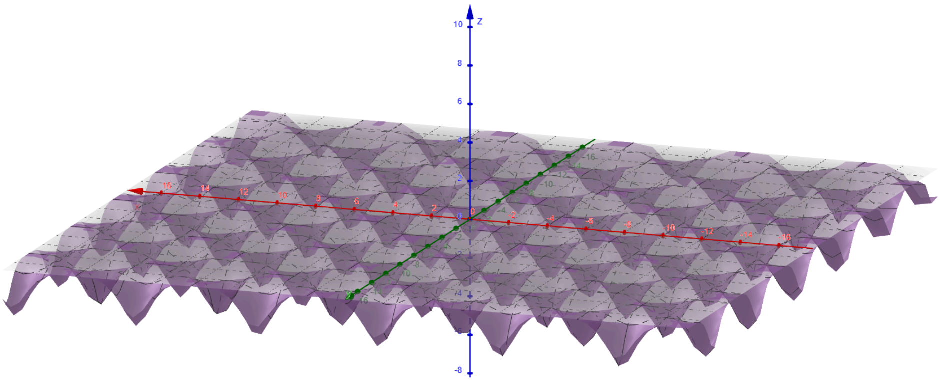
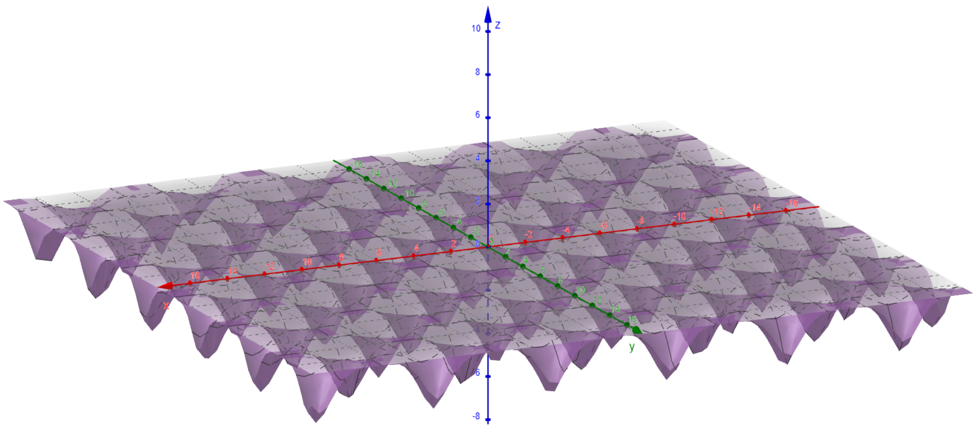
Tarea de Modelos de Optimización.

Nombre: Ronald Cabrera Martínez.
Grupo: 311.

Función objetivo a minimizar:

$$f(x, y) = -\tan^2(0.5sen(x) + 0.5sen(y)).$$





Estamos en presencia de un problema de optimización no lineal sin restricciones y con variables continuas.

<u>Analicemos ahora la existencia o no de solución:</u>

Recordando…..

Por el **teorema de Weierstrass**: Si f es **continua** y el conjunto factible S es **no vacío, cerrado y acotado**, entonces existe al menos un mínimo global.

Sea:

$$f(x, y) = -tan^2(0.5sen(x) + 0.5sen(y)) := g(h(x, y))$$

donde $h(x, y) = 0.5sen(x) + 0.5sen(y)$ y $g(u) = -tan^2(u)$.

Para todo x,y ∈ R se tiene que $sen(x), sen(y)$ ∈ [-1,1] por tanto h(x,y)=[-1,1], $g(u) = -tan^2(u)$ es continua en [-1,1] ya que ($u \leq 1 < \pi/2$) por lo que se cumple Weierstrass para g y podemos garantizar la existencia de solución óptima.

En el intervalo [-1,1] la función $tan^2(u)$ alcanza su máximo en los extremos u=±1.

. $u = 1$ cuando $sen(x) = sen(y) = 1$ o sea cuando,

$x = (\pi/2) + 2k_x\pi$

$y = (\pi/2) + 2k_y\pi$

. $u = -1$ cuando $sen(x) = sen(y) = -1$ o sea cuando,

$x = (3\pi/2) + 2k_x\pi$

$y = (3\pi/2) + 2k_y\pi$

Por tanto el problema tiene infinitos óptimos globales y el valor óptimo es ≈−2.4255.

Analicemos ahora la convexidad:

Para verificar que nuestra función sea convexa debemos ver que su hessiana sea semidefinida positiva.

Derivadas parciales de primer orden:

$$fx = \partial f/\partial x$$

$$fy = \partial f/\partial y$$

La hessiana queda:

$$H_{f(x,y)} = \begin{bmatrix} fxx & fxy \\ fyx & fyy \end{bmatrix}$$

En $(x, y) = (0, 0)$:

$$H_{f(0,0)} = \begin{bmatrix} -1/2 & -1/2 \\ -1/2 & -1/2 \end{bmatrix}$$

Sus autovalores son {0,−1}, por lo que es semidefinida negativa.
Con esto concluimos que no estamos en presencia de un problema convexo.

Existen mínimos locales además de los óptimos globales ?

Notemos que como estamos trabajando en un problema irrestricto los mínimos locales serán aquellos cuyo $\nabla f(x\ *) = 0$ (condición de primer orden) y su matriz hessiana sea definida positiva (condición de segundo orden).

Las condiciones de primer orden dan:

$$fx =- cos(x)\, tan(u) sec^{2}(u), \qquad fy =- cos(y)\, tan(u) sec^{2}(u)$$

Por tanto, los puntos críticos aparecen cuando:
. $u = 0 \Leftrightarrow sinx + siny = 0.$
. $o\ bien\ cosx = 0\ y\ cosy = 0\ simultáneamente \Leftrightarrow x = \pi/2 + k\pi,\ y = \pi/2 + m\pi.$

Para u = 0 obtenemos Hessiana semidefinida negativa por lo que no hay mínimos locales , para el segundo caso los valores posibles de u son {-1,0,1} cumpliendo mínimo local para ± 1, que es justamente los valores de u que cumplen mínimo global. Por tanto concluimos que no existen mínimos locales que no sean mínimos globales .

Descripción de los algoritmos utilizados:

Método de Newton

El primer algoritmo que escogimos para nuestra función fue el algoritmo de Newton. Este es un algoritmo para problemas irrestrictos que se aprovecha del gradiente y la hessiana de la función del problema por lo que f debe ser dos veces diferenciable y la función de nuestro problema cumple esto.
Newton es un método cuadrático e iterativo que se basa en descenso por direcciones .
La idea fundamental en la que se basa el método de Newton es la siguiente: la función a minimizar f(x) se aproxima en una vecindad del punto de iteración x_k por una función cuadrática $q_k(x)$. La nueva aproximación x_{k+1} de la solución x* se define como el punto en que $q_k(x)$ alcanza su valor mínimo. La interpretación geométrica del método de Newton es que, en cada iteración, equivale a ajustar la función parabólica q(x) (q(x) es la expansión cuadrática de Taylor de f(x)) a la función gráfica de f(x) en el valor de prueba x_k, con la misma pendiente y curvatura que la gráfica en ese punto, y luego proceder al máximo o mínimo de dicha parábola .

Algoritmo:
1- Fijo x_0, k=0 **.**

. Se elige un punto inicial x_0 y se arranca el contador de iteraciones k = 0.

2- Si $\nabla f(x_k) \neq 0$, buscar d_k tal que $\nabla^2 f(x_k) d_k = -\nabla f(x_k)$.

o sea ,

$$d_k = -(\nabla^2 f(x_k))^{-1} \nabla f(x_k)$$

. d_k es la dirección de Newton, que apunta hacia donde el modelo cuadrático cree que está el mínimo.

. Si la matriz Hessiana es definida positiva, d_k es una dirección de descenso.

3- Hallar $\alpha_k = \min_{\alpha \geq 0} f(x_k + \alpha d_k)$.

. Se busca el mejor tamaño de paso α_k en la dirección d_k. Aunque en teoría Newton puro usa un tamaño de paso fijo , en práctica con tal de mejorar el algoritmo se hace una búsqueda lineal con backtracking para garantizar que el paso realmente disminuya la función.

4- Actualizar: $x_{k+1} = x_k + \alpha_k d_k$ **,** k = k+1**.**
5- Repetir hasta criterio de parada.

. Nuestro criterio de parada puede ser variado , normalmente se pone que se cumpla $\| \nabla f(x_k) \| \approx 0$ y también ponemos un límite de número de iteraciones.

Para Newton tenemos algunas limitaciones fundamentales , una es el cálculo de la inversa de la matriz hessiana en cada iteración lo cual puede llegar a ser computacionalmente costoso, y el otro gran problema es que como tenemos que usar la inversa de la hessiana , si en algún punto x_k esta es singular entonces nuestro algoritmo falla. También si el punto inicial está lejos del óptimo, Newton puede tardar en converger o incluso desviarse hacia un punto no deseado (máximo o silla).

Sin embargo vale la pena destacar que si nuestro punto inicial x_k está lo suficientemente cerca de un x* y $\nabla f(x^*) = 0$, $\nabla^2 f(x^*)$ definida positiva , entonces el método tendrá para nuestra función una convergencia cuadrática .

En la implementación hemos hecho Newton con tamaño de paso fijo y con tamaño de paso ajustable mediante la búsqueda lineal.

Método Quasi-Newton (BFGS)

El otro algoritmo que usaremos para la resolución de nuestro problema es el de Quasi-Newton.

Este algoritmo tiene una formulación general que es relativa a la de Newton, pero la inversa de la matriz Hessiana $\nabla^2 f(x_k)$ se sustituye por una aproximación W_k de dicha matriz, la cual es actualizada en cada iteración mediante una fórmula de corrección, en nuestro caso usaremos BFGS, también Quasi-Newton implementa la búsqueda lineal para el tamaño de paso α_k (Se usan reglas como Armijo o Wolfe para decidir cuándo parar la búsqueda y aceptar un α_k) .

Ideas generales del Algoritmo:

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

.s_k es el desplazamiento entre iteraciones.

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

.y_k es el cambio de gradiente .

. $W_k \approx [\nabla^2 f(x_k)]^{-1}$.

. W_k definida positiva.

. $W_{k+1} y_k = s_k$. (aquí garantizamos que nuestra matriz W se comporte igual a la inversa de la matriz Hessiana).

. $W_{k+1} = W_k + B_k$, B_k mínima.

donde B_k queda de la siguiente forma usando BFGS:

$$W_{k+1} = W_k + \frac{s_k y_k^T W_k + W_k y_k s_k^T}{y_k^T s_k} - \left(1 + \frac{y_k^T W_k y_k}{y^T s}\right) \frac{s_k s_k^T}{y_k^T s_k}$$

De $M_k \approx \nabla^2 f(x_k)$ se tiene :

$$M_{k+1} = M_k + \frac{y y^T}{y^T s} - \frac{M_k s_k s_k^T M_k}{s_k^T M_k s_k}$$

Algunas de las limitaciones de BFGS son parecidas a las de Newton , aunque evita el cálculo directo de la inversa de la hessiana , si la hessiana es singular la aproximación puede volverse inestable; al igual que Newton un punto inicial muy lejos de algun optimo puede conllevar a convergencia a un punto no deseado (máximo o silla). Aunque evita calcular la Hessiana exacta, BFGS aún necesita almacenar y actualizar una matriz n×n lo cual es bastante costoso para un n grande.

Destacar que BFGS no necesita calcular inversa de hessiana en cada iteración, también tiene mayor estabilidad numérica al forzar que

la hessiana sea definida positiva lo cual nos da mayores posibilidades de encontrar un mínimo y tiene una pequeña ventaja de implementación sobre Newton ya que solo requiere de gradientes .

Comparación de resultados:

Para evaluar el desempeño de los métodos **Newton** y **Quasi-Newton (BFGS)** sobre la función:

$$f(x, y) = -\tan^2(0.5sen(x) + 0.5sen(y))$$

se realizaron experimentos variando:

. Los **puntos iniciales** (x_0,y_0).

. Los **tamaños de paso (α)**.

. máximo de iteraciones : 100

Resultados numéricos obtenidos:

Se probaron 4 configuraciones representativas de puntos iniciales:

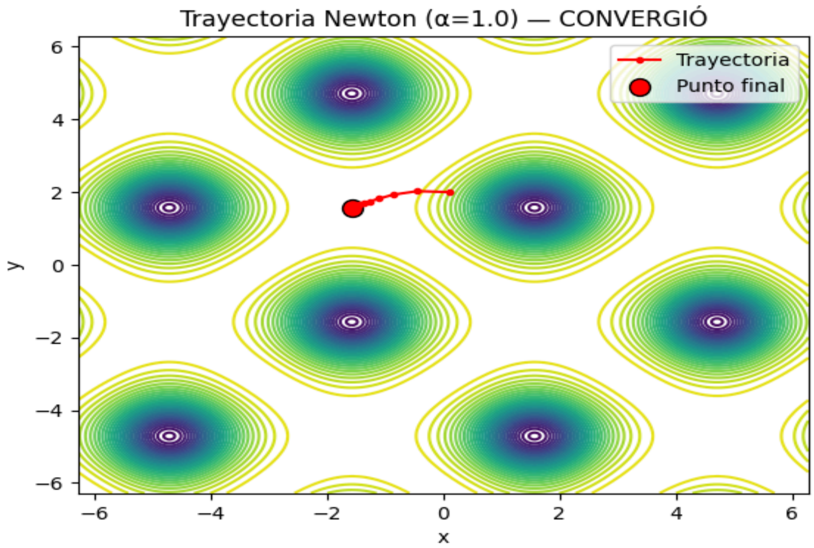
1. **Cerca del mínimo global:** $(x_0,y_0) = (1.5,1.5)$
2. **Lejos del mínimo, en otra "cuenca" de atracción:** $(x_0,y_0)=(4.5,4.5)$.
3. **Región plana o inestable:** $(x_0,y_0)=(0,\pi)$.
4. **No cumple sen(x)=sen(y):** $(x_0,y_0)=(0.1,2.0)$.

Método	(x_0,y_0)	α inicial	Iteraciones	f(x*, y*)	Observaciones
Newton	(0.1, 2.0)	1.0 (fijo)	12	0.0	Converge a un máximo local.
Newton damped	(0.1, 2.0)	adaptativo	100	-0.30	divergencia detectada por iteraciones máximas sin lograr progresar
BFGS	(0.1, 2.0)	adaptativo	8	-2.4255	Estable; alcanza mínimo global.

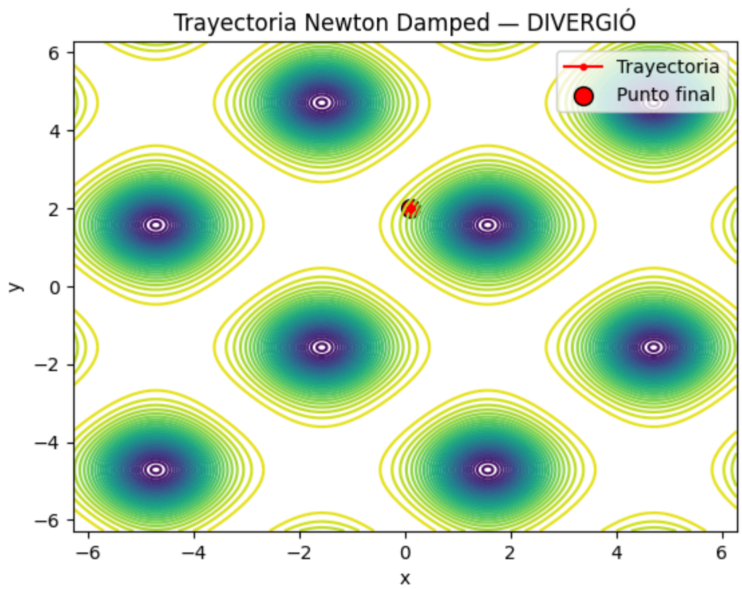
Newton	(1.5, 1.5)	2.0 (fijo)	41	0.0	Converge a máximo local no global debido a que el paso grande se salta el global y cae en un valle plano de la función .
Newton	(1.5,1.5)	1 (fijo)	3	-2.4255	Converge estable al óptimo global(no hace salto hacia región crítica).
Newton damped	(1.5, 1.5)	adaptativo	3	-2.4255	Convergencia rápida (Newton cerca del mínimo global muestra su cuasicuadrática).
BFGS	(1.5,1.5)	adaptativo	4	-2.4255	Robusto; converge sin problemas al óptimo global desde este inicio.
Newton	(4.5,4.5)	1.0 (fijo)	6	0.0	Converge a máximo local no global debido a que el paso grande se salta el global y cae en un valle plano de la función .
Newton	(4.5,4.5)	0.5 (fijo)	18	-2.4255	Converge al óptimo global sin problemas.
Newton damped	(4.5,4.5)	adaptativo	3	-2.4255	Converge al óptimo global sin problemas.
BFGS	(4.5,4.5)	adaptativo	4	-2.4255	Converge al óptimo global sin problemas.
Newton	(0,π)	1.0 (fijo)	0	0.00	converge hacia máximo local porque el gradiente inicial lleva allí.
Newton damped	(0,π)	adaptativo	0	0.00	converge hacia el máximo local porque el gradiente inicial lleva allí.
BFGS	(0,π)	adaptativo	0	0.00	converge hacia el máximo local porque el gradiente inicial lleva allí.

Tabla con muestra visual del comportamiento de los puntos analizados:

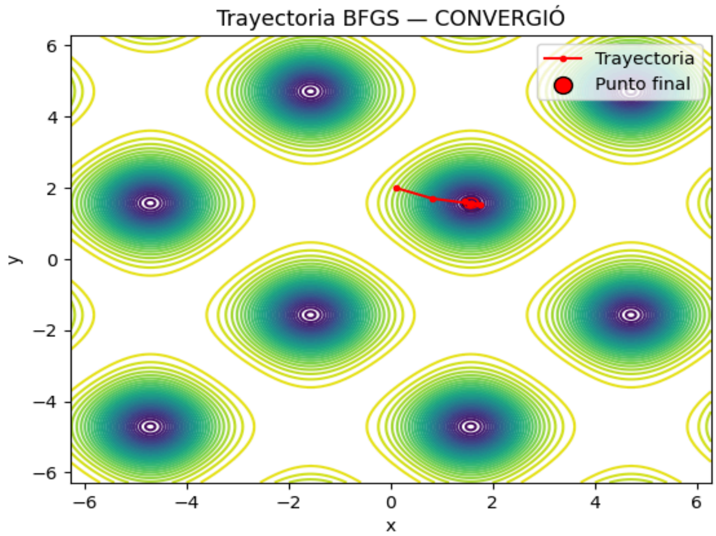
Método	(x_0,y_0)	α inicial	(x*,y*)	f(x*, y*)
Newton	(0.1,2)	1.0 (fijo)	(-1.558, 1.577)	0.0



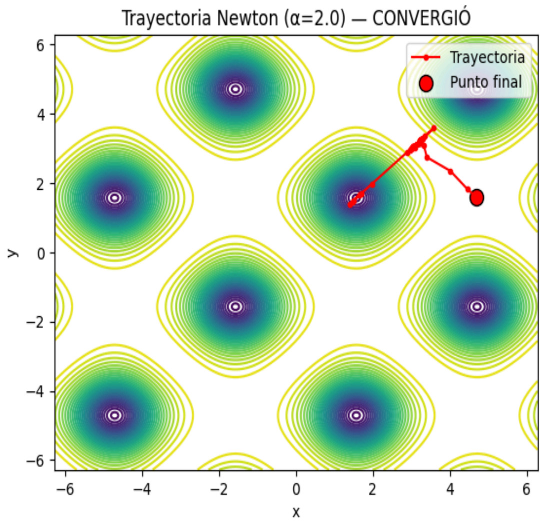
Newton damped	(0.1,2)	adaptativo	(0.099 2.0)	-0.30

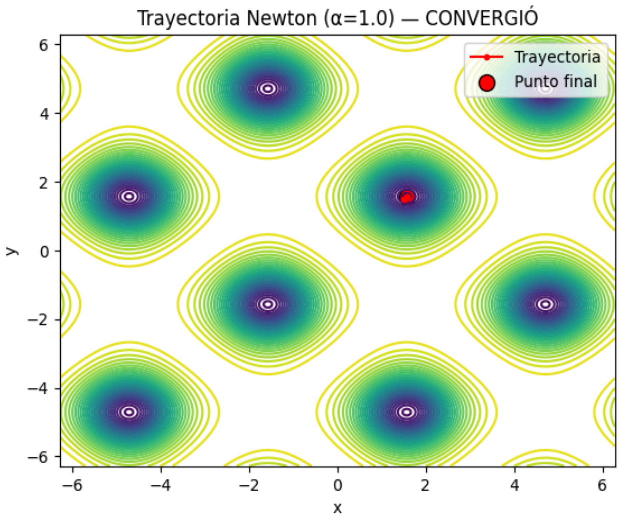


BFGS	(0.1,2)	adaptativo	(1.5707 ,1.5707)	-2.4255

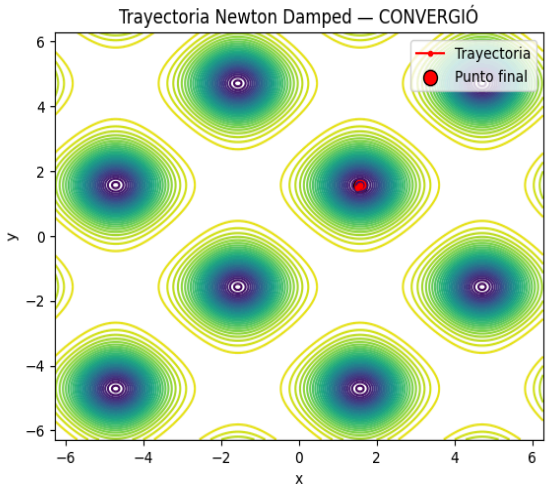


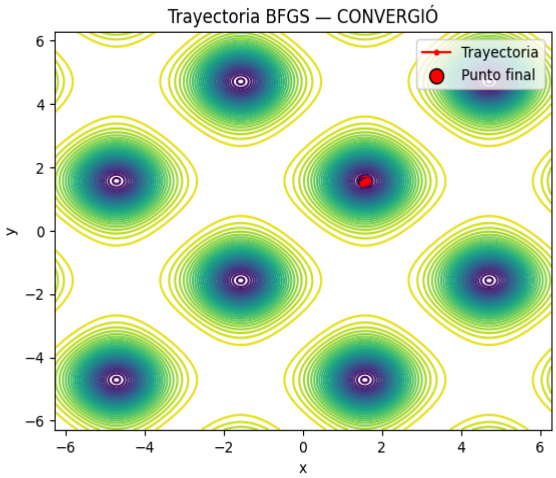
Newton	(1.5,1.5)	2.0 (fijo)	(4.68, 1.60)	0.0
Newton	(1.5,1.5)	1.0 (fijo)	(1.57, 1.57)	-2.4255



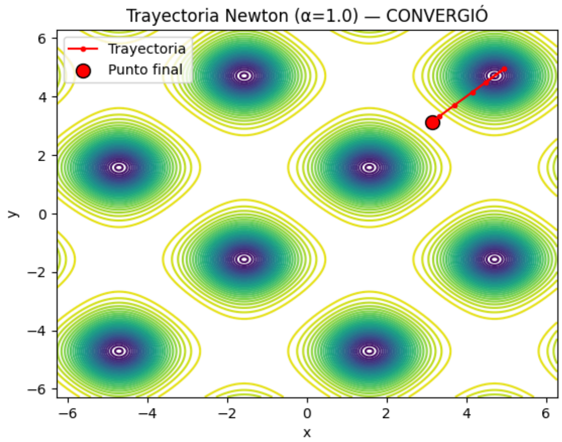


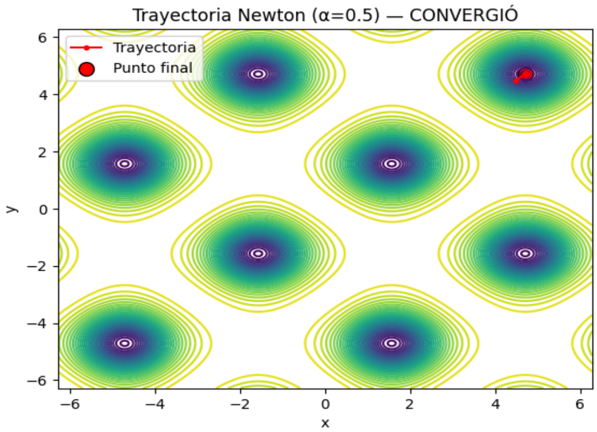
Newton damped	(1.5,1.5)	adaptativo	(1.57, 1.57)	-2.4255
BFGS	(1.5,1.5)	adaptativo	(1.57, 1.57)	-2.4255



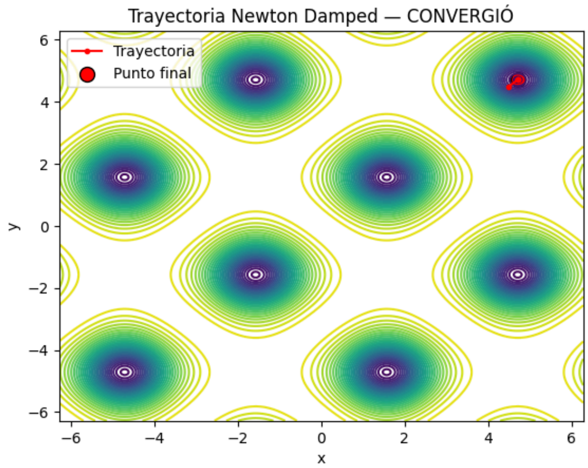


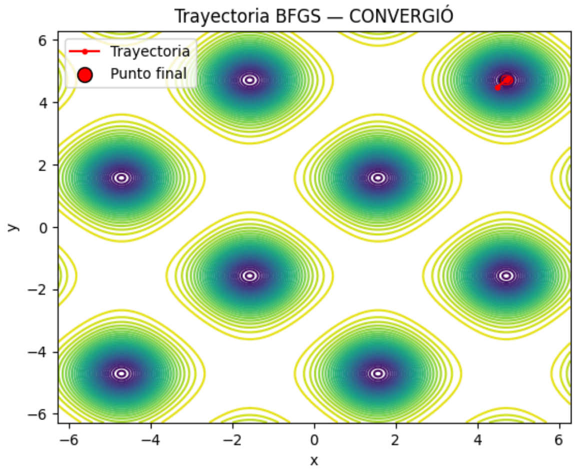
Newton	(4.5,4.5)	1.0 (fijo)	(3.14, 3.14)	0.0
Newton	(4.5,4.5)	0.5 (fijo)	(4.71, 4.71)	-2.4255



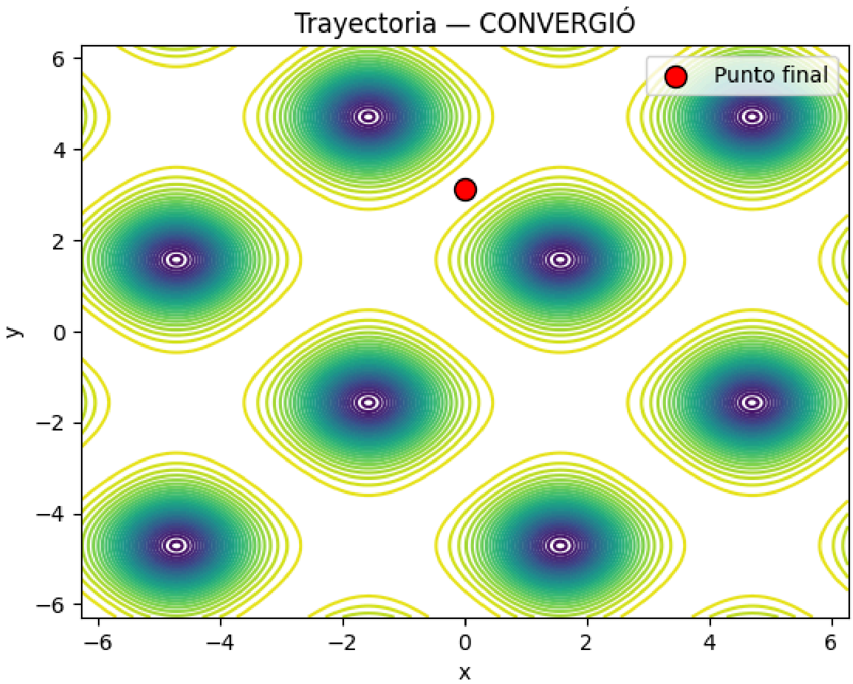


Newton damped	(4.5,4.5)	adaptativo	(4.71, 4.71)	-2.4255
BFGS	(4.5,4.5)	adaptativo	(4.71, 4.71)	-2.4255





Newton	(0,π)	1.0 (fijo)	(0, 3.14)	0.0
Newton damped	(0,π)	adaptativo	(0, 3.14)	0.0
BFGS	(0,π)	adaptativo	(0, 3.14)	0.0



Efecto del tamaño de paso (α):

En el método Newton Damped y en BFGS, la búsqueda lineal adaptativa ajusta el paso automáticamente, garantizando estabilidad sin necesidad de calibración manual

Para el método de Newton tradicional, se tiene que:

> **Un alfa grande** disminuye la cantidad de iteraciones pero pueden ocurrir saltos grandes que nos lleven a zonas problemáticas como en el caso de (1.5,1.5).
>
> **Un alfa pequeño** disminuye la posibilidad de ir a una zona crítica pero aumenta el número de iteraciones significativamente.

* Si no se controla el paso, Newton puede “saltar” a regiones donde tan(u) diverge (especialmente si sen(x) + sen(y) ≈ π). Por otro lado, un tamaño de paso muy pequeño aumenta las iteraciones y con ello la complejidad temporal.

Conclusiones:

El presente estudio permitió analizar el comportamiento de dos métodos de optimización no lineal (el método de Newton y el método Quasi-Newton (BFGS)) aplicados a la minimización de la función:

$$f(x, y) = -\tan^2(0.5sen(x) + 0.5sen(y))$$

Mediante el análisis teórico y la experimentación numérica, se verificó la existencia de infinitos mínimos globales, con un valor óptimo de ≈ −2.4255, de. También, se pudo verificar la no convexidad de la función a través del análisis de la matriz hessiana, la cual resultó semidefinida negativa en el origen.

Respecto al desempeño de los algoritmos, se observó que el método de Newton, si bien muestra una convergencia cuadrática en proximidades de un óptimo, presenta una marcada sensibilidad a la elección del tamaño de paso α y del punto inicial. En varios casos, con valores de α fijos grandes, el algoritmo converge a máximos locales no deseados. No obstante, la incorporación de una estrategia adaptativa para el tamaño de paso (Newton damped) ayuda a corregir este comportamiento, logrando convergencia al óptimo global en la mayoría de los casos.

Por su parte, el método BFGS demostró ser notablemente robusto y eficaz, alcanzando el óptimo global en casos donde Newton y Newton Damped fallaban, además BFGS superó a Newton especialmente en puntos iniciales alejados de la solución. Si bien es cierto que hay casos donde Newton nos brinda una convergencia cuadrática que es mejor que la superlineal de BFGS , el algoritmo BFGS encuentra un equilibrio favorable entre velocidad de convergencia y solidez.

Se destaca además, la importancia de la selección del punto inicial el cual puede condicionar el fallo de cualquiera de los tres métodos(Newton, Newton damped, BFGS) vistos en este artículo.

En fin, este trabajo refuerza la relevancia de adaptar el algoritmo de optimización a las particularidades de la función de estudio, además evidencia que en problemas de optimización no convexos (caracterizados por múltiples óptimos locales y regiones críticas) la selección del algoritmo no puede basarse únicamente en su velocidad de convergencia. Resulta necesario priorizar métodos que incorporen estrategias adaptativas de búsqueda lineal y garanticen direcciones de descenso robustas. La superioridad demostrada por BFGS bajo estas condiciones sugiere que, en entornos no convexos, la estabilidad numérica debe prevalecer sobre la convergencia acelerada cuando esta última compromete la confiabilidad del resultado.

Enlace al código de los algoritmos en github:
https://github.com/ronaldcbmtnz/tarea_optimizacion.