Ejercicio 1

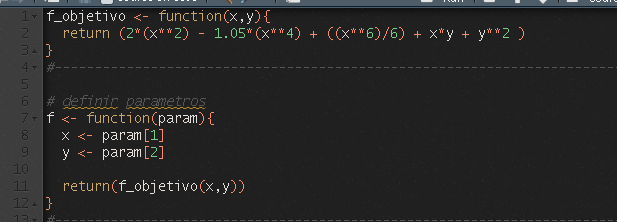
El método de Newton es altamente eficiente en funciones convexas, pero su comportamiento puede ser impredecible en funciones no convexas. Este ejercicio explora la sensibilidad del método al punto inicial y la naturaleza de los puntos críticos que encuentra. Consideremos la función “Three-Hump Camel”, una prueba estándar para algoritmos de optimización, definida como:

𝑓(𝑥,𝑦) = 2𝑥2 −1.05𝑥4 + 𝑥6 6 +𝑥𝑦 +𝑦2

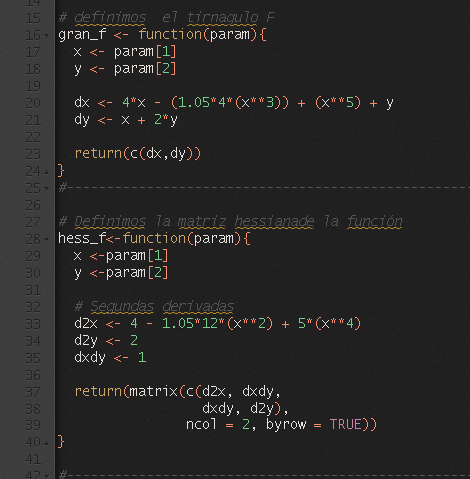
1. **Derivación analítica:**
   1. Calcule el vector gradiente ∇𝑓(𝑥,𝑦) y la matriz Hessiana H𝑓(𝑥,𝑦) de la función.
2. **Implementación:**
   1. implemente el método de Newton multivariado para encontrar los mínimos de esta función. El código debe permitir especificar un punto de partida inicial.
3. **Experimentación:**
   1. Ejecute su algoritmo desde los siguientes tres puntos iniciales:
      1. x0 =(1.5,1.0)
      2. x0 =(−1.5,−1.0)
      3. x0 =(0.1,−0.2), cerca de un máximo local
4. Para cada caso, informe el punto final al que converge el algoritmo, el valor de la función en ese punto y el número de iteraciones requeridas.
5. Genere un gráfico de contorno de la función y trace la trayectoria de optimización (los puntos x𝑘 en cada iteración) para cada uno de los tres puntos iniciales.
6. Discuta por qué diferentes puntos de partida conducen a diferentes resultados y analice el comportamiento del algoritmo cuando se inicia cerca de un punto no mínimo.

Soluciones:

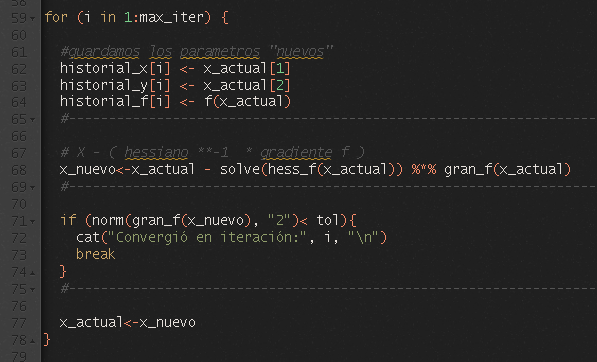
Función de manera de Código de lenguaje R

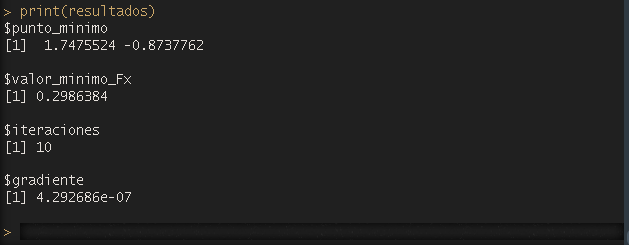


Gradiente de la función y se matriz Hessiana mediante código de lenguaje R

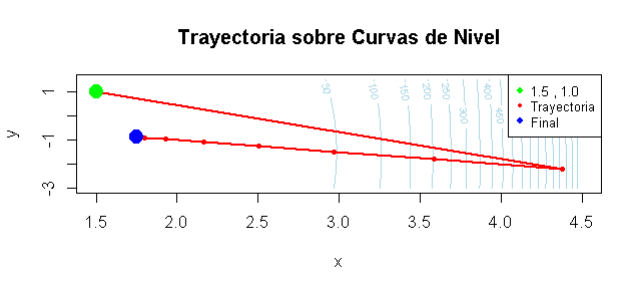


Cómo obtener el nuevo X mediante código en lenguaje R

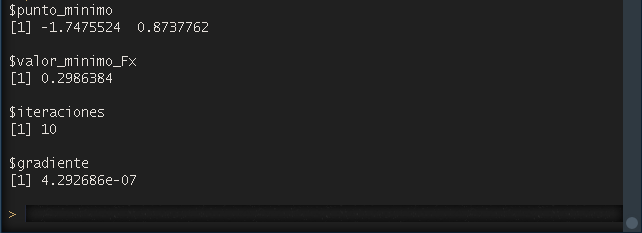
X inicial en el punto (1.5 , 1)



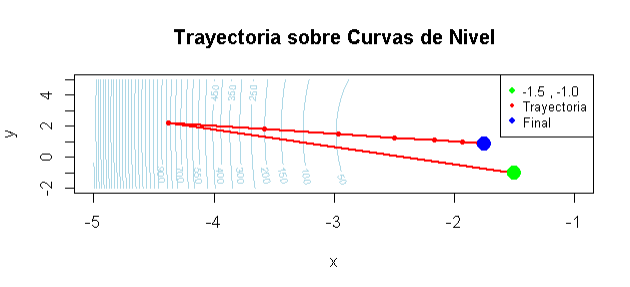
Gráfico



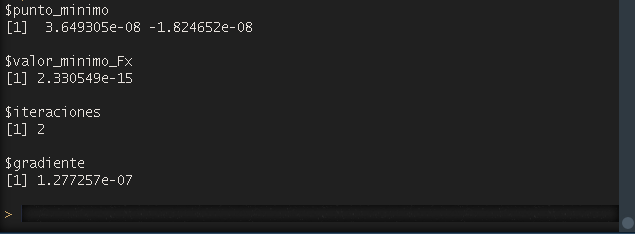
X inicial en punto (-1.5 , -1)



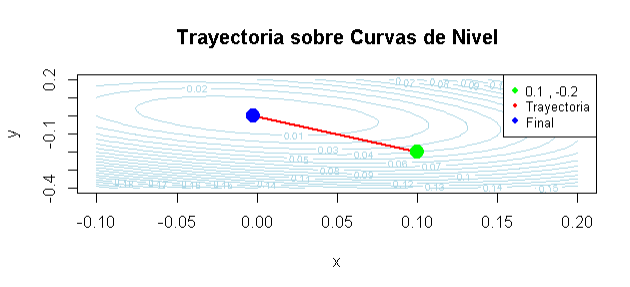
Gráfico



X inicial en el punto (0.1 , -0.2)



Gráfico



La razón de porqué distintos inicios generan distintos puntos mínimos es porque cada uno es mínimo pero puede ser unos locales y el otro global, y esta caída en un mínimo local y no en uno global se debe que esta función solo se acerca el mínimo más cercano de su punto inicial por eso también si elegimos un punto cerca del mini global se obtiene aquel mínimo, pero si elegimos un punto inicial lejano al mínimo global lo más probable es que llegue a un mínimo local. También hay que mencionar que entre más lejano esté el punto inicial de un mínimo ( ya sea local o global) más iteraciones le tomará llegar

Ejercicio 2

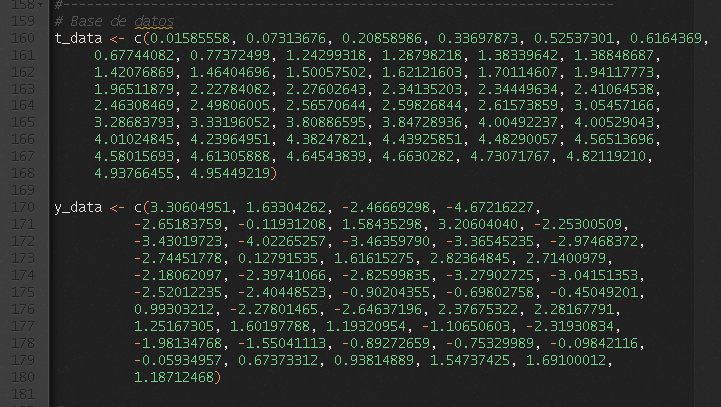
En ingeniería, es común necesitar ajustar un modelo teórico a datos experimentales. Este ejercicio consiste en usar el método de Newton para encontrar los parámetros de un modelo de oscilación 1 amortiguada, un problema fundamental en el procesamiento de señales y el control de sistemas. Se ha medido una señal que sigue el modelo:

𝑦(𝑡) = 𝐴𝑒−𝜆𝑡cos(𝜔𝑡 +𝜙)

El objetivo es encontrar el vector de parámetros 𝛽 = [𝐴,𝜆,𝜔,𝜙] que mejor se ajuste a un conjunto de mediciones (𝑡𝑖,𝑦𝑖). Para ello, se debe minimizar la suma de los errores al cuadrado (problema de mínimos cuadrados no lineales):

𝑆(𝛽) =𝑚 ∑ 𝑖=1 (𝑦𝑖 − 𝐴𝑒−𝜆𝑡𝑖 cos(𝜔𝑡𝑖 + 𝜙))2

1. **Base de datos dada**



1. .**Derivación para el método de Newton**: Minimizar 𝑆(𝛽) con el método de Newton requiere su gradiente y su Hessiana. En problemas de mínimos cuadrados, la Hessiana se puede aproximar eficientemente (aproximación de Gauss-Newton) como

H𝑆(𝛽) ≈ 2J⊤J,

donde J es la matriz Jacobiana del modelo.

* Derive las expresiones para el vector de residuos:

𝑟𝑖(𝛽) = 𝑦𝑖 −𝑦(𝑡𝑖,𝛽)

* Derive las expresiones para la matriz Jacobiana:

J𝑖𝑗 = 𝜕𝑟𝑖 / 𝜕𝛽𝑗

* Implemente funciones en R o Python que calculen el gradiente ∇𝑆(𝛽) = −2J⊤𝑟 y la Hessiana aproximada H𝑆(𝛽).

1. **Implementación y ejecución:**

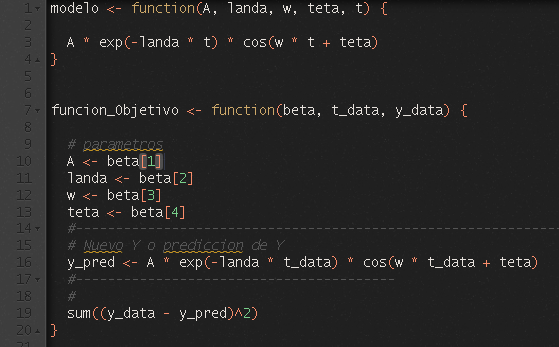
* Adapte su código del método de Newton para usar el gradiente y la Hessiana aproximada que acaba de derivar.
* Inicie la optimización desde un punto razonable pero no exacto (por ejemplo, 𝛽actual = [4.5, 0.3, 6.0, 0.5]).
* Ejecute el algoritmo para encontrar los parámetros óptimos 𝛽estimado.

1. **Análisis y visualización:**

* Imprima los parámetros estimados y compárelos con los valores verdaderos, para este ejercicio, los valores verdaderos de los parámetros son: 𝐴 = 5, 𝜆 = 0.2, 𝜔 = 2𝜋 y 𝜙 =𝜋/4.
* Cree un gráfico que muestre
  + Los datos ruidosos originales (como un diagrama de dispersión).
  + La curva del modelo con los parámetros verdaderos
  + La curva del modelo ajustado con los parámetros estimados por su algoritmo.
* Discuta la precisión del ajuste y la eficacia del método para este problema de ingeniería.

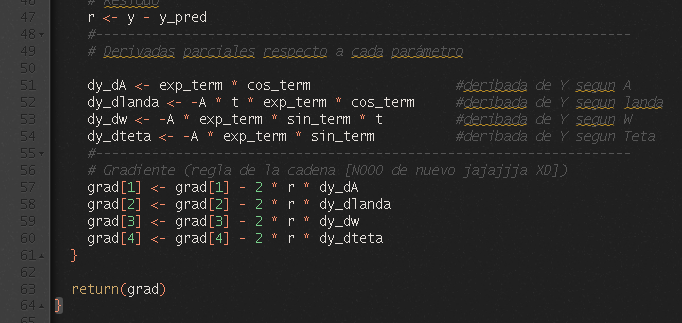
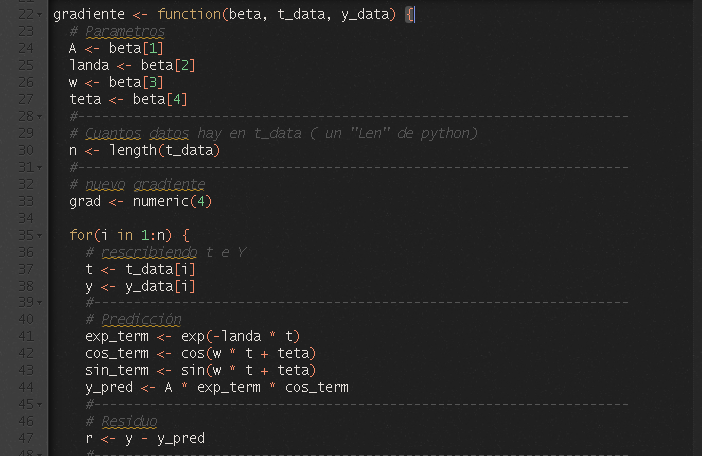
Solución:

Función objetivo en lenguaje de R

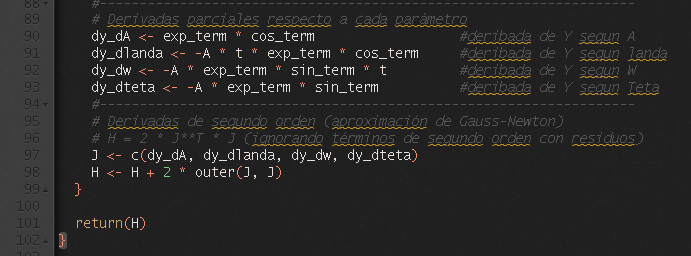
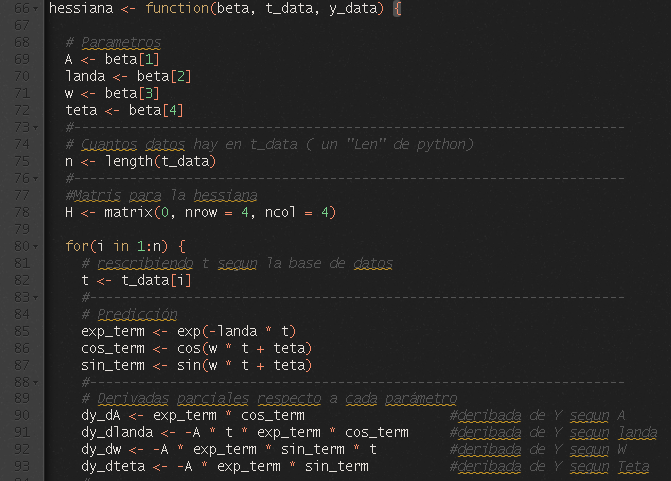


## Sin Jacobiano

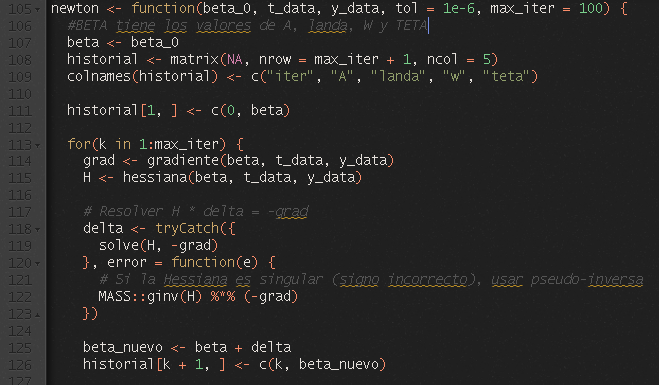
Código de Gradiente de la función



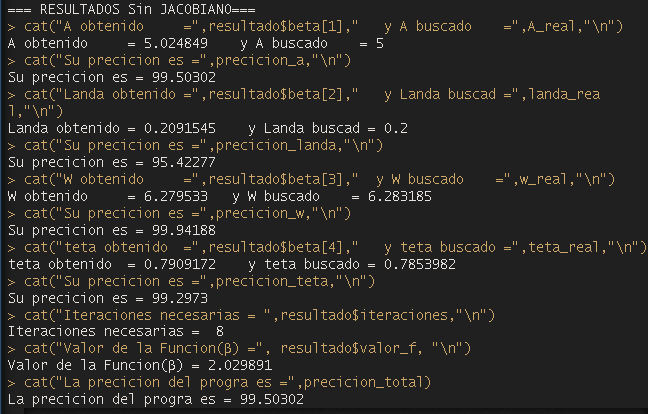
Código de la Matriz Hessiana normal



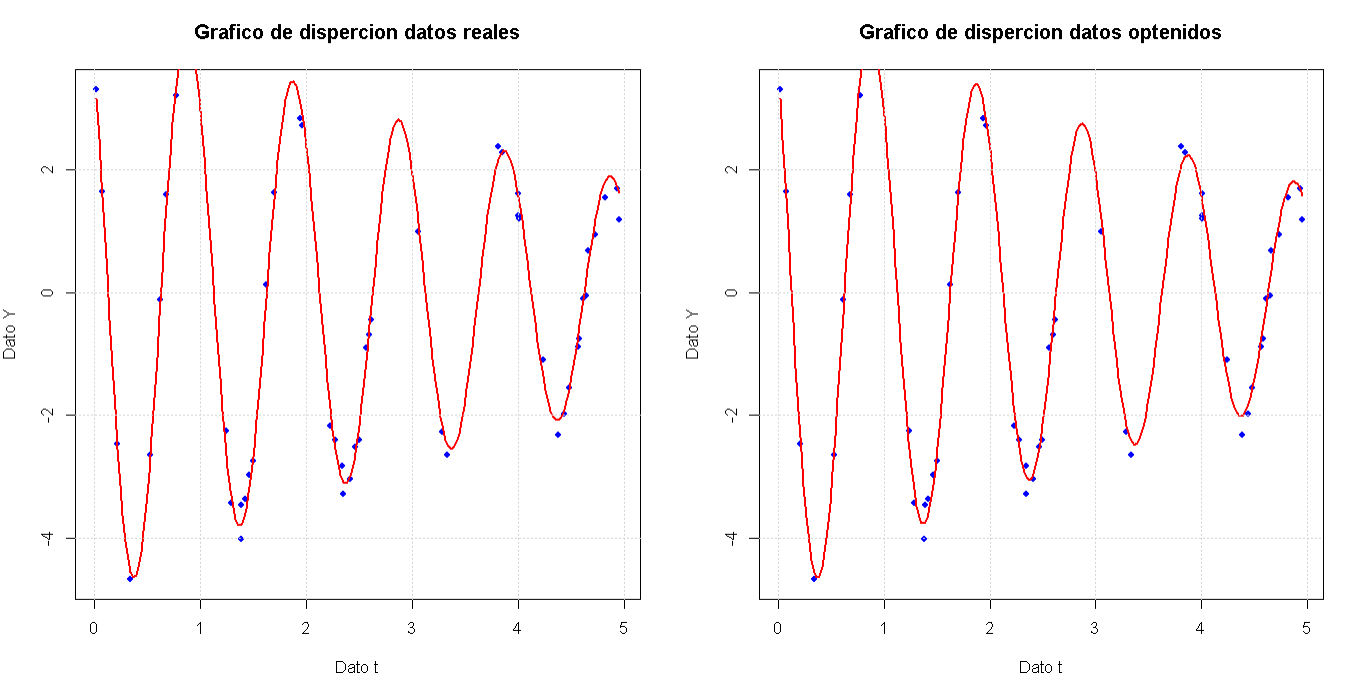
Código de Newton para calcular los nuevos valores de A, Lambda, Omega y Phi

****

Datos de los datos obtenidos sin la Matriz Jacobiana

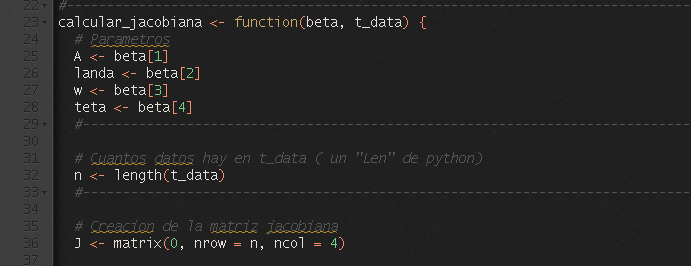


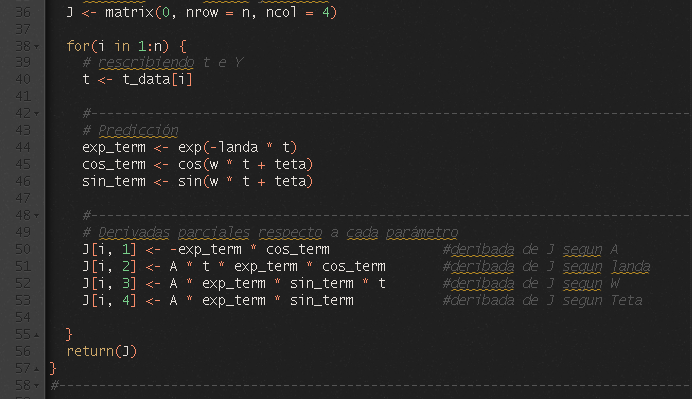
Gráficos de los resultados sin Jacobiano



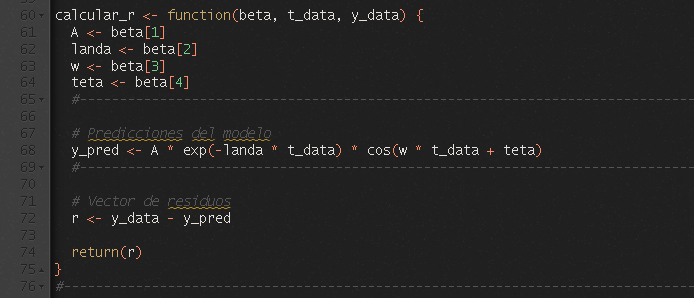
## Con Jacobiano

Calculo de Jacobiano

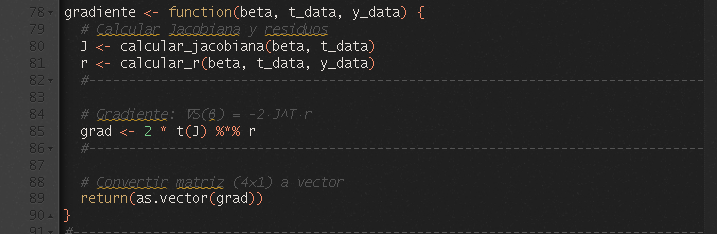




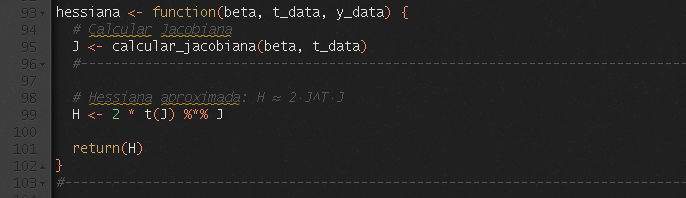
Cálculo de Residuo ( r )



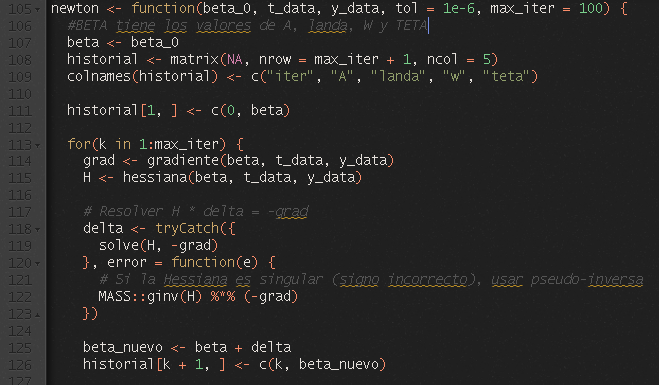
Gradiente de la función con Jacobiano



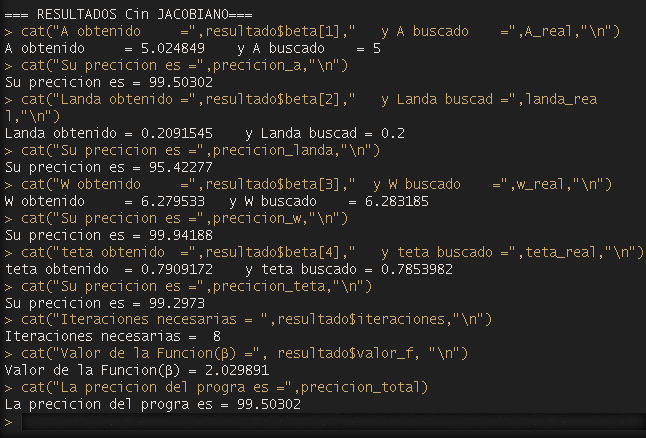
Código de la matriz Hessiana con Jacobiano



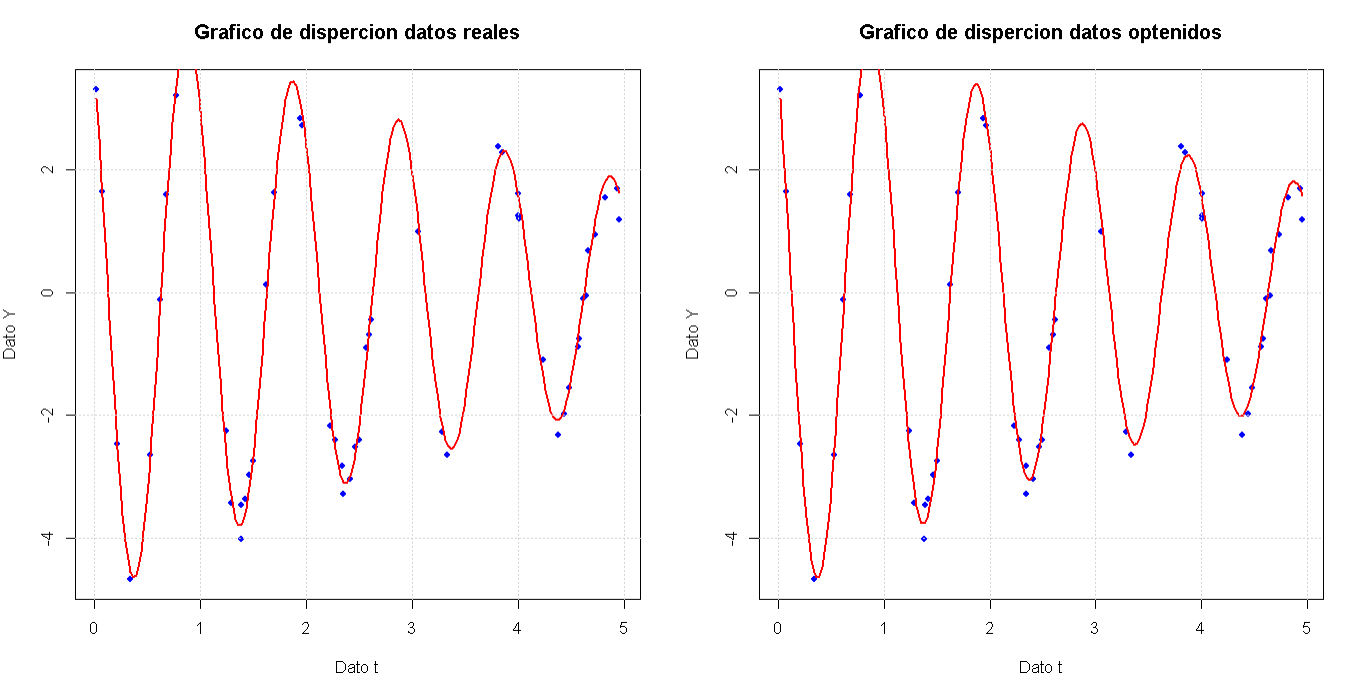
Código de Newton para calcular los nuevos valores de A, Lambda, Omega y Phi

****

Datos de los datos obtenidos Con la Matriz Jacobiana



Gráficos de los datos con Matriz Jacobiano



Por alguna razón los resultados obtenidos a través de la Hessiana sin Jacobiano y la Hessiana con Jacobiano dan los mismos resultados para todos los valores.

|  | Sin Jacobiano | Con Jacobiano | Teórico |
| --- | --- | --- | --- |
| A | 5.024849 | 5.024849 | 5 |
| Landa | 0.2091545 | 0.2091545 | 0.2 |
| Omega | 6.279533 | 6.279533 | 6.283185 |
| Phi | 0.7909172 | 0.7909172 | 0.7853982 |
| Iteraciones | 8 | 8 | —---------------- |
| Valor de la función | 2.029891 | 2.029891 | —---------------- |

Con esta tabla se puede observar que los datos obtenidos son muy similares a los datos obtenidos.

La precisión obtenida en el programa es de 99.50302%

Como la precisión es alta se puede decir que su eficiencia también es alta

Ejercicio 3

La principal ventaja de los métodos Quasi-Newton es que evitan el cálculo explícito de la Hessiana, lo que los hace ideales para funciones complejas. En este ejercicio, se utilizará el algoritmo BFGS para explorar la superficie de una función con múltiples mínimos y analizar cómo el punto de partida determina el resultado.

Utilice la función de Himmelblau, una prueba clásica para optimización que tiene cuatro mínimos locales idénticos:|

𝑓(𝑥,𝑦) = (𝑥^2 +𝑦 −11)2 +(𝑥+𝑦^2 −7)^2

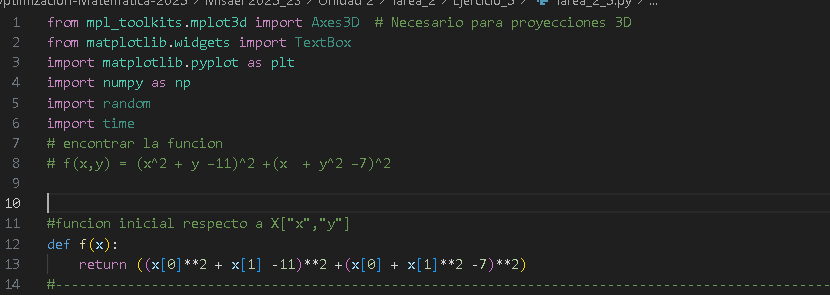
El valor del mínimo de la función es 𝑓(𝑥,𝑦) = 0. Actividades:

1. 1. Componentes del algoritmo:
   1. Calcule analíticamente el gradiente ∇𝑓(𝑥,𝑦). En este ejercicio, asumimos que la Hessiana es demasiado compleja o costosa de calcular, justificando el uso de BFGS.
   2. Implemente el algoritmo Quasi-Newton con la actualización BFGS y la búsqueda de paso por retroceso (backtracking).
2. Exploración de los mínimos:
   1. Investigue y encuentre las coordenadas de los cuatro mínimos de la función de Himmelblau.
   2. Ejecute su algoritmo BFGS partiendo de los siguientes tres puntos iniciales:
      1. x0 =(0,0)
      2. x0 =(−1,3)
      3. x0 =(4,−2)

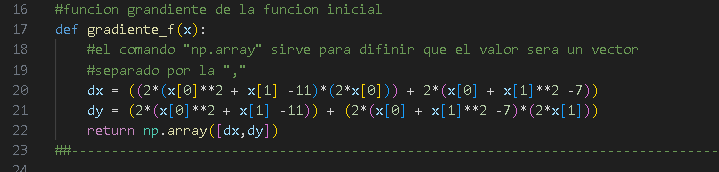
1. Análisis y visualización:
   1. Para cada punto de partida, informe a cuál de los cuatro mínimos converge el algoritmo y cuántas iteraciones fueron necesarias.
   2. Genere un único gráfico de contorno de la función de Himmelblau.
   3. En el gráfico, marque la ubicación de los cuatro mínimos teóricos.
   4. Superponga las trayectorias de optimización de sus tres ejecuciones.
   5. Discuta cómo la topografía de la función (sus “valles” y “cuencas de atracción”) guía al algoritmo hacia diferentes soluciones dependiendo de dónde comience.

Solución :

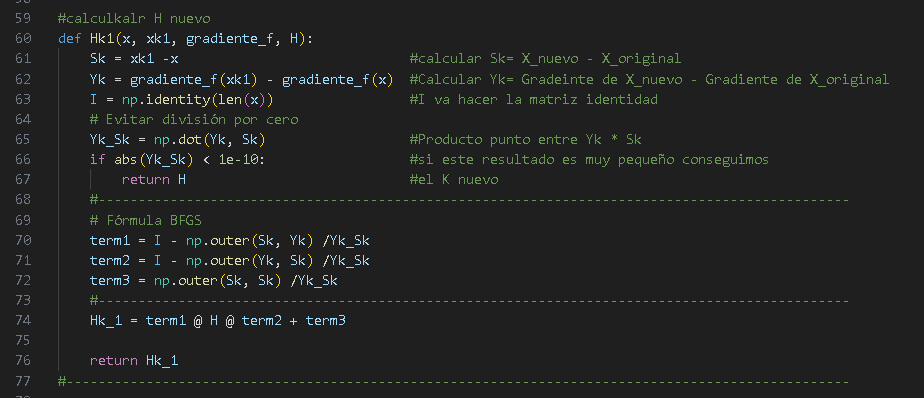
Función de manera código en lenguaje Python



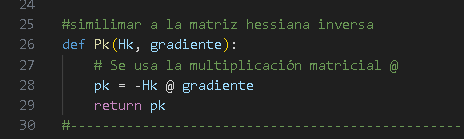
Gradiente



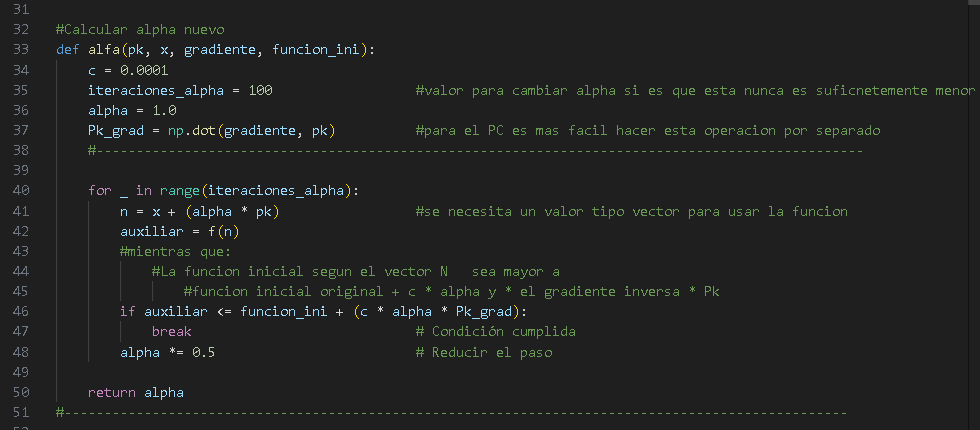
Codigo hessiana



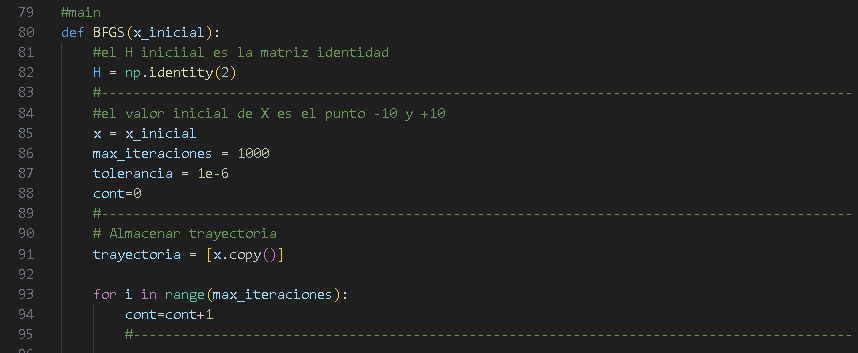
Codigo Hessiana inversa

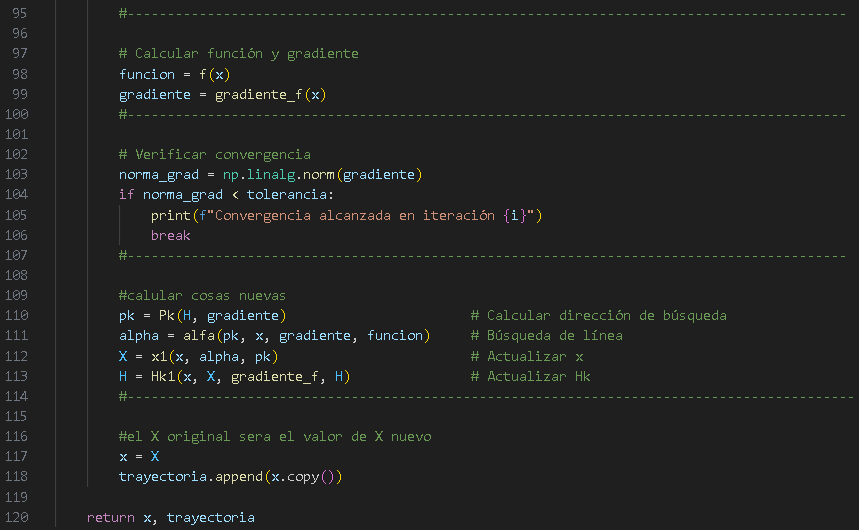


Codigo para calcular Alpha

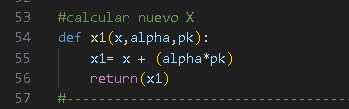


Codigo de BFGS

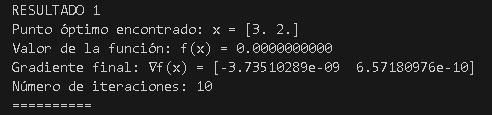




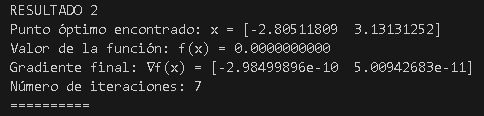
Codigo para calcular nuevo X



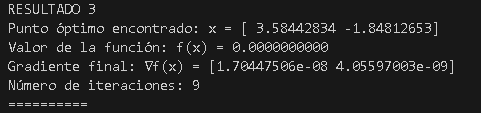
Valores obtenidos del primer punto inicial



Valores obtenidos del segundo punto inicial



Valores obtenidos del tercer punto inicial



Valores obtenidos del cuarto punto inicial

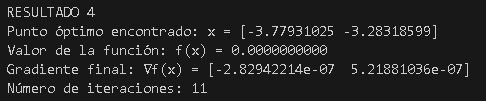
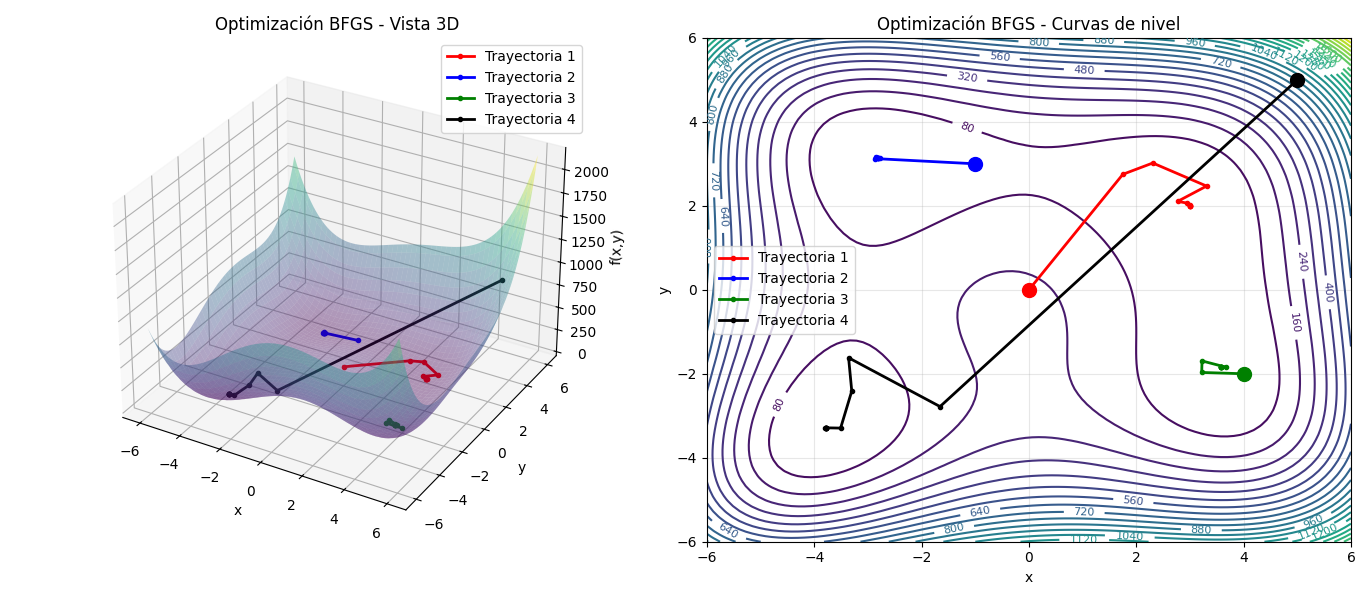


Gráfico de los puntos iniciales y sus valores



La razón de porque los puntos tienden a sus respectivos mínimos indicados es porque y en especial el punto 4, es porque la función BFGS no busca el mínimo más cercano si el mínimo más estable a través de la predicción de la hessiana y su gradiente, es decir que a través de matriz hessiana estimada se avanza en dirección del gradiente una distancia determinada.

Ejercicio 4

Cuando la matriz Hessiana no es positiva definida, la dirección de búsqueda del método de Newton puede apuntar hacia un máximo o un punto silla, causando que el algoritmo falle. Una solución práctica es la regularización.

Considere la función:

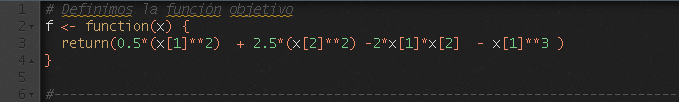
𝑓(𝑥,𝑦) = 0.5\*𝑥^2 +2.5\*𝑦^2 −2\*𝑥\*𝑦 −𝑥^3

Esta función tiene un mínimo local y un punto silla, lo que la hace interesante para probar la robustez. Actividades:

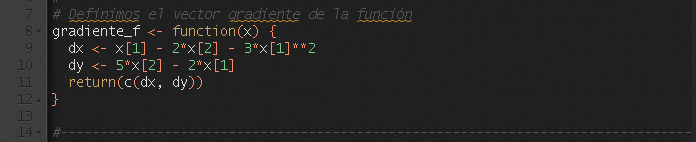
1. **Análisis de la Hessiana:**
   1. Calcule la matriz Hessiana H𝑓 (𝑥,𝑦).
   2. Evalúe la Hessiana en el punto x0 = (1.5,0.5). Calcule sus valores propios (eigenvalues). ¿Es la matriz positiva definida en este punto?
2. **Implementación estándar:**
   1. Ejecuta el método de Newton estándar comenzando desde x0 =(1.5,0.5). Describa lo que sucede. ¿Converge el algoritmo a un mínimo?
3. **Implementación regularizada**
   1. Modifique su algoritmo de Newton para incorporar la técnica de regularización descrita en clases. Específicamente, antes de calcular el paso de actualización, verifique si algún valor propio de la Hessiana es menor o igual a cero. Si es así, modifique la Hessiana sumándole una matriz identidad multiplicada por una constante: Modificada = H + C\*I, donde C es un valor pequeño pero suficiente para hacerla positiva definida.
4. **Comparación:**
   1. Ejecuta el algoritmo regularizado desde el mismo punto de partida x0 = (1.5,0.5).
   2. Visualice en un gráfico de contorno las trayectorias del método estándar (si es posible) y del método regularizado.
   3. Explique por qué la regularización permite al algoritmo converger correctamente hacia un mínimo, mientras que la versión estándar falla o se comporta de manera errática.

Solución :

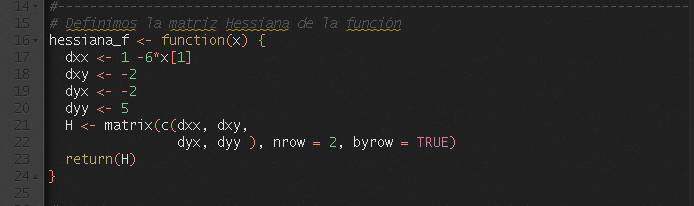
Función en forma de código en lenguaje R



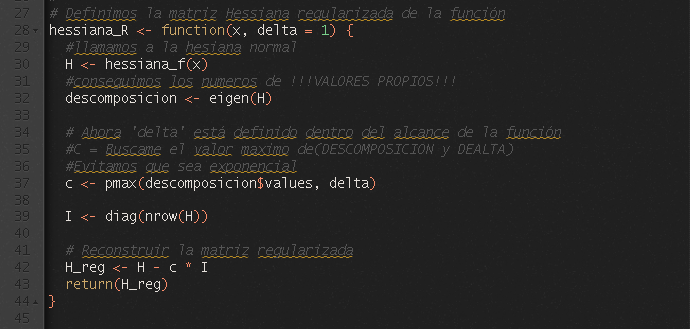
Gradiente



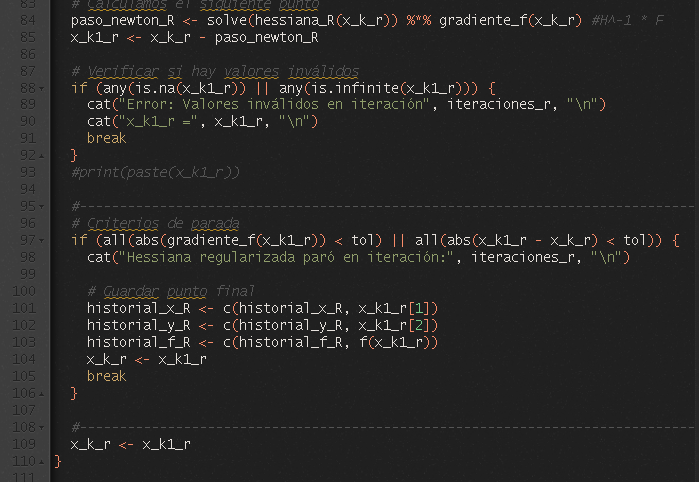
Hessiana normal



Hessiana regularizada



Cálculo de nuevo X



Datos obtenidos

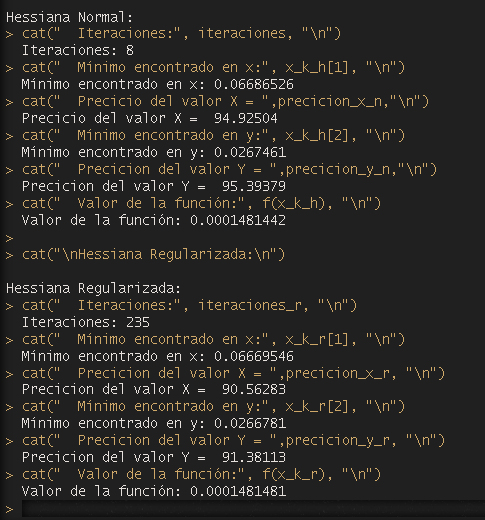
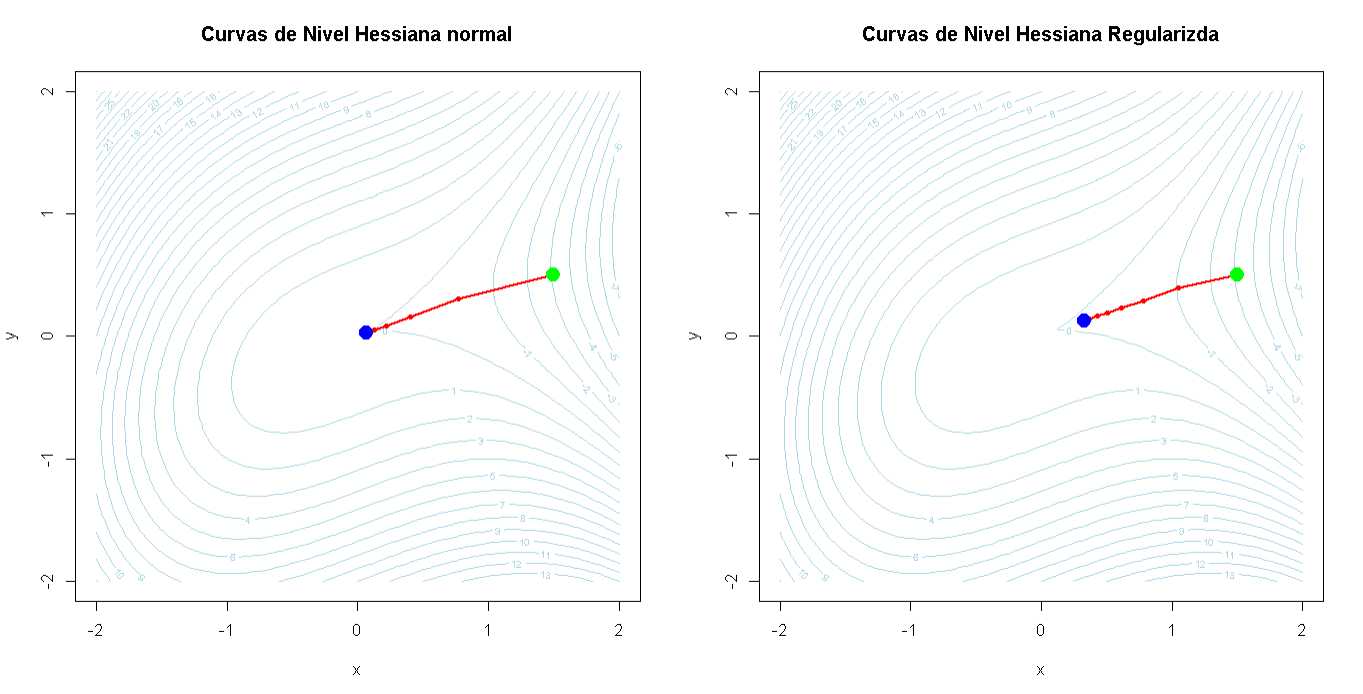


Tabla de diferencia entre Hessiana Normal y Regularizada

|  | Hessiana normal (N) | Hessiana regularizada ® | Diferencia (R-N) |
| --- | --- | --- | --- |
| Iteraciones | 8 | 235 | 227 |
| Punto X | 0,06686526 | 0,06669546 | -0,0001698 |
| Punto Y | 0,0267461 | 0,0266781 | -0,000068 |
| F(X,Y) | 0,0001481442 | 0,0001481481 | 0,0000000039 |

Gráfico



En ambos casos con hessiana normal y regularizada se obtiene datos muy similares a tal grado que su precisión son las misma y sus puntos mínimos encontrados solo se diferencian por un error del 1e-4 decimales entre ambos métodos con una diferencia significativa el número de iteraciones donde para este programa el método más eficiente vendría siendo el de la hessiana normal