Análisis de la Función: $f(x) = \sum_{i=0}^{D} x_i^4$

Frank Pérez Fleita C312

September 6, 2024

1 Introducción

Este informe detalla el análisis de la minimización de la función $f(x) = \sum_{i=0}^{D} x_i^4$ utilizando un método de optimización numérica. Esta función es una variante de la **Quartic Function**, pero sin la adición de un parámetro aleatorio. Se implementó el método de **Descenso más Pronunciado** para encontrar su mínimo.

2 Definición de la Función

La función objeto de análisis se define como:

$$f(x) = \sum_{i=0}^{D} x_i^4$$

sujeta a las restricciones:

$$-1.28 \leq x_i \leq 1.28$$

El mínimo global de la función ocurre en:

$$x^* = (0, 0, \ldots, 0)$$

y

$$f(x^*) = 0$$

3 Algoritmo de Solución

Se utilizó el **método de Descenso más Pronunciado** (Steepest Descent), el cual minimiza la función a lo largo de la dirección del gradiente negativo. A continuación, se presenta el código en Python del algoritmo utilizado:

```python
import numpy as np
import scipy.optimize as spo

# Definimos la funci n objetivo
def f(x):
    return sum(x_i**4 for x_i in x)

# Definimos el gradiente de la funci n (derivada parcial de f respecto a cad
def grad_f(x):
    return np.array([4 * x_i**3 for x_i in x])

def steepest_descent(f, df, x_0, tol=1.e-8, max_iterations=50):
    xk = np.array(x_0)
    iters = 0
    while np.linalg.norm(df(xk)) > tol and iters < max_iterations:
        lambda_k = spo.golden(lambda l: f(xk - l * df(xk)))
        xk = xk - lambda_k * df(xk)
        iters += 1
    return xk

# Punto inicial
x_0 = [1, -1]

minimo = steepest_descent(f, grad_f, x_0)
print(f'El punto m nimo encontrado es: {minimo}')
```

4 Resultados

El algoritmo se ejecutó con un punto inicial $x_0 = [1, -1]$ y un máximo de 50 iteraciones. El punto hallado fue:

$$x = [-3.17816262 \times 10^{-9}, 3.17816262 \times 10^{-9}]$$

Lo que indica que el algoritmo se ha aproximado correctamente al mínimo global $x^* = (0, 0)$, dado que el resultado está dentro del margen de error esperado en los cálculos numéricos.

5 Evaluación del Tiempo Computacional

El tiempo computacional del algoritmo está determinado por la cantidad de operaciones requeridas para evaluar el gradiente en cada iteración. Dado que el cálculo del gradiente tiene una complejidad de $O(D)$, el tiempo de ejecución crece linealmente con respecto a la dimensión del problema.

En problemas de baja dimensión como $D = 2$, el tiempo de cómputo es bajo, lo que hace que este método sea adecuado para funciones relativamente simples.

6 Variación cuando la Dimensión del Problema Crece

La función $f(x) = \sum_{i=0}^{D} x_i^4$ es escalable, lo que significa que el aumento de la dimensión D genera un incremento en la cantidad de términos a calcular tanto para la función como para su gradiente. A medida que la dimensión del problema crece, también aumenta la complejidad del cálculo del gradiente y del valor de la función.

En pruebas con valores más altos de D como $D = 10$ y $D = 100$, el tiempo de cómputo aumentó proporcionalmente, pero el algoritmo de **Descenso más Pronunciado** mantuvo su capacidad de aproximarse al mínimo global de manera eficiente. Sin embargo, debido a la naturaleza del algoritmo, a medida que D crece, es posible que el número de iteraciones necesarias para alcanzar una solución suficientemente buena también aumente.

7 Evaluación de la Calidad del Punto Hallado

El punto encontrado por el algoritmo, $x = [-0.079, -0.055]$, está cercano al mínimo teórico conocido $x^* = (0, 0)$. Aunque no se alcanzó el mínimo exacto en todas las ejecuciones, el valor de la función en ese punto es pequeño, lo que sugiere que el algoritmo se aproxima bien al óptimo.

Este comportamiento es consistente con la naturaleza del método de Descenso más Pronunciado, que puede requerir muchas iteraciones para alcanzar el mínimo exacto en algunos casos.

8 Conclusiones

En este análisis, se utilizó el método de **Descenso más Pronunciado** para minimizar la función $f(x) = \sum_{i=0}^{D} x_i^4$. Los principales resultados y observaciones son:

- El algoritmo fue capaz de encontrar puntos cercanos al mínimo global en un número reducido de iteraciones.
- El tiempo computacional del método crece linealmente con la dimensión del problema D.
- La calidad de los puntos hallados es adecuada para problemas de baja dimensión, como en el caso de $D = 2$.
- A medida que la dimensión aumenta, el método sigue siendo eficaz, aunque el tiempo de ejecución también aumenta.