OPTIMIZACIÓN II
Práctica 5

Francisco J. Aragón Artacho
Dpto. Matemáticas, Universidad de Alicante

31 de marzo – 1 de abril de 2022

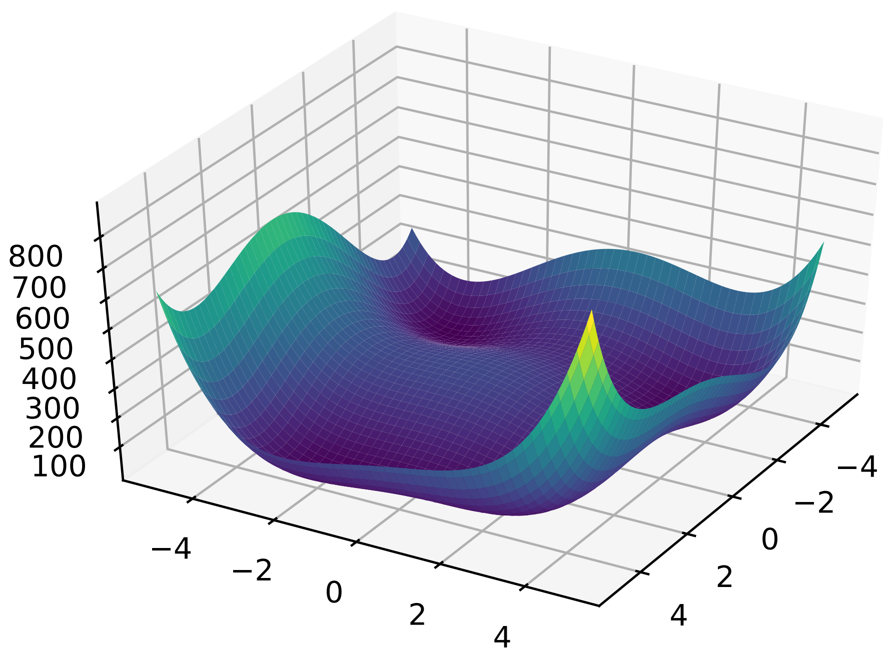
En esta práctica programaremos el algoritmo sin derivadas de Nelder y Mead. Para testar el método, utilizaremos la función de Himmelblau[1] $f : \mathbb{R}^2 \to \mathbb{R}$ *definida por*

$$f(x, y) = \left(x^2 + y - 11\right)^2 + \left(x + y^2 - 7\right)^2.$$

Esta función tiene un máximo local en $(-0.270845, -0.923039)^T$ *con valor 181.617 y cuatro mínimos globales en*

$$(3, 2)^T, \quad (-2.805118, 3.131312)^T, \quad (-3.779310, -3.283186)^T, \quad (3.584428, -1.848126)^T,$$

donde la función vale 0.



[1]Recibe su nombre del ingeniero químico David M. Himmelblau.

Crea un programa en Python llamado `Pr6.py` donde se realice lo siguiente:

1. Se defina la función f.
2. Crea una función llamada `NM` que ejecute el algoritmo de Nelder y Mead para cualquier función $f : \mathbb{R}^n \to \mathbb{R}$:
   - Los parámetros de entrada de la función deben ser: la función `f`, el punto inicial `x0` y el número de iteraciones `it`.
   - Crea el símplex inicial a partir del punto `x0` usando los vectores de la base canónica: tendrá vértices en `x0`, `x0+`$(1,0,\ldots,0)^T$, `x0+`$(0,1,0,\ldots,0)^T$,..., `x0+`$(0,0,\ldots,0,1)^T$. Almacena estos puntos en una matriz `X` de tamaño $n \times (n+1)$.
   - Calcula el valor de la función f en cada punto del símplex y almacena el resultado en un vector llamado `fX`, utilizando el mismo orden que en `X`. Utiliza la función `np.argsort` para obtener los índices que ordenan ascendentemente el vector `fX`. Guarda el vector en la variable `orden`.
   - Por tanto, tendremos que $x_{\min} =$ `X[:,orden[0]]` y $f(x_{\min}) =$ `fX[orden[0]]`, mientras que $x_{\max} =$ `X[:,orden[n]]` y $f(x_{\max}) =$ `fX[orden[n]]`. Define `xb` como el baricentro entre los puntos que no son $x_{\max}$ (ver apuntes).
   - Ahora ya tenemos todos los ingredientes necesarios para escribir el Algoritmo 3 de los apuntes. Crea un bucle donde se ejecuten el número de iteraciones dado por `it`. En cada iteración, se reemplazará $x_{\text{máx}}$ por x_{new} siempre que $f(x_{\text{new}}) \leq f(x_{\max})$ (en caso contrario, se contraerá el símplex hacia el vértice $x_{\text{mín}}$). Actualiza solamente los valores de `fX` que sean necesarios (no vuelvas a calcularlos todos, pues esto ralentizaría el algoritmo).
   - Crea una matriz 3D de tamaño $n \times (n+1) \times$ `it` donde se guarden los valores de `X` en todas las iteraciones: en `saveX[:,:,k]` se almacenarán las coordenadas del símplex `X` correspondiente a la iteración `k`.
   - Tras ejecutar `iter` iteraciones, la función debe dar como valores de salida $x_{\min}$, $f(x_{\min})$ y `saveX`.
3. Aplica el algoritmo `NM` a la función f de Himmelblau, con `it=80 y x0=`$(-0.3, 3.7)^T$ y guarda el resultado en `sol`. Comprueba que converge al mínimo global $(3,2)^T$.
4. Muestra 50 curvas de nivel de la función f de Himmelblau (dibujadas sobre $[-5,5] \times [-5,5]$). Dibuja los triángulos obtenidos por la función `NM` en el apartado anterior. Usando el comando adecuado, muestra ambos ejes usando la misma escala (quedando así un dibujo cuadrado).
5. (OPCIONAL) Aplica `NM` a la generalización de la función de Rosenbrock

$$h(x_1,x_2,x_3) := 100\left(x_2 - x_1^2\right)^2 + 100\left(x_3 - x_2^2\right)^2 + (1-x_1)^2 + (1-x_2)^2,$$

   con `iter=150` y `x0=`$(-4,0,4)^T$. ¿Converge al mínimo global de h? Dibuja en una nueva figura tridimensional los símplex obtenidos durante todas las iteraciones, usando de manera adecuada el método `plot3D`.