1.0$$

En cálculo de las variables en la siguiente iteración queda como:

$$x^{(2)} = \begin{bmatrix} 2.0 \\ 0.0 \end{bmatrix} + 1 \times \begin{bmatrix} -1.0 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix}$$

El conjunto activo de restricciones queda $W^{(2)} = \{\}$ dado los valores de $x^{(2)}$ no violan las restricciones y que λ_5 es negativo.

Iteración 3

Dado el punto $x^{(2)} = [1.0, 0.0]^T$ y el conjunto activo es $W^{(2)} = \{\}$, tenemos que solucionar el sistema de ecuaciones dado como:

$$\begin{bmatrix} 2.0 & 0.0 \\ 0.0 & 2.0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} 0.0 \\ -5.0 \end{bmatrix}$$

La solucion es $[p^{(3)}]^T = [0.0, 2.5]^T$ y el cáculo de $Ap^{(3)} = [-5.0, -5.0, 5.0, 0.0, 2.5]^T$ por lo que la restriccion 1 y 2 se violan. Para ello calculamos los valores de $\alpha^{(3)} = [0.6, 1.0, *, *, *]$ y el valor ḿinimo es:

$$\alpha^{(3)}_{min} = min\{1, [0.6, 1.0, *, *, *]\} = 0.6$$

En cálculo de las variables en la siguiente iteración queda como:

$$x^{(3)} = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} + 0.6 \times \begin{bmatrix} 0.0 \\ 2.5 \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.5 \end{bmatrix}$$

El conjunto es $W^{(3)} = \{1\}$ dado que los valores de $x^{(3)}$ están sobre la restricción 1.

Iteración 4

Dado el punto $x^{(3)} = [1, 1.5]^T$, el conjunto activo es $W^{3)} = \{1\}$ y tenemos que solucionar el sistema

$$\begin{bmatrix} 2.0 & 0.0 & -1.0 \\ 0.0 & 2.0 & 2.0 \\ -1.0 & 2.0 & 0.0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} 0.0 \\ -2.0 \\ 0 \end{bmatrix}$$

La solucion es $[p^{(4)}]^T = [0.4, 0.2, 0.8]^T$ y el cáculo de $Ap^{(2)} = [0.0, -0.8, 0.0, 0.4, 0.2]^T$ por lo que la restricción 2 es violada. Para ello calculamos los valores de $\alpha^{(3)} = [*, 2.5, *, *, *]$ y el valor ḿinimo es:

$$\alpha^{(3)}_{min} = min\{1, [*, 2.5, *, *, *]\} = 1.0$$

En cálculo de las variables en la siguiente iteración queda como:

$$x^{(3)} = \begin{bmatrix} 1.0 \\ 1.5 \end{bmatrix} + 1.0 \times \begin{bmatrix} 0.4 \\ 2.2 \end{bmatrix} = \begin{bmatrix} 1.4 \\ 1.7 \end{bmatrix}$$

y el conjunto activo de restricciones queda $W^{(3)} = \{1\}$ dado no hubo cambios en los valores α

Iteración 5

Dado el punto $X^{(4)} = [1.4, 1.7]^T$, el conjunto activo es $W^{4)} = \{1\}$ y tenemos que solucionar el sistema

$$\begin{bmatrix} 2.0 & 0.0 & -1.0 \\ 0.0 & 2.0 & 2.0 \\ -1.0 & 2.0 & 0.0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} 0.8 \\ 1.6 \\ 0.0 \end{bmatrix}$$

La solucion es $[p^{(5)}]^T = [0.0, 0.0, 0.8]^T$ y el cáculo de $Ap^{(5)} = [0.0, 0.0, 0.0, 0.0, 0.0]^T$ por lo que no se viola ninguna restricción. El valor ḿinimo es por lo tanto:

$$\alpha_{min}^{(5)} = min\{1, [*, *, *, *, *]\} = 1.0$$

En cálculo de las variables en la siguiente iteración queda como:

$$x^{(5)} = \begin{bmatrix} 1.4 \\ 1.7 \end{bmatrix} + 1.0 \times \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 1.4 \\ 1.7 \end{bmatrix}$$

El conjunto activo de restricciones queda $W^{(3)} = \{1\}$ dado que $x^{(5)}$ esta sobre el borde de la restricción 1.

El resultado final se obtine en 5 iteraciones y los resultados obtenidos del algoritmo del conjunto activo para el problema planteado se muestran en la tabla 6.3 y en la figura 6.6 se muestra gráficamente el proceso de solución.

6.8. El método de Barrera Logaritmica

Comenzaremos por describir el concepto de función de barrera en terminos de un problema con restricciones de desigualdad. Dado el problema

$$\min_x f(x) \ s.a \ c_i(x) \geq 0 \ i \in \mathcal{I}$$

k	x		p		λ					Notas
0	2.0	0.0	0.0	0.0	0.0	0.0	-2.0	0.0	-1.0	Sacar λ_3
1	2.0	0.0	-1.0	0.0	0.0	0.0	0.0	0.0	-5.0	
$\alpha = 1.0$ $X_{k+1} = [1.0, 0.0]^T$										
2	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-5.0	Sacar λ_5
3	1.0	0.0	0.0	2.5	0.0	0.0	0.0	0.0	0.0	Activar λ_1
$\alpha = 0.6$ $X_{k+1} = [1.0, 1.5]^T$										
4	1.0	1.5	0.4	0.2	0.8	0.0	0.0	0.0	0.0	
$\alpha = 1.0$ $X_{k+1} = [1.4, 1.7]^T$										
5	1.4	1.7	0.0	0.0	0.8	0.0	0.0	0.0	0.0	
$F(x)_{min} = 0.8$ $X_{min} = [1.4, 1.7]^T$										

Tabla 6.3: Resultados del ***Método del Conjunto Activo*** para $q(X) = (x_1 - 1)^2 + (x_2 - 2.5)^2$; $X_0 = [2, 0]^T$.

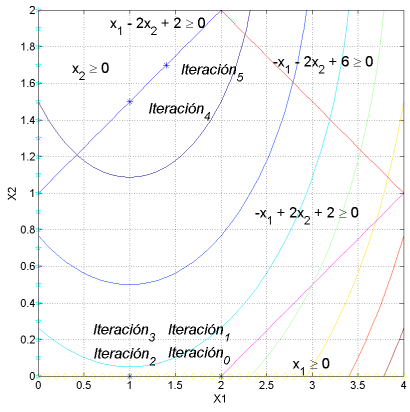


Figura 6.6: Solución del ***Método del Conjunto Activo*** para $q(X) = (x_1 - 1)^2 + (x_2 - 2.5)^2$; $X_0 = [2, 0]^T$.

con una region de factivilidad definida como

$$\mathcal{F} \stackrel{def}{=} \{x \in \mathcal{R} | c_i(x) \geq 0 \text{ para todo } i \in \mathcal{I}\}$$

Asumiremos que $\mathcal{F}$ es un conjunto no vacio .

La mas importante función de barrera es la función de barrera logaritmica, la cual es utilizada para el conjunto de restricciones $c_i \geq 0$, y tiene la forma

$$-\sum_{i \in \mathcal{I}} \log c_i(x)$$

donde log(.) denota el logaritmo natural. Para un problema de optimización con restricciones de desigualdad, la función objetivo/barrera combinada esta dada por

$$P(x; \mu) = f(x) - \mu \sum_{i \in \mathcal{I}} \log c_i(x)$$

donde μ es conocido como el parámetro de barrera. Desde este momento, nos referiremos a $P(x; \mu)$ como la función de barrera logarítmica.

El gradiente de la nueva función $P(x, \mu)$ se calcula como:

$$\nabla P(x, \mu) = \nabla f(x) - \mu \sum_{i=1}^{M} \frac{\nabla C_i(x)}{C_i(x)}$$

y el Hessiano de como:

$$\nabla^2 P(x, \mu) = \nabla^2 f(x) - \mu \sum_{i=1}^{M} \frac{\nabla^2 C_i(x) C_i(x) - \nabla C_i(x) \nabla^T C_i(x)}{C_i^2(x)}$$

6.8.1. Calculo de Gradiente y Hessiano con Restricciones Lineales

En el caso de tener restricciones lineales dada como

$$\begin{bmatrix} c_1(x) \\ c_2(x) \\ c_3(x) \\ \vdots \\ c_M(x) \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,N} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{M,1} & a_{M,2} & a_{M,3} & \cdots & a_{M,N} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} \geq \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_M \end{bmatrix}$$

El gradiente lo podemos calcula como sigue

$$\nabla P(x;\mu) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} - \mu\left(\frac{\partial log(c_1(x))}{\partial x_1} + \frac{\partial log(c_2(x))}{\partial x_1} + \frac{\partial log(c_3(x))}{\partial x_1} + \cdots \frac{\partial log(c_M(x))}{\partial x_1}\right) \\ \frac{\partial f(x)}{\partial x_2} - \mu\left(\frac{\partial log(c_1(x))}{\partial x_2} + \frac{\partial log(c_2(x))}{\partial x_2} + \frac{\partial log(c_3(x))}{\partial x_2} + \cdots \frac{\partial log(c_M(x))}{\partial x_2}\right) \\ \vdots \\ \frac{\partial f(x)}{\partial x_N} - \mu\left(\frac{\partial log(c_1(x))}{\partial x_N} + \frac{\partial log(c_2(x))}{\partial x_N} + \frac{\partial log(c_3(x))}{\partial x_N} + \cdots \frac{\partial log(c_M(x))}{\partial x_N}\right) \end{bmatrix}$$

$$\nabla P(x;\mu) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} - \mu\left(\frac{a_{1,1}}{c_1(x)} + \frac{a_{2,1}}{c_2(x)} + \frac{a_{3,1}}{c_3(x)} \cdots \frac{a_{M,1}}{c_M(x)}\right) \\ \frac{\partial f(x)}{\partial x_2} - \mu\left(\frac{a_{1,2}}{c_1(x)} + \frac{a_{2,2}}{c_2(x)} + \frac{a_{3,2}}{c_3(x)} \cdots \frac{a_{M,2}}{c_M(x)}\right) \\ \frac{\partial f(x)}{\partial x_3} - \mu\left(\frac{a_{1,3}}{c_1(x)} + \frac{a_{2,3}}{c_2(x)} + \frac{a_{3,3}}{c_3(x)} \cdots \frac{a_{M,3}}{c_M(x)}\right) \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} - \mu\left(\frac{a_{1,N}}{c_1(x)} + \frac{a_{2,N}}{c_2(x)} + \frac{a_{3,N}}{c_3(x)} \cdots \frac{a_{M,N}}{c_M(x)}\right) \end{bmatrix}$$

$$\nabla P(x;\mu) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \frac{\partial f(x)}{\partial x_3} \\ \vdots \\ \frac{\partial f(x)}{\partial x_N} \end{bmatrix} - \begin{bmatrix} a_{1,1} & a_{2,1} & a_{3,1} & \cdots & a_{M,1} \\ a_{1,2} & a_{2,2} & a_{3,2} & \cdots & a_{M,2} \\ a_{1,3} & a_{2,3} & a_{3,3} & \cdots & a_{M,3} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_{1,N} & a_{2,N} & a_{3,N} & \cdots & a_{M,N} \end{bmatrix} \begin{bmatrix} 1/c_1(x) \\ 1/c_2(x) \\ 1/c_3(x) \\ \vdots \\ 1/c_M(x) \end{bmatrix}$$

En forma compacta podemos representar el Gradiente como

$$\nabla P(x;\mu) = \nabla f(x) - \mu A^T C$$

donde

$$C = \begin{bmatrix} \frac{1}{c_1(x)} \\ \frac{1}{c_2(x)} \\ \frac{1}{c_3(x)} \\ \vdots \\ \frac{1}{c_M(x)} \end{bmatrix}$$

Para el calculo del Hessiano procedemos a derivar nuevamente el vector Gradiente, llegando a una forma matricial como la siguiente

$$\nabla^2 P(x;\mu) = \nabla^2 f(x) + \mu A^T \widehat{C} A$$

donde $\widehat{C}$ esta dada por la expresión

$$\widehat{C} = \begin{bmatrix} \frac{1}{c_1^2(x)} & 0 & 0 & \cdots & 0 \\ 0 & \frac{1}{c_2^2(x} & 0 & \cdots & 0 \\ 0 & 0 & \frac{1}{c_3^2(x)^2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{c_M^2(x)} \end{bmatrix}$$

6.8.2. Ejemplo

Considere el siguiente problema con una simple variable x

$$\min_x x \ s.a \ x \geq 0, \ 1 - x \geq 0$$

la función de barrera para este caso es

$$P(x;\mu) = f(x) - \mu \log x - \mu \log(1 - x) \tag{6.23}$$

La gráfica correspondiente a la función de barrera dada por la ecuación 6.23, se muestra en la figura 6.7, en esta se presentan diferentes valores de μ dados como 1, 0.4, 0.1 y 0.01. Note que a medida que se reduce el parámetro de barrera μ, el mínimo de la función se acerca a la realidad y las barreras laterales, que impiden que el método salga de la región de factibilidad, se reducen.

6.8.3. Método

De acuerdo con el ejemplo anterior una estrategia, para encontrar la solución utilizando el método de Barrera, es comenzar con un valor μ_0 grande y a medida que se alcanza un mínimo en la función de Barrera 6.23, reducirlo. Esto debe ser repetido hasta alcanzar convergencia. El valor inicial del parámetro de Barrera debe ser lo suficientemente grande de acuerdo con las características de la función a minimizar. Un criterio de paro, recomendable, será cuando $\mu \leq \tau$ y $\| \nabla F(x;\mu) \| \leq \tau$. En el algoritmo 19

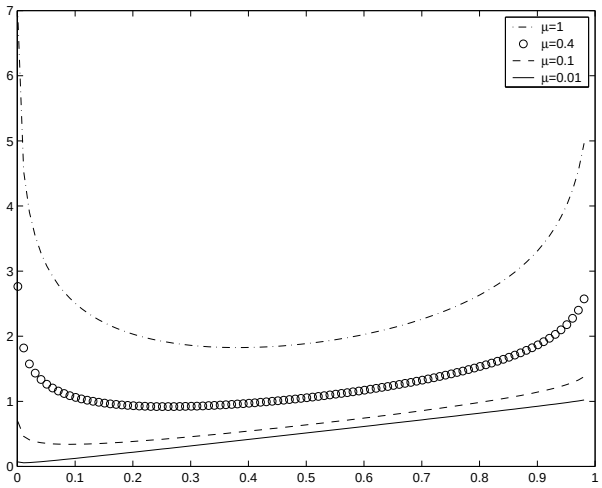


Figura 6.7: Función lineal con diferentes valores de μ

Algoritmo 19 Método de Barrera

MÉTODO DE BARRERA$(f(x), x_0, Ax - b, \mu)$

1 Dado una función diferenciable $f(x)$, un valor inicial x_0, A y μ_0
2 **para** $k \leftarrow 1$ hasta MAX
3 {
4 Encontrar una minimización aproximada de x_k de $P(.;\mu)$,
5 comenzando en un punto x_k^s y finalizar cuando $\| \nabla P(x;\mu) \leq \tau_k \|$
6 **si** la prueba de de convergencia es satisfactoria
7 **entonces** Parar con una solucion aproximada x_k
8 Seleccionar un nuevo parámetro de Barrera $\mu_{k+1} \in (0, \mu_k)$
9 Seleccionar un nuevo punto de arranque x_{k+1}^s
10 }
11 **regresar** x_k

6.8.4. Ejemplo

Calcular el mínimo de la función

$$\begin{aligned} f(x) &= (x_1-2)^2+(x_2-2)^2 \\ & s.a \\ x_1+x_2 &\geq 4 \\ 2x_1-x_2 &\geq 0 \end{aligned}$$

considerando como punto inicial $x_0=[2,4]^T$ y $\mu=100$.

La función de Barrera en este ejemplo esta dada como

$$P(x;\mu)=(x_1-2)^2+(x_2-2)^2-\mu\log(x_1+x_2-4)-\mu\log(2x_1-x_2)$$

y la solución para este problema se da en la tabla 6.4

Note como el valor de μ, se reduce en cada vez que el algoritmo alcanza un estado de sub-óptimo.

6.9. Métodos de Punto Interior.

Por simplicidad, únicamente utilizaremos restricciones de desigualdad

$$\begin{aligned} \text{mín}\, q(x) &= \frac{1}{2}x^TGx+x^Td \\ & s.a \\ Ax &\geq b \end{aligned}$$

donde G es una matriz definida semi positiva, A es una matriz de $m\times n$ definida como $A=[a_i]_{i\in\mathcal{I}}$ y $b=[b_i]_{i\in\mathcal{I}}$ con $\mathcal{I}=\{1,2,3,...,m\}$.

Podemos especializar las condiciones KKT para obtener el conjunto necesario de condiciones: si x^* es solución, hay un vector de multiplicadores de lagrange λ^*, tales que las

k	μ	x_1	x_2
1	100	0.7206	-0.5504
2	100	0.4529	-2.9862
3	100	0.8728	-5.4046
4	100	2.2058	-6.8144
5	100	3.0694	-7.4638
6	100	3.1919	-7.5593
7	100	3.1937	-7.5606
8	100	3.1937	-7.5606
9	100	3.1937	-7.5606
10	66.66666667	2.8886	-5.6015
11	66.66666667	2.9024	-5.75
12	66.66666667	2.9023	-5.7514
13	66.66666667	2.9023	-5.7514
14	44.44444444	2.6565	-4.1549
15	44.44444444	2.6655	-4.2739
16	44.44444444	2.6654	-4.2749
17	44.44444444	2.6654	-4.2749
...	...	...	...
176	4.02E-06	1.999	1.999
177	4.02E-06	1.999	1.999
178	2.68E-06	1.9992	1.9992
179	2.68E-06	1.9992	1.9992
180	2.68E-06	1.9992	1.9992
181	1.79E-06	1.9993	1.9993
182	1.79E-06	1.9993	1.9993
183	1.79E-06	1.9993	1.9993
184	1.19E-06	1.9995	1.9995
185	1.19E-06	1.9995	1.9995
186	1.19E-06	1.9995	1.9995

Tabla 6.4: Mínimo de la función

siguientes condiciones se satisfacen para $(x, \lambda) = (x^*, \lambda^*)$

$$\begin{aligned} Gx - A^T\lambda + d &= 0 \\ Ax - b &\geq 0 \\ (Ax - b)\,\lambda_i &= 0 \\ \lambda_i &\geq 0 \\ i &= 1, 2, .., m \end{aligned} \tag{6.24}$$

Introduciendo el vector laxo $y = Ax - b$, el cual mide la brecha entre la desigualdad y la igualdad, podemos escribir las ecuaciones 6.24 como:

$$\begin{aligned} Gx - A^T\lambda + d &= 0 \\ Ax - y - b &= 0 \\ y_i\lambda_i &= 0 \\ (y, \lambda) &\geq 0 \end{aligned} \tag{6.25}$$

Este sistema de ecuaciones 6.25, lo podemos escribir como

$$\begin{aligned} F(x, y, \lambda) &= \begin{bmatrix} Gx - A^T\lambda + d \\ Ax - y - b = 0 \\ Y\Lambda e \end{bmatrix} \\ (y, \lambda) &\geq 0 \end{aligned}$$

donde $Y = diag[y_1, y_2, ..., y_m]$, $\Lambda = diag[\lambda_1, \lambda_2, ..., \lambda_m]$ y $e = [1, 1, ...1]^T$.

Dada una iteración (x, y, λ), que satisface $(y, \lambda) \geq 0$, podemos definir una medida de dualidad μ como

$$\mu = \frac{1}{m}\sum_{i=1}^{m} y_i\lambda_i = \frac{y^T\lambda}{m}$$

donde esta medida de dualidad μ, representa el promedio del producto $y^T\lambda$. La ruta central $\mathcal{C}$, es el conjunto de puntos $(x_\tau, y_\tau, \lambda_\tau)$, con $\tau > 0$, tal que

$$\begin{aligned} F\left(x_\tau, y_\tau, \lambda_\tau\right) &= \begin{bmatrix} 0 \\ 0 \\ \tau e \end{bmatrix} \\ \left(y_\tau, \lambda_\tau\right) &> 0 \end{aligned} \tag{6.26}$$

El paso genérico es un paso dado por el algoritmo de Newton-Raphson, para el punto actual (x, y, λ) hacia el punto en la ruta central $(x_{\sigma\mu}, y_{\sigma\mu}, \lambda_{\sigma\mu})$ donde σ es un parámetro, en el intervalo $[0, 1]$, seleccionado de acuerdo con las características de la función y elegido de manera manual.

Utilizando el algoritmo de Newton-Raphson, podemos hacer una aproximación lineal para cada una de las funciones, dado como

$$F_j\left(x_{k+1}, y_{k+1}, \lambda_{k+1}\right) = F_j\left(x_k, y_k, \lambda_k\right) + F'_{j,x_k}\Delta x + F'_{j,y_k}\Delta y + F'_{j,\lambda_k}\Delta\lambda$$

para las funciones dadas por la ecuación 6.26, tenemos que la actualización se calcula resolviendo el sistema de ecuaciones dado por

$$\begin{bmatrix} G & -A^T & 0 \\ A & 0 & -I \\ 0 & Y & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta y \end{bmatrix} = \begin{bmatrix} -rd \\ -rb \\ -AYe + \sigma\mu e \end{bmatrix}$$

donde $rd = Gx - A^T\lambda + d$, $rb = Ax - y - b$. Los valores nuevos se calculan haciendo

$$\left(x_{k+1}, y_{k+1}, \lambda_{k+1}\right) = \left(x_k, y_k, \lambda_k\right) + \alpha\left(\Delta x, \Delta y, \Delta\lambda\right)$$

y α es seleccionado para mantener $\left(y_{k+1}, \lambda_{k+1}\right) > 0$

6.9.1. Ejemplo

Calcular el mínimo de la función

$$\begin{aligned} f(x) &= (x_1 - 2)^2 + (x_2 - 2)^2 \\ &\quad s.a \\ x_1 + x_2 &\geq 4 \\ 2x_1 - x_2 &\geq 0 \end{aligned}$$

Considere como punto inicial $x_0 = [2, 4]^T$, $\lambda_0 = [100, 100]^T$, $\alpha = 0.1$ y $\sigma = 0.2$.

Para la primer iteración la matriz característica que debemos resolver es

$$\begin{bmatrix} 2.0 & 0.0 & -1.0 & -2.0 & 0.0 & 0.0 \\ 0.0 & 2.0 & -1.0 & 1.0 & 0.0 & 0.0 \\ 1.0 & 1.0 & 0.0 & 0.0 & -1.0 & -1.0 \\ 2.0 & -1.0 & 0.0 & 0.0 & -1.0 & -1.0 \\ 0.0 & 0.0 & 2.0 & 0.0 & 100.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 100.0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \lambda_1 \\ \lambda_2 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 300.0 \\ -4.0 \\ 0.0 \\ 0.0 \\ -180.0 \\ 20.0 \end{bmatrix}$$

la solución en la primer iteración es $x = [2.0227, 4.0114]^T$, $\lambda = [90.2969, 89.8742]^T$ y $y = [2.0141, 0.02]^T$. Después de Iteracion 381, los resultados son $x = [2.0000, 2.0000]^T$, $\lambda = [0, 0]^T$ y $y = [0.0, 2.0]^T$. Analizando el vector y podemos ver que la solución se encuentra sobre la primer restricción y a dos unidades de distancia de la segunda restricción.

Apéndice A

Algoritmo de factorización de Cholesky

A.1. Factorización de Cholesky

Dado un sistema de ecuaciones $Ax = b$, si sustituimos la matriz A por una matriz factorizada utilizando Choleski, tenemos que $A = L^T L$ y el sistema de ecuaciones a resolver es:

$$[L^T L]x = b \tag{A.1}$$

donde L es una matriz triangular superior, D es una matriz diagonal y L^T una matriz triangular inferior. En lugar de resolver el sistema de ecuaciones A.1, resolveremos dos sistemas de ecuaciones dados por:

$$L^T y = b$$

$$Lx = y$$

A.1.1. Ejemplo

Dada la factorización de una matriz A, determinar

- El producto de las $L^T L$ y
- la solución del sistema de ecuaciones

$$L = \begin{bmatrix} 4 & 2 & 3 & 1 \\ 0 & -5 & 3 & 2 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

$$L^T = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 2 & -5 & 0 & 0 \\ 3 & 3 & 2 & 0 \\ 1 & 2 & 1 & 2 \end{bmatrix}$$

El producto $L^T L$ es :

$$\begin{bmatrix} 4 & 2 & 3 & 1 \\ 0 & -5 & 3 & 2 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 4 & 0 & 0 & 0 \\ 2 & -5 & 0 & 0 \\ 3 & 3 & 2 & 0 \\ 1 & 2 & 1 & 2 \end{bmatrix}$$

lo cual da

$$A = \begin{bmatrix} 16 & 8 & 12 & 4 \\ 8 & 29 & -9 & -8 \\ 12 & -9 & 22 & 11 \\ 4 & -8 & 11 & 10 \end{bmatrix}$$

Comenzaremos por resolver el sistema de ecuaciones $L^T y = b$

$$\begin{bmatrix} 4 & 0 & 0 & 0 \\ 2 & -5 & 0 & 0 \\ 3 & 3 & 2 & 0 \\ 1 & 2 & 1 & 2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 20 \\ -4 \\ 30 \end{bmatrix}$$

Utilizando sustitución hacia adelante (ver ecuación A.2) tenemos

$$y = [0.2500, -3.9000, 3.4750, 17.0375]^T$$

$$y_k = \frac{b_k - \sum_{j=0}^{j<k} a_{k,j} * b_j}{a_{k,k}} \tag{A.2}$$

Como segundo paso resolvemos el sistema $Lx = y$ utilizando sustitución hacia atrás (ver ecuación A.3)

$$\begin{bmatrix} 4 & 2 & 3 & 1 \\ 0 & -5 & 3 & 2 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \begin{bmatrix} 0.2500 \\ -3.9000 \\ 3.4750 \\ 17.0375 \end{bmatrix}$$

con $x = [-1.5130, 2.6744, -2.5219, 8.5188]^T$

$$x_k = \frac{b_k - \sum_{j=k+1}^{j=N} a_{k,j} * y_j}{a_{k,k}} \tag{A.3}$$

A.2. Algoritmo

La factorización de Cholesky solamente es aplicable a sistemas de ecuaciones cuya matriz sea definida positiva y simétrica. El algoritmo lo podemos desarrollar a partir de una matriz A la cual escribimos como el producto de $L^T * L$.

Así, dada una matriz A

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

podemos escribir esta como

$$A = L^T L = \begin{bmatrix} l_{1,1} & 0 & 0 \\ l_{2,1} & l_{2,2} & 0 \\ l_{3,1} & l_{3,2} & l_{3,3} \end{bmatrix} \begin{bmatrix} l_{1,1} & l_{1,2} & l_{1,3} \\ 0 & l_{2,2} & l_{2,3} \\ 0 & 0 & l_{3,3} \end{bmatrix}$$

Para desarrollar el algoritmo hacemos elemento a elemento el producto de las matrices $L^T L$,

así para los elementos del primer renglón se calculan como:

$$
\begin{aligned}
j = 1 \quad & a_{1,1} = l_{1,1} l_{1,1} \\
& l_{1,1} = \sqrt{a_{1,1}} \\
\therefore \quad & a_{1,1} = \sqrt{a_{1,1}} \\
j = 2 \quad & a_{1,2} = l_{1,1} l_{1,2} \\
& l_{1,2} = a_{1,2} / l_{1,1} \\
\therefore \quad & a_{1,2} = a_{1,2} / a_{1,1} \\
j = 3 \quad & a_{1,3} = l_{1,1} l_{1,3} \\
& l_{1,3} = a_{1,3} / l_{1,1} \\
\therefore \quad & a_{1,3} = a_{1,3} / a_{1,1}
\end{aligned}
$$

para el segundo renglón $i = 2$

$$
\begin{aligned}
j = 1 \quad & a_{2,1} = l_{2,1} l_{1,1} \\
& l_{2,1} = a_{2,1} / l_{1,1} \\
\therfore \quad & a_{2,1} = a_{2,1} / a_{1,1} \\
j = 2 \quad & a_{2,2} = l_{2,1} l_{1,2} + l_{2,2} l_{2,2} \\
& l_{2,2} = \sqrt{a_{2,2} - l_{2,1} l_{1,2}} \\
\therefore \quad & a_{2,2} = \sqrt{a_{2,2} - a_{2,1} a_{1,2}} \\
j = 3 \quad & a_{2,3} = l_{2,1} l_{1,2} + l_{2,2} l_{2,3} \\
& l_{2,3} = [a_{2,3} - l_{2,1} l_{1,3}] / l_{2,2} \\
\therefore \quad & a_{2,3} = [a_{2,3} - a_{2,1} a_{1,3}] / a_{2,2}
\end{aligned}
$$

Finalmente para $i = 3$

$$\begin{aligned}
j = 1 \qquad & a_{3,1} = l_{3,1}l_{1,1} \\
& l_{3,1} = a_{3,1}/l_{1,1} \\
\therefore \qquad & a_{3,1} = a_{3,1}/a_{1,1} \\
j = 2 \qquad & a_{3,2} = l_{3,1}l_{1,2} + l_{3,2}l_{2,2} \\
& l_{3,2} = [a_{3,2} - l_{3,1}l_{1,2}]/l_{2,2} \\
\therefore \qquad & a_{3,2} = [a_{3,2} - a_{3,1}a_{1,2}]/a_{2,2} \\
j = 3 \qquad & a_{3,3} = l_{3,1}l_{1,3} + l_{3,2}l_{2,3} + l_{3,3}l_{2,3} \\
& l_{3,3} = \sqrt{a_{3,3} - l_{3,1}l_{1,3} - l_{3,2}l_{2,3}} \\
\therefore \qquad & a_{3,3} = \sqrt{a_{3,3} - a_{3,1}a_{1,3} - a_{3,2}a_{2,3}}
\end{aligned}$$

Dado que la matriz es simétrica, solamente calcularemos los elementos de L^T, con :

$$a_{i,j} = \frac{a_{i,j} - \sum_{k=0}^{k<j} a_{i,k} * a_{j,k}}{a_{j,j}}$$

Para los elementos en la diagonal de L hacemos:

$$a_{i,i} = \sqrt{a_{i,i} - \sum_{k=0}^{k<i} a_{i,k}^2}$$

La implementación en Java es

```
static public void Cholesky(double A[][]) {
  int i, j, k, n, s;
  double fact, suma = 0;

  n = A.length;

  for (i = 0; i < n; i++) {
    for (j = 0; j <= i - 1; j++) {
     suma = 0;
      for (k = 0; k <= j - 1; k++)
         suma += A[i][k] * A[j][k];
      A[i][j] = (A[i][j] - suma) / A[j][j];
    }

    suma = 0;
    for (k = 0; k <= i - 1; k++)
      suma += A[i][k] * A[i][k];
    A[i][i] = Math.sqrt(A[i][i] - suma);
  }
}
```

Ejemplo

Realizar la factorización de Cholesky de la matriz

$$A = \begin{pmatrix} 4 & 2 & 0 & 2 \\ 2 & 5 & 2 & 1 \\ 0 & 2 & 17 & -4 \\ 2 & 1 & -4 & 11 \end{pmatrix}$$

Primer renglón

$a_{1,1} = \sqrt{a_{1,1}} = \sqrt{4} = 2$

Segundo renglón

$a_{2,1} = a_{2,1}/a_{1,1} = 2/2 = 1$

$a_{2,2} = \sqrt{a_{2,2} - a_{2,1}^2} = \sqrt{5 - 1^2} = 2$

Tercer renglón

$a_{3,1} = a_{3,1}/a_{1,1} = 0/2 = 0$

$a_{3,2} = [a_{3,2} - a_{3_1} * a_{2,1}]/a_{2,2} = [1 - 0 * 1]/1 = 1$

$a_{3,3} = \sqrt{a_{3,3} - a_{3,1}^2 - a_{3,2}^2} = \sqrt{17 - 0^2 - 1^2} = 4$

Cuarto renglón

$a_{4,1} = a_{4,1}/a_{1,1} = 2/2 = 1$

$a_{4,2} = [a_{4,2} - a_{4,1} * a_{2,1}]/a_{2,2} = [1 - 1 * 1]/1 = 0$

$a_{4,3} = [a_{4,3} - a_{4,1} * a_{3,1} - a_{4,2}a_{3,2}]/a_{3,3} = [-4 - 1 * 0 - 01]/4 = -1$

$a_{4,4} = \sqrt{a_{4,4} - a_{4,1}^2 - a_{4,2}^2 - a_{4,3}^2} = \sqrt{11 - 1^2 - 0^2 - 1^2} = \sqrt{9} = 3$

Finalmente la matriz L^T queda

$$L^T = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 1 & 4 & 0 \\ 1 & 0 & -1 & 3 \end{bmatrix}$$

A.3. Algoritmo de Cholesky incompleto

Para este algoritmo procedemos de manera similar que el Algoritmo complejo, la diferencia estriba que en aquellas localidades donde exista un cero en la matriz original, los

elementos de la matriz factorizada no se calculan y en su lugar se coloca un cero

Ejemplo

Para la siguiente matriz

$$A = \begin{pmatrix} 3 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 3 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 3 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 3 \end{pmatrix}$$

con un vector de términos independientes

$$b = [1, 2, 3, 2, 1, 2, 3, 2]$$

Calcular:

a) La solución utilizando matriz inversa. La matriz inversa de A es :

$$A^{-1} = \begin{pmatrix} 0.4414 & 0.1466 & 0.0534 & 0.0253 & 0.1776 & 0.0915 & 0.0418 & 0.0224 \\ 0.1466 & 0.3482 & 0.1184 & 0.0534 & 0.0915 & 0.1280 & 0.0720 & 0.0418 \\ 0.0534 & 0.1184 & 0.3482 & 0.1466 & 0.0418 & 0.0720 & 0.1280 & 0.0915 \\ 0.0253 & 0.0534 & 0.1466 & 0.4414 & 0.0224 & 0.0418 & 0.0915 & 0.1776 \\ 0.1776 & 0.0915 & 0.0418 & 0.0224 & 0.4414 & 0.1466 & 0.0534 & 0.0253 \\ 0.0915 & 0.1280 & 0.0720 & 0.0418 & 0.1466 & 0.3482 & 0.1184 & 0.0534 \\ 0.0418 & 0.0720 & 0.1280 & 0.0915 & 0.0534 & 0.1184 & 0.3482 & 0.1466 \\ 0.0224 & 0.0418 & 0.0915 & 0.1776 & 0.0253 & 0.0534 & 0.1466 & 0.4414 \end{pmatrix}$$

y la solución del sistema de ecuaciones calculado con $A^{-1} * b$ es

$$x = [1.4762, 1.9524, 2.3810, 2.1905, 1.4762, 1.9524, 2.3810, 2.1905]^T$$

b) La solución aplicando factorización de Cholesky.
Aplicando la factorización de Cholesky tenemos $A = L^T L$ donde L^T es :

$$L^T = \begin{bmatrix} 1,7321 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0,5774 & 1,9149 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0,5222 & 1,9306 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0,518 & 1,6528 & 0 & 0 & 0 & 0 \\ -0,5774 & -0,1741 & -0,0471 & -0,0148 & 1,6229 & 0 & 0 & 0 \\ 0 & -0,5222 & -0,1413 & -0,0443 & -0,6767 & 1,8021 & 0 & 0 \\ 0 & 0 & -0,518 & -0,1623 & -0,0165 & -0,6057 & 1,8271 & 0 \\ 0 & 0 & 0 & -0,605 & -0,0055 & -0,0169 & -0,6067 & 1,5052 \end{bmatrix}$$

Dado que $L^T Ly = b$, usando sustitución hacia adelante, encontramos:

$$y = \begin{bmatrix} 0.5774 & 1.2185 & 1.8835 & 1.8004 & 1.02328 & 2.0391 & 3.0211 & 3.2970 \end{bmatrix}^T$$

Ahora aplicando sustitución hacia atrás, resolvemos el sistema Lx=y, cuya solución es:

$$x = \begin{bmatrix} 1.4762 & 1.9524 & 2.3810 & 2.1905 & 1.4762 & 1.9524 & 2.3810 & 2.1905 \end{bmatrix}^T$$

Note que la solución en similar a la del inciso (a)

c) La solución aplicando Cholesky incompleto.

La solución aplicando Cholesky incompleto es:

$$\hat{L} = \begin{bmatrix} 1,7321 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ -0,5774 & 1,9149 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -0,5222 & 1,9306 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -0,518 & 1,6528 & 0 & 0 & 0 & -1 \\ -0,5774 & 0 & 0 & 0 & 1,633 & -1 & 0 & 0 \\ 0 & -0,5222 & 0 & 0 & -0,6124 & 1,8309 & -1 & 0 \\ 0 & 0 & -0,518 & 0 & 0 & -0,5462 & 1,8529 & -1 \\ 0 & 0 & 0 & -0,605 & 0 & 0 & -0,5397 & 1,5306 \end{bmatrix}$$

Dado que $\hat{L}^T \hat{L}x = b$, usando sustitución hacia adelante, encontramos:

$$\hat{y} = \begin{bmatrix} 0.5774 & 1.2185 & 1.8835 & 1.8004 & 0.8165 & 1.71300 & 2.6505 & 2.9529 \end{bmatrix}^T$$

Usando sustitución hacia atrás:

$$\hat{x} = \begin{bmatrix} 1.2235 & 1.5969 & 1.9919 & 1.7955 & 1.0737 & 1.5299 & 1.9924 & 1.9292 \end{bmatrix}$$

Apéndice B

Ejemplos

B.1. Resultados

A continuación se presentan los resultados obtenidos con la implementación para la siguiente función:

$$f(x, y, z) = \cos(z^2 - x^2 - y^2) \tag{B.1}$$

con un punto inicial $x = -2$, $y = -1$ y $z = -0.1$

El gradiente para esta función:

$$\nabla f(x, y, z) = \begin{pmatrix} 2x \sin(z^2 - x^2 - y^2) \\ 2y \sin(z^2 - x^2 - y^2) \\ -2z \sin(z^2 - x^2 - y^2) \end{pmatrix} \tag{B.2}$$

Su Hessiano:

$$\nabla^2 f(x, y, z) =$$

$$\begin{pmatrix} 2\sin(z^2 - x^2 - y^2) - 4x^2\cos(z^2 - x^2 - y^2) & -4xy\cos(z^2 - x^2 - y^2) & 4xz\cos(z^2 - x^2 - y^2) \\ -4xy\cos(z^2 - x^2 - y^2) & 2\sin(z^2 - x^2 - y^2) - 4y^2\cos(z^2 - x^2 - y^2) & 4yz\cos(z^2 - x^2 - y^2) \\ 4xz\cos(z^2 - x^2 - y^2) & 4yz\cos(z^2 - x^2 - y^2) & -2\sin(z^2 - x^2 - y^2) - 4z^2\cos(z^2 - x^2 - y^2) \end{pmatrix} \tag{B.3}$$

Iteraciones

La siguiente tabla muestra los valores de $|\nabla f(x_k, y_k, z_k)|^2$ para los diferentes métodos, se asume que el algoritmo ha concluido cuando $|\nabla f(x_k, y_k, z_k)|$ es menor que 1×10^{-6}

k	GC FR	GC PR+	Newton	Secantes	BFGS	DG
0	18.534829	18.534829	18.534829	18.534829	18.534829	18.53482
1	5.9961×10^{-12}	18.534829	19.628691	19.628691	19.628691	4.9263×10
2	1.4080×10^{-15}	5.9961×10^{-12}	1.758437	6137.0593	217310.73	1.4081×10
3	0	5.9952×10^{-12}	6.9861×10^{-7}	144.96555	808444.91	0
4	0	1.4081×10^{-15}	3.4495×10^{-17}	351.09507	674857.35	0
5	0	0	0	107.18124	107.18124	0
6	0	0	0	284.91499	284.91495	0
7	0	0	0	297.44935	297.44960	0
8	0	0	0	3.679177	3.678912	0
9	0	0	0	396.48686	396.47662	0
10	0	0	0	0.505325	0.505207	0
11	0	0	0	0.053475	0.053454	0
12	0	0	0	5.5675×10^{-10}	5.5652×10^{-10}	0
13	0	0	0	5.0543×10^{-18}	5.0476×10^{-18}	0
$f(x,y,z)$	-0.99999999	-0.99999999	1.0	-1.0	-1.0	-0.999999

Resultados

Para el método GC-FR la solución obtenida es:

$$x = -1.588993403399196$$

$$y = -0.794496701699598$$

$$z = -0.120550325305159$$

Para el método GC-PR+ la solución obtenida es

$$x = -1.588993403364383$$

$$y = -0.794496701682191$$

$$z = -0.120550324628847$$

Para el método de Newton la solución obtenida (*que no es mínimo*) es

$$x = -3.173837263856202$$

$$y = -1.586918631928101$$

$$z = -0.158691863192810$$

Para el método de secantes la solución obtenida es

$$x = -9.388337235773939$$
$$y = -4.694168617886971$$
$$z = -0.469416861788697$$

Para el método de secantes con actualización BFGS la solución obtenida es

$$x = -9.388337235773939$$
$$y = -4.694168617886971$$
$$z = -0.469416861788697$$

Para el método de descenso de gradiente la solución obtenida es

$$x = -1.588993398143894$$
$$y = -0.794496699071947$$
$$z = -0.120550326121605$$

Conclusiones

Las conclusiones aplican únicamente para esta función en particular, pues el comportamiento de los algoritmos depende en gran medida de la función y del punto inicial que se dé.

En cuestión de rendimiento el mejor desempeño lo tiene el método de Gradiente Conjugado de Flecher-Reeves junto con el Descenso de gradiente utilizando búsqueda lineal con muestreo hacia atrás y refinando con razón dorada. El peor desempeño se observó en el método de secantes y secantes con actualización BFGS, ambos con 13 iteraciones.

Cabe observar que ambos métodos de secantes llegan a un mínimo, aunque bastante lejano del punto inicial y de las soluciones dadas por los otros algoritmos.

El método de Newton, a pesar de tener un desempeño comparable al de Polak-Ribiere, no tiene un comportamiento deseable, pues la solución a la que llegó es un máximo.

B.2. Ejemplo 2

Dado $f(x,y) = x^2 + y^2 - y^3$ y $x_0 = [-0.1, -0.5]$ calcular x^* por los siguientes métodos:

a)El método descenso de gradiente

b)El método Flecher y Reaves

c)El método de Polak - Ribiere

d)El método de Newton

e)El método de Broyden's o secante

f)El método de Secante con actualización BFGS

El gradiente de la función es:

$$\nabla f(x) = \begin{bmatrix} 2x \\ 2y - 3y^2 \end{bmatrix}$$

El Hessiano de la función es:

$$\nabla^2 f(x) = \begin{bmatrix} 2 & 0 \\ 0 & 2 - 6y \end{bmatrix}$$

Resultados: en las tablas se muestra las iteraciones, el gradiente y en la parte inferior de las tablas se muestra el mínimo

Iteracion	**DG**	**FR**	**PR**
1	1.0529000000E+00	1.0261091560E+00	1.0529000000E+00
2	8.6524469501E-04	2.8641989200E-02	8.2036354532E-04
3	3.7906801746E-07	9.5247029376E-04	9.0815819313E-07
4	4.9409060070E-11	3.1628052834E-05	2.5771723344E-11
5	6.9412498475E-15	1.0500360359E-06	
x	-6.5710455395E-13	-6.0350436068E-09	-1.8607331472E-13
y	4.9395327839E-10	1.6352268545E-08	7.0174475477E-13

Iteracion	**Newton**	**Secante**	**BFGS**
1	1.0261091560E+00	1.0529000000E+00	1.0529000000E+00
2	2.3476331361E-02	5.5113813417E-04	5.5113813417E-04
3	3.8615695475E-04	2.4805908846E-05	2.4105600535E-05
4	1.1170848425E-07	7.0759602555E-09	6.8456011946E-09
5			
x	0.0000000000E+00	0.0000000000E+00	-6.0633893583E-11
y	-4.6795429763E-15	-9.8360197309E-12	1.0984631815E-11

En los resultados se observa que los mejores métodos para esta función son PR+, Newton, secante, BFGS ya que convergen en 4 iteraciones.

B.3. Ejemplo

A continuación se presentan los resultados obtenidos con la implementación para la siguiente función:

$$f(x,y) = 100(y-x^2)^2 + (1-x)^2 \tag{B.4}$$

desarrollando se tiene $f(x,y) = 100y^2 - 200x^2y + 100x^4 + x^2 - 2x + 1$. El punto inicial dado es: $x = 2$ e $y = 2$.

El gradiente para esta función:

$$\nabla f(x,y) = \begin{pmatrix} 400x^3 - 400xy + 2x - 2 \\ 200y - 200x^2 \end{pmatrix} \tag{B.5}$$

Su Hessiano:

$$\nabla^2 f(x,y) = \begin{pmatrix} 1200x^2 - 400y + 2 & -400x \\ -400x & 200 \end{pmatrix} \tag{B.6}$$

Iteraciones

La siguiente tabla muestra el valor de $|\nabla f(x_k, y_k)|^2$ en las primeras iteraciones para los diferentes métodos, se asume que el algoritmo ha concluido cuando $|\nabla f(x_k, y_k)|$ es menor que 1×10^{-6}

k	GC FR	GC PR+	Newton	Secantes	BFGS	
0	2726404.00	2726404.00	2726403.99	2726403.99	2726403.99	2
1	3.102538	2726404.00	3.999926	3.999926	3.999926	
2	9.092944	3.102538	197423.209	4.019346	3.979929	
3	1388.452	9.044424	6.0804×10^{-6}	66835.009	196440.73	
4	2250.645	1384.391	4.6213×10^{-7}	2.584125	4.649582	
5	2687.140	2195.172	1.3331×10^{-28}	8.053000	4.537298	
6	2897.356	113.0063	0	74.92009	7.120902	
7	2977.956	4.429209	0	34.93566	1.751424	
8	2980.577	111.1229	0	26.62128	3.199347	
9	2935.235	168.1488	0	39778.14	3.641651	
10	2860.440	17.50143	0	0.516530	4.272829	
⋮	⋮	⋮	⋮	⋮	⋮	
$f(x,y,z)$	9.0483×10^{-14}	0.0	0.0	0.0	0.0	3.0
Iteraciones	163	53	5	154	131	

Resultados

Para el método GC-FR la solución obtenida es:

$$x = 0.9999997237601747$$

$$y = 0.9999994446670981$$

Para el método GC-PR+ la solución obtenida es

$$x = 1.0000000002512976$$

$$y = 1.000000004827209$$

Para el método de Newton la solución obtenida es

$$x = 1.0$$

$$y = 1.0$$

Para el método de secantes la solución obtenida es

$$x = 0.9999999991748632$$

$$y = 0.9999999983510515$$

Para el método de secantes con actualización BFGS la solución obtenida es

$$x = 0.9999999999946342$$

$$y = 0.9999999999891986$$

Para el método de descenso de gradiente la solución obtenida es

$$x = 1.000000552035356$$

$$y = 1.000001103577654$$

Conclusiones

Las conclusiones aplican únicamente para esta función en particular, pues el comportamiento de los algoritmos depende en gran medida de la función y del punto inicial que se dé.

En cuestión de rendimiento el mejor desempeño lo tiene el método de Newton que además de ser el único en llegar a la solución exacta hace sólo 5 iteraciones. Seguido por Polak-Ribiere en 53 iteraciones. Tanto Gradiente Conjugado de Flecher-Reeves, secantes y secantes con actualización BFGS muestran un desempeño similar con 163, 154 y 131 iteraciones respectivamente. El método Descenso de gradiente utilizando búsqueda lineal con muestreo hacia atrás y refinando con razón dorada exhibe el mayor número de iteraciones, haciendo 2037 iteraciones.