

Máster en
Técnicas Estadísticas | **Doctorado en Estadística**
e Investigación Operativa

Programación Matemática

Trabajo 2

Tema 4

Guillermo Portela Vázquez

UNIVERSIDADE DA CORUÑA

Índice general

2.1. introducción	3
2.2. Descenso por coordenadas	6
2.3. Hooke y Jeeves	8
2.4. Máximo descenso	10
2.5. Método de Newton	11
2.6. Método del Gradiente conjugado	14
Bibliografía	17

2.1. introducción

Para este trabajo, tras observar en la primera oportunidad, que los problemas cuadráticos convergen muy rápido usando métodos que involucren gradiente. Estuve decidiéndome entre varias funciones no convexas, dos candidatas fueron dos generadas por conceptos de superficies de revolución. En concreto las más interesantes fueron:

1. Una extensión de la función seno cardinal como superficie de revolución:

$$f(x_1, x_2) = \frac{\sin(\sqrt{x_1^2 + x_2^2})}{\sqrt{x_1^2 + x_2^2}}$$

2. Una superficie de revolución basada en la función $f(x) = \cos(x) + \frac{|x|}{6}$

$$f(x_1, x_2) = \cos\left(\sqrt{x_1^2 + x_2^2}\right) + \frac{\sqrt{x_1^2 + x_2^2}}{6}$$

En papel he realizado más desarrollos teóricos para la primera, llevándome por ideas muy bonitas que intentaré plasmar bien estructuradas en algún documento futuro. Pero numéricamente no me compensa aplicar en ella los métodos pedidos en el trabajo pues la norma del gradiente de la función es cada vez más pequeña al aumentar la distancia del punto de evaluación respecto del origen. Por lo tanto me quedo con la segunda función. En la figura 2.1 se puede ver el aspecto de la función $f(x_1, x_2) = \cos\left(\sqrt{x_1^2 + x_2^2}\right) + \frac{\sqrt{x_1^2 + x_2^2}}{6}$ con mayor o menor zoom. Además de la función unidimensional con la que se obtiene la bidimensional rotándola alrededor del eje vertical.

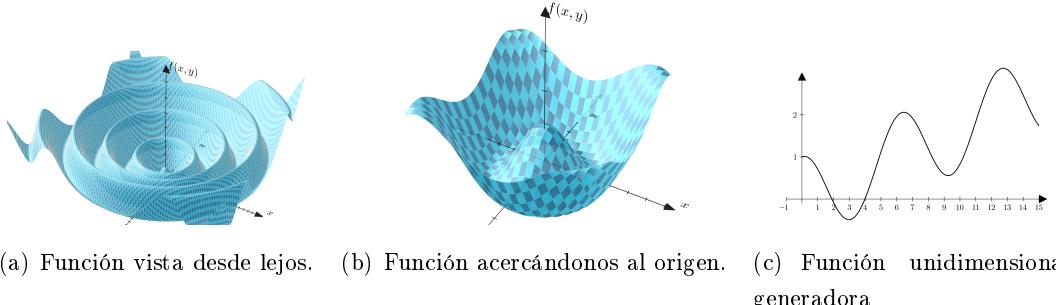


Figura 2.1: Representación gráfica de la función con distintos zooms

Tras probar numéricamente con [R](#) he visto que los resultados de convergencia son poco satisfactorios cuando se tiene simetría circular en la función a minimizar. Por ejemplo el algoritmo de descenso de coordenadas a pesar de no converger en 9 iteraciones (lo cual me interesa pues la exigencia de que los métodos tarden al menos 4 iteraciones es difícil de conseguir si se plantean funciones sencillas), la función objetivo apenas se reduce de una iteración a otra, simplemente alternando entre valores muy cercanos a los puntos $(0, -2,97)$ y el $(0, 2,97)$ muy cerca del óptimo teórico: $D_{\pi - \arcsin(\frac{1}{6})} = \{(x_1, x_2) \in \mathbb{R}^2 : \|(x_1, x_2)\| = \pi - \arcsin(\frac{1}{6})\}$. Esto, junto a otros problemas. Me ha hecho decidirme por complicar todavía más la función, añadiendo un término que rompiera un poco la simetría circular sin alterar demasiado la forma de la función. En resumen, finalmente el problema es:

$$\text{Minimizar: } f(x_1, x_2) = 2 \cos\left(\sqrt{x_1^2 + x_2^2}\right) + \frac{\sqrt{x_1^2 + x_2^2}}{6} - \log(x_1^2 + x_1 \sin(x_2 - x_1) + 2) \quad (2.1)$$

La forma de esta función se puede ver en la figura 2.2 Notar que he multiplicado por dos el

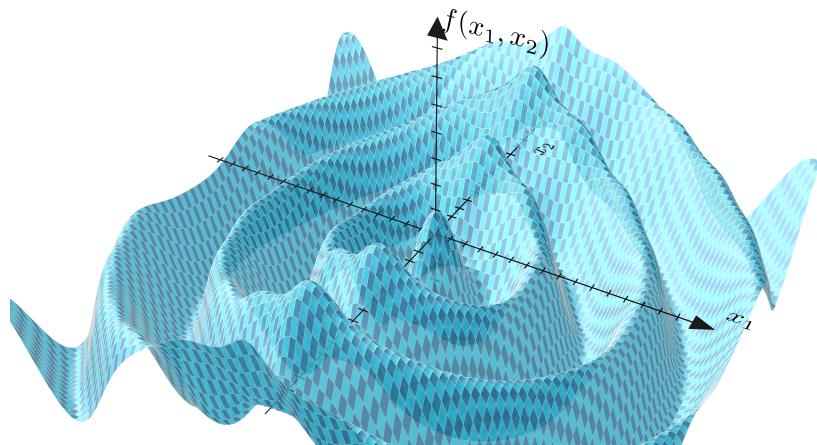


Figura 2.2: Función en la que se centrarán los algoritmos

sumando coseno. También como se ve claramente en los dibujos esta función no es convexa y tiene muchos mínimos locales. Lo cual también va a mostrar un mal desempeño de los algoritmos propuestos en el trabajo. Notar sin embargo que la función cumple:

$$\lim_{\|(x_1, x_2)\| \rightarrow \infty} f(x_1, x_2) = \infty$$

Pues la parte negativa del logaritmo es, en valor absoluto, más pequeña que el término $\frac{\sqrt{x_1^2 + x_2^2}}{6}$ para valores suficientemente grandes de x_1 o x_2 . Este último resultado nos permite afirmar que existe un mínimo global, si bien este puede no ser único. Si bien los algoritmos no van a dar un comportamiento muy bueno, esta es la función que he probado con la que mejor se comportan dichos algoritmos sin hacer que convergiese en una cantidad muy pequeña de pasos. El problema se ha resuelto también usando AMPL \blacktriangleleft probando distintos solvers y tomando de punto de partida $x^1 = \begin{pmatrix} 5 \\ 9 \end{pmatrix}$. Los resultados se pueden ver en la tabla 2.1

Notar que es necesario empezar en un punto distinto al $(0, 0)$ pues ese es el único punto donde la función no es diferenciable. Si no se indica el punto de inicio el programa inicia en $(0, 0)$ y no puede hacer ningún paso.

Solver	x_1	x_2	$f(x_1^t, x_2^t)$
Bonmin	9.42719	0.468807	-4.95194
Couenne	-9.43863	0.430782	-4.9521
Conopt	9.42719	0.468807	-4.95194
Ipopt	9.42719	0.468807	-4.95194

2.2. Descenso por coordenadas

Primeramente voy a reescribir el algoritmo expuesto en los apuntes adaptado a nuestro caso particular, pues en dichos apuntes está expresado para un número de dimensiones arbitrario y puede causar confusión (sin duda a mí me la causó).

- Inicialización

Elegir $\epsilon > 0$. Elegir un punto inicial $x^1 = \begin{pmatrix} x_1^1 \\ x_2^1 \end{pmatrix}$. Definir $\begin{pmatrix} y_1^1 \\ y_2^1 \end{pmatrix} = y^1 = x^1 = \begin{pmatrix} x_1^1 \\ x_2^1 \end{pmatrix}$ y $t = j = 1$

- Paso 1

Calcular λ^j , solución del problema de minimizar $\lambda \in \mathbb{R} f(y^j + \lambda e^j)$ definir $y^{j+1} = y^j + \lambda^j e^j$

- Si $j < n$ reemplazar j por $j + 1$ y repetir el Paso 1.
- si $j = n$, ir al Paso 2.

Notar que si bien la t en este algoritmo indica el número de iteración de las x la j se usa para indexar qué coordenada del punto y se está optimizando.

- Paso 2 Definir $x^{t+1} = y^{n+1}$.

- Si $\frac{\|x^{t+1} - x^t\|}{\|x^t\|} \leq \epsilon$, Fin. Devolvemos x^{t+1} .
- En otro caso, definir $y^1 = x^{t+1}$ y $j = 1$. Reemplazar t por $t + 1$ e ir al Paso 1.

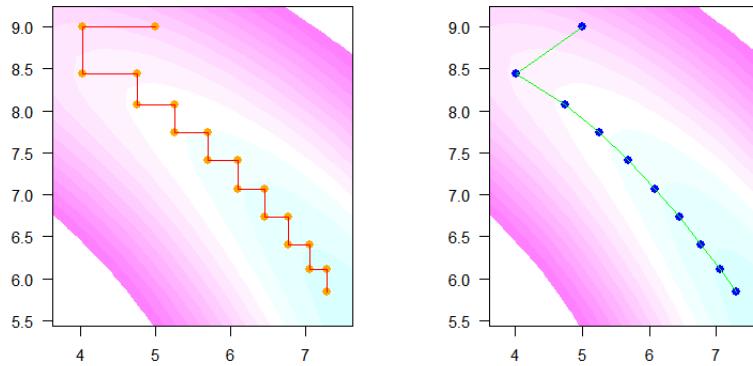
Notar que este algoritmo da libertad sobre como calcular λ^j al inicio del Paso 1. Es decir en principio puedo usar el algoritmo de optimización unidimensional que yo quiera. Voy a definir como x^1 el punto $\begin{pmatrix} 4 \\ 4 \end{pmatrix}$ y utilizar una rutina en el software  que me haga las 10 etapas solicitadas en el enunciado (tal vez menos si converge antes). La rutina de 

Tabla 2.2: tabla descenso coordenadas

t	x_1^t	x_2^t	$f(x_1^t, x_2^t)$	dirección normalizada	tamaño de paso
1	5.000000	9.000000	-2.931286	no procede	no procede
2	4.025934	8.438298	-3.439959	(-0.866285, -0.499550)	1.124417
3	4.748000	8.070521	-3.683062	(0.891073, -0.453859)	0.810332
4	5.257600	7.740601	-3.838107	(0.839436, -0.543458)	0.607074
5	5.694428	7.408321	-3.975269	(0.795907, -0.605419)	0.548844
6	6.093289	7.067942	-4.107304	(0.760670, -0.649139)	0.524355
7	6.456174	6.730312	-4.229423	(0.732123, -0.681172)	0.495660
8	6.777011	6.408953	-4.333636	(0.706531, -0.707682)	0.454101
9	7.053619	6.111476	-4.416928	(0.680954, -0.732326)	0.406207
10	7.289720	5.838615	-4.481529	(0.654330, -0.756209)	0.360828

se puede consultar en el script que se adjunta en el correo donde se envió este pdf. Tras ejecutar dicho script los resultados obtenidos son los de la tabla 2.2

No parece dar problemas claros teóricos. Esto se debe a que el algoritmo es bastante simple y no necesita mucho de condiciones de regularidad. Para obtener los mínimos en la búsqueda de línea se ha usado la función nlm de R. Ahora haciendo uso de las herramientas proporcionadas en la web del máster generamos la imagen



(a) Recorrido de las y^t en el algoritmo. (b) Recorrido de las x^t en el algoritmo

Figura 2.3: Representación gráfica del funcionamiento de descenso por coordenadas

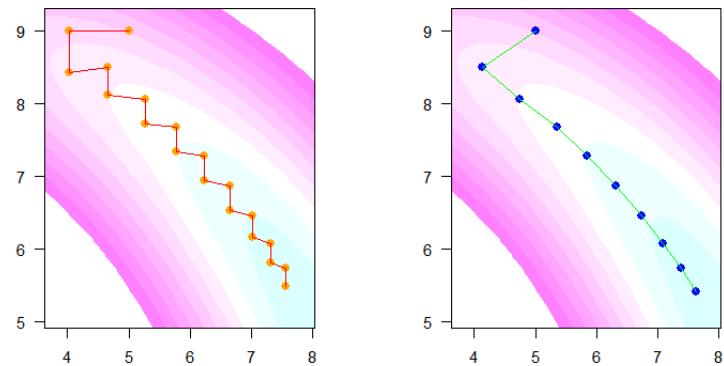
2.3. Hooke y Jeeves

Este algoritmo se parece mucho al de descenso por coordenadas. Sólo hay que añadir un paso intermedio de aceleración. La tabla obtenida al implementar el método es 2.3

Tabla 2.3: Tabla Hooke y Jeeves

t	x_1^t	x_2^t	$f(x_1, x_2)$	dirección normalizada	tamaño de paso
1	5.000000	9.000000	-2.931286	no procede	no procede
2	4.130221	8.498435	-3.459672	(0.866285, 0.866285)	0.120383
3	4.747873	8.059633	-3.670325	(0.815215, 0.815215)	0.101109
4	5.350699	7.680879	-3.854651	(0.846742, 0.846742)	0.092347
5	5.851306	7.281576	-4.013030	(0.781769, 0.781769)	0.104722
6	6.317410	6.865728	-4.168787	(0.746189, 0.746189)	0.112360
7	6.732655	6.457633	-4.306907	(0.713221, 0.713221)	0.112426
8	7.085928	6.078189	-4.415852	(0.681417, 0.681417)	0.106629
9	7.379748	5.731838	-4.496397	(0.646909, 0.646909)	0.099928
10	7.626467	5.411710	-4.556111	(0.610435, 0.610435)	0.095330

Viendo los resultados de la tabla parece haberse conseguido un mínimo mejor algo no funciona en mi código con las direcciones de descenso normalizadas. por ello anoto las direcciones de descenso sin normalizar en la tabla 2.3. En los resultados dados por el algoritmo de Hooke y Jeeves se ve que este algoritmo funciona ligeramente mejor que el de descenso por coordenadas. Ahora haciendo uso de las herramientas proporcionadas en la web del máster generamos la imagen 2.3



(a) Recorrido de las y^t en el algoritmo.
(b) Recorrido de las x^t en el algoritmo

Figura 2.4: Representación gráfica del funcionamiento de Hooke y Jeeves

Tabla 2.4: Dirección de Hooke y Jeeves sin normalizar

t	dirección descenso
1	no procede
2	(0.104286, 0.060138)
3	(0.082426, -0.058558)
4	(0.078194, -0.049129)
5	(0.081868, -0.065301)
6	(0.083842, -0.074802)
7	(0.080184, -0.078804)
8	(0.072659, -0.078042)
9	(0.064645, -0.076202)
10	(0.058192, -0.075507)

2.4. Máximo descenso

Para este método vamos a necesitar calcular el gradiente de la función objetivo. Es decir necesitamos $\nabla f(x_1, x_2)$. Calculemos pues las derivadas parciales.

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = \frac{x_1}{6\sqrt{x_1^2 + x_2^2}} - \frac{2x_1 \sin(\sqrt{x_1^2 + x_2^2})}{\sqrt{x_1^2 + x_2^2}} - \frac{\cos(x_1 - x_2) + 2x_1}{\sin(x_1 - x_2) + x_1^2 + 3} \quad (2.2)$$

La expresión para la segunda derivada parcial es parecida:

$$\frac{\partial f(x_1, x_2)}{\partial x_2} = \frac{x_2}{6\sqrt{x_2^2 + x_1^2}} - \frac{2x_2 \sin(\sqrt{x_2^2 + x_1^2})}{\sqrt{x_2^2 + x_1^2}} + \frac{\cos(x_2 - x_1)}{-\sin(x_2 - x_1) + x_2^2 + 3} \quad (2.3)$$

Notar que los denominadores no se anulan para ningún valor de (x_1, x_2) salvo el origen. La tabla obtenida al implementar este método es 2.4 Vemos que x^t tiene una trayectoria

Tabla 2.5: tabla del algoritmo de máximo descenso

	x_1^t	x_2^t	$f(x_1^t, x_2^t)$	dirección normalizada	tamaño de paso
1	5.000000	9.000000	-2.931286	(-0.323279, -0.946304)	0.000000
2	2.546857	1.819159	-3.795858	(0.946304, -0.323279)	7.588306
3	9.421782	-0.529476	-4.940879	(0.323279, 0.946304)	7.265032
4	9.461181	-0.414149	-4.942201	(-0.946304, 0.323279)	0.121871
5	-13.775218	7.523948	-4.641960	(-0.323279, -0.946304)	24.554910
6	-15.577264	2.248992	-4.883303	(0.946304, -0.323279)	5.574275
7	-9.448906	0.155404	-4.951104	(0.323278, 0.946304)	6.476101
8	-9.437362	0.189197	-4.951237	(-0.946304, 0.323278)	0.035710
9	-9.447995	0.192829	-4.951353	(0.323278, 0.946304)	0.011236
10	-9.437774	0.222747	-4.951455	(0.000000, 0.000000)	0.031615

más a trompicones. Pegando saltos bastante grandes en algunos pasos. Se ve mejor en la imagen de las curvas de nivel representando la trayectoria 2.4. Sin embargo el método acaba proponiendo un punto muy bueno como óptimo.

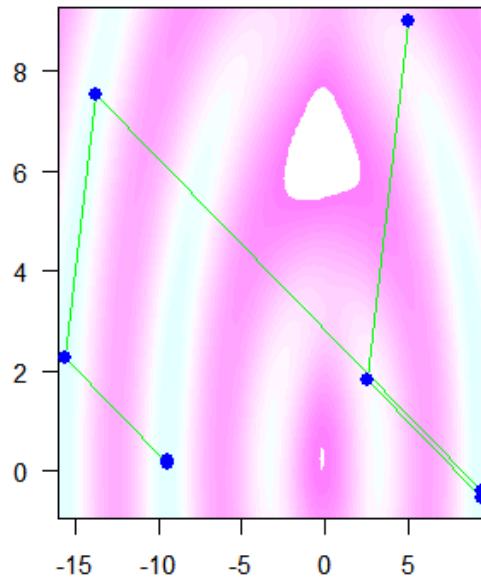


Figura 2.5: Recorrido realizado por el algoritmo de máximo descenso

2.5. Método de Newton

Para este método también necesitaremos la Hessiana. En este punto he recurrido a una calculadora de derivadas de internet porque hacerlas a mano es una tortura y no creo que ese sea el objetivo del máster.

Denotando la hessiana como

$$Hess = \begin{pmatrix} \frac{\partial^2 f(x_1, x_2)}{\partial x_1^2} & \frac{\partial^2 f(x_1, x_2)}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f(x_1, x_2)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x_1, x_2)}{\partial x_2^2} \end{pmatrix}$$

Los 4 elementos, más bien funciones, que componen esa matriz son los siguientes:

Primera parcial segunda $\frac{\partial^2 f(x_1, x_2)}{\partial x_1^2}$

$$\frac{\partial^2 f(x_1, x_2)}{\partial x_1^2} = -\frac{2 \sin(\sqrt{x_1^2 + x_2^2})}{\sqrt{x_1^2 + x_2^2}} + \frac{2x_1^2 \sin(\sqrt{x_1^2 + x_2^2})}{(x_1^2 + x_2^2)^{\frac{3}{2}}} - \frac{2x_1^2 \cos(\sqrt{x_1^2 + x_2^2})}{x_1^2 + x_2^2} \\ (2.4)$$

$$-\frac{2 - \sin(x_1 - x_2)}{\sin(x_1 - x_2) + x_1^2 + 3} + \frac{(\cos(x_1 - x_2) + 2x_1)^2}{(\sin(x_1 - x_2) + x_1^2 + 3)^2} + \frac{1}{6\sqrt{x_1^2 + x_2^2}} - \frac{x_1^2}{6(x_1^2 + x_2^2)^{\frac{3}{2}}} \\ (2.5)$$

Segunda parcial segunda $\frac{\partial^2 f(x_1, x_2)}{\partial x_1 \partial x_2}$

$$\frac{\partial^2 f(x_1, x_2)}{\partial x_1 \partial x_2} = \frac{2x_1 x_2 \sin(\sqrt{x_2^2 + x_1^2})}{(x_2^2 + x_1^2)^{\frac{3}{2}}} - \frac{2x_1 x_2 \cos(\sqrt{x_2^2 + x_1^2})}{x_2^2 + x_1^2} \\ (2.6)$$

$$+ \frac{\sin(x_2 - x_1)}{-\sin(x_2 - x_1) + x_1^2 + 3} - \frac{\cos(x_2 - x_1)(\cos(x_2 - x_1) + 2x_1)}{(-\sin(x_2 - x_1) + x_1^2 + 3)^2} - \frac{x_1 x_2}{6(x_2^2 + x_1^2)^{\frac{3}{2}}} \\ (2.7)$$

Tercera parcial segunda $\frac{\partial^2 f(x_1, x_2)}{\partial x_2 \partial x_1}$

$$\frac{\partial^2 f(x_1, x_2)}{\partial x_2 \partial x_1} = -\frac{2 \sin(\sqrt{x_2^2 + x_1^2})}{\sqrt{x_2^2 + x_1^2}} + \frac{2x_2^2 \sin(\sqrt{x_2^2 + x_1^2})}{(x_2^2 + x_1^2)^{\frac{3}{2}}} \\ (2.8)$$

$$- \frac{2x_2^2 \cos(\sqrt{x_2^2 + x_1^2})}{x_2^2 + x_1^2} - \frac{\sin(x_2 - x_1)}{-\sin(x_2 - x_1) + x_1^2 + 3} + \frac{\cos^2(x_2 - x_1)}{(-\sin(x_2 - x_1) + x_1^2 + 3)^2} \\ (2.9)$$

$$+ \frac{1}{6\sqrt{x_2^2 + x_1^2}} - \frac{x_2^2}{6(x_2^2 + x_1^2)^{\frac{3}{2}}} \\ (2.10)$$

Cuarta parcial segunda $\frac{\partial^2 f(x_1, x_2)}{\partial x_2^2}$

$$\frac{\partial^2 f(x_1, x_2)}{\partial x_2^2} = -\frac{2 \sin(\sqrt{x_2^2 + x_1^2})}{\sqrt{x_2^2 + x_1^2}} + \frac{2x_2^2 \sin(\sqrt{x_2^2 + x_1^2})}{(x_2^2 + x_1^2)^{\frac{3}{2}}} - \frac{2x_2^2 \cos(\sqrt{x_2^2 + x_1^2})}{x_2^2 + x_1^2} \\ (2.11)$$

$$- \frac{\sin(x_2 - x_1)}{-\sin(x_2 - x_1) + x_1^2 + 3} + \frac{\cos^2(x_2 - x_1)}{(-\sin(x_2 - x_1) + x_1^2 + 3)^2} + \frac{1}{6\sqrt{x_2^2 + x_1^2}} - \frac{x_2^2}{6(x_2^2 + x_1^2)^{\frac{3}{2}}} \\ (2.12)$$

Tabla 2.6: Tabla del algoritmo de Newton

t	x_1^t	x_2^t	$f(x_1^t, x_2^t)$	dirección normalizada	tamaño de paso
1	5.000000	9.000000	-2.931286	(0.597080 , -0.802181)	6.779233
2	9.047748	3.561826	-4.723327	(0.087412 , -0.996172)	0.994746
3	9.134700	2.570888	-4.876885	(0.212828 , -0.977090)	0.668591
4	9.276995	1.917614	-4.917986	(0.152369 , -0.988324)	0.592186
5	9.367226	1.332342	-4.940568	(0.094594 , -0.995516)	0.469893
6	9.411675	0.864557	-4.949653	(0.048224 , -0.998837)	0.287868
7	9.425557	0.577024	-4.951769	(0.000000 , 0.000000)	0.000000

Tras implementar el algoritmo en obtenemos la tabla 2.6

El algoritmo llega, cada vez dando pasos más pequeños, a un candidato a óptimo muy bueno. La imagen correspondiente a este método es la figura 2.6

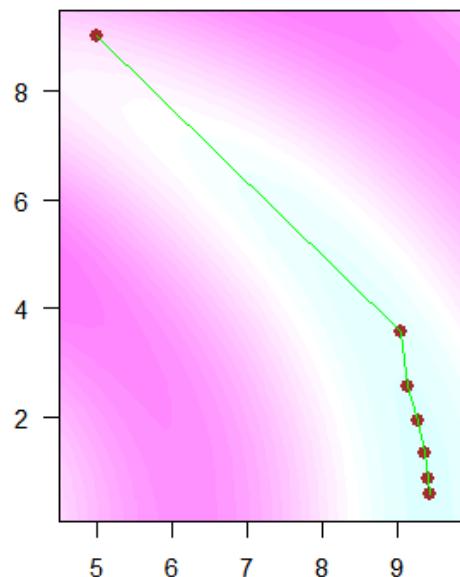


Figura 2.6: Recorrido realizado por el algoritmo de Newton

2.6. Método del Gradiente conjugado

Para este método vamos a necesitar usar las derivadas parciales ya calculadas previamente. El código está implementado en [R](#). Tras implementarlo se obtiene la tabla 2.7 A

Tabla 2.7: Tabla gradiente conjugado

	x_1^t	x_2^t	$f(x_1^t, x_2^t)$	dirección normalizada	tamaño de paso
1	5.000000	9.000000	-2.931286	(-0.323279, -0.946304)	1.543068
2	2.546861	1.819171	-3.795858	(0.805873, -0.592088)	0.508242
3	8.981121	-2.908183	-4.845639	(0.443267, 0.896389)	0.064815
4	9.267365	-2.329331	-4.868313	(0.193054, 0.981188)	0.842616
5	9.530712	-0.990881	-4.910297	(-0.478310, 0.878191)	0.442279
6	9.392963	-0.737969	-4.935894	(0.105617, 0.994407)	0.077264
7	9.478880	0.070962	-4.948345	(-0.101516, 0.994834)	0.400093
8	9.434595	0.504944	-4.951838	(-0.164597, -0.986361)	0.070434
9	9.427729	0.463798	-4.951940	(0.000000, 0.000000)	0.041715

la vista de los resultados vemos que el método del gradiente conjugado ha sido capaz de igualar a todos los solvers menos a Couenne. La imagen que representa este proceso es la 2.7.

Si bien este último método da un buen resultado. Alcanzando el mismo punto que varios solvers de AMPL [▲](#). Cabe destacar que se trata de un algoritmo de minimización local. Si al método de gradiente conjugado le propongo tomar como punto inicial $\begin{pmatrix} 75 \\ -90 \end{pmatrix}$ el recorrido del algoritmo sería el de la figura 2.8 Donde, tras haber aumentado el número de pasos, el criterio de parada se activa tras recorrer "discos localmente óptimos" muy alejados del óptimo global y valores de la función objetivo rondando el 15,52253.

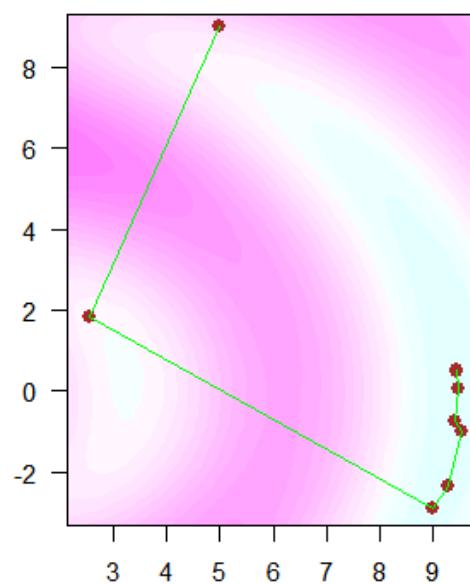


Figura 2.7: Recorrido realizado por el algoritmo de gradiente conjugado

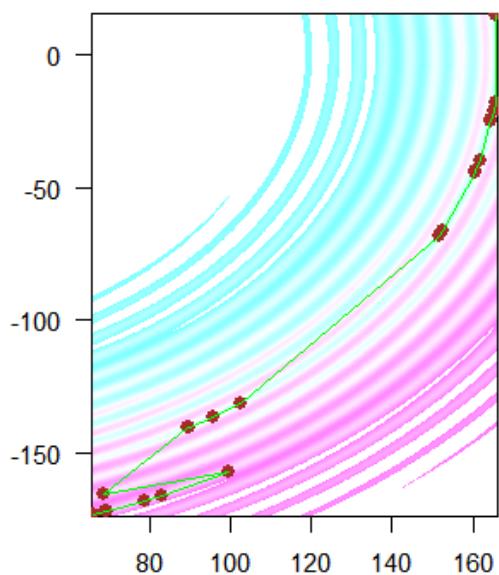


Figura 2.8: Recorrido iniciado en otro punto

Bibliografía

- [1] Gonzalez, J. (Septiembre 2022) Apuntes Programación Matemática