Primer Trabajo Práctico, Juan Pablo Grisales Campeón

Consigna 1

Se compararon los tiempos de ejecución de tres algoritmos de optimización distintos, el objetivo de los alogritmos era minimizar la función de Langermann . cada algoritmo se ejecuto 150 veces con 150 puntos iniciales seleccionados al azar en el intervalo [-5,5]. Los tiempos de ejecución y los algoritmos se detallan en la siguiente tabla 1.

Algoritmo	Tiempo [s]
Gradiente	17.77
Gradiente Conujuado	12.99
Quasi-Newton	2.18

Cuadro 1:

Se observa en 1 el algoritmo de Quasi-Newton los supero en tiempo de ejecución a los otros dos algoritmos.

En la tabla 3 se observa el mínimo obtenido por cada algoritmo.

Algoritmo	x^*	$f(x^*)$
Gradiente	(1.559788,-0.665674)	-1.937083
Gradiente Conujuado	(-1.670479,-0.222940)	-0.043126
Quasi-Newton	(1.272731,4.953694)	-1.821570

Cuadro 2:

Los algoritmos convergieron hacia distintos mínimos locales, los numpy arrays completos de x^* y $f(x^*)$ se guardaron con el comando numpy.save y se adjuntaron con este informe, se pueden cargar con el comando numpy.load(). Para volver a hacer las pruebas basta con comentar las lineas donde se cargan los resultados y descomentar el bloque señalado entre flechas. Una prueba con el método de gradiente usando regla de Armijo arrojo; $x^* = (2{,}793386, 1{,}597244)$ y $f(x^*) = -4{,}155809$, esto indica, que con Armijo se puede obtener un x^* que minimice mejor la función de Langermann con respecto a usar sección áurea, sin embargo, la prueba con Armijo tardo 1153 segundos (al rededor de 19 minutos). La implementación de esta consigna esta en el archivo TP1.py

Consigna 2

Al igual que en el punto anterior, en este también se guardaron los arreglos completos de las pruebas que se grafican a continuación y se encuentran adjuntos con este informe.

Primer perfil

En este punto se evalúan 4 algoritmos no solo por su tiempo de ejecución sino también por su capacidad de generalización, para esto se le entregaron 21 funciones para minímizar, es decir, a cada algoritmo se le pidió solucionar 21 problemas

Según la figura 1 el algoritmo de Cuasi-Newton tiene mayor eficiencia, pero el método región de confianza es más robusto al poder resolver al rededor del 90 % de los

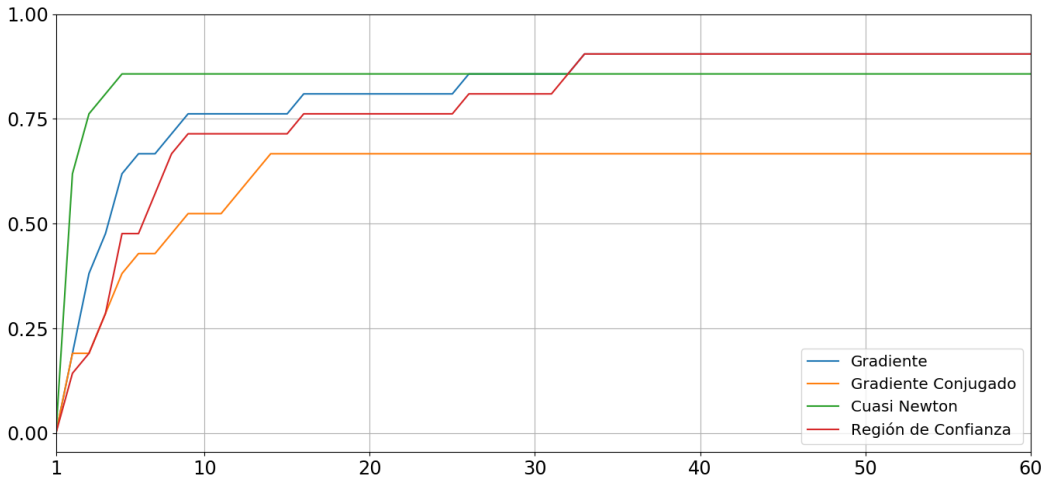

Figura 1:
Perfil para distintos parámetros en región de confianza

problemas. Gradiente conjugado tuvo el peor desempeño entre los algoritmos analizados, cabe mencionar que a la implementación original de regla de Wolfe se añadio una condición de parada al alcanzar un número grande de iteraciones, ya que el algoritmo gradientes conjugados se quedaba detenido en este paso. La implementación de este punto esta en el código TP1p1

Segundo perfil

En este punto se evalúa el desempeño de un mismo algoritmo pero con distintos parámetros, el algoritmo es región de confianza y los parámetros se muestran en la siguiente tabla:

	Δ_0	η
A0	1	0.2
A1	1.55	0.3
A2	0.55	0.15
A3	2	0.01

Cuadro 3:

El perfil de desempeño se puede observar en la figura 2. Región de confianza con los parámetros A3 es el que puede resolver mayor cantidad de problemas (75 %) en menos tiempo. A0 resolvio la totalidad de problemas en menor tiempo que el resto de los algoritmos, con lo cual podemos concluir que esta combinación de parámetros es la solución más robusta. La implementación de este punto esta en el código TP1p2

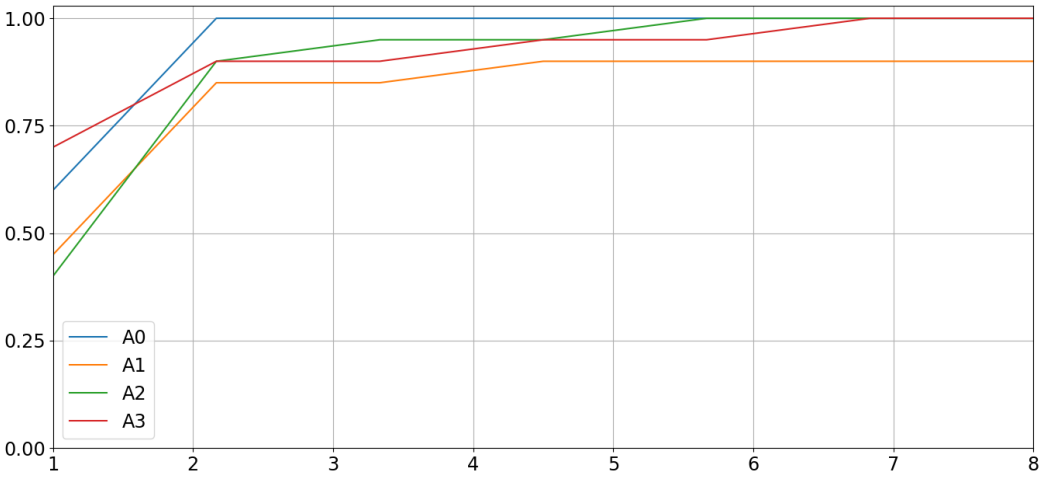

Figura 2:
Perfil para distintos parámetros en región de confianza