Tarea 3: Enjambre de Partículas

Fernando Carrillo A01194204

07/09/20

Descripción del problema:

Utilizando el algoritmo de enjambre de partículas se van a buscar los coeficientes de la función: $a_1 x^2 - a_2 x^{\frac{(e^{-a_3 * x})}{2}}$, tales que se ajusten a las siguientes evaluaciones para los distintos valores de 'x':

x	**f(x)**
0	-3.271085
1	2.994633
2	14.999975
3	34.999999
4	62.999999
5	98.999999
6	143
7	195
8	255
9	323

Descripción de la representación de los individuos:

Para este problema se van a representar las soluciones como vectores de 3 dimensiones, donde cada elemento representa una coeficiente de la función, Ej. $[a_1, a_2, a_3]$. Los elementos pueden tener cualquier valor numérico real, y sus valores serán inicializados aleatoriamente con valores en un rango de [-10, 10]. Para fines de simplicidad, se manejarán los valores acotados a seis decimales a través del documento, pero en el programa no se limitaron estos.

Descripción de la función de evaluación:

Para evaluar las soluciones, se van a reemplazar los coeficientes de la función $a_1 x^2 - a_2 \frac{(e^{-a_3 * x})}{2}$, con sus respectivos coeficientes en el vector solución. Con esta nueva función, se harán evaluaciones para los valores de 'x' donde $x \in [0...9]$. Para cada evaluación, se va a comparar el resultado de la evaluación con el valor meta correspondiente indicado en la siguiente tabla:

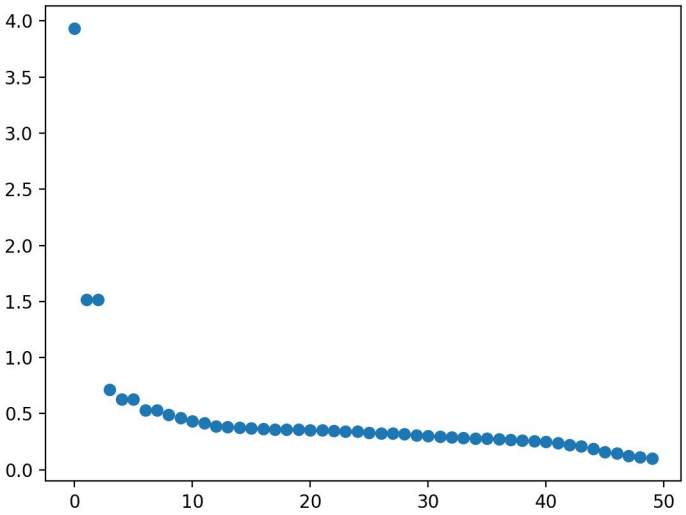
x	f(x)
0	-3.271085
1	2.994633
2	14.999975
3	34.999999
4	62.999999
5	98.999999
6	143
7	195
8	255
9	323

Se calculará la diferencia entre el resultado de la evaluación y el valor meta para cada valor de 'x' con el método de mínimos cuadrados. Buscaremos reducir el error total entre el resultado meta y el obtenido con cada solución para encontrar los coeficientes que mejor se ajusten a la meta.

Resultados obtenidos:

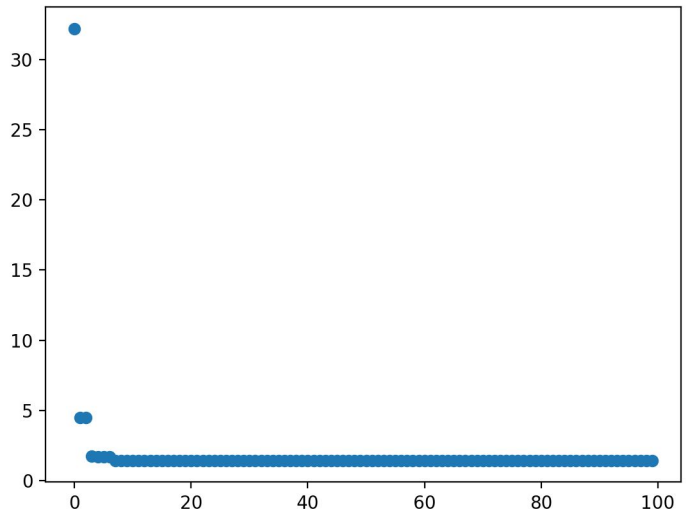
Se corrió el programa repetidas veces con distintos valores para el número de iteraciones y el número de partículas, con la meta de discernir si es mejor elevar las iteraciones o la cantidad de partículas. A continuación se presentan algunos resultados obtenidos para los diferentes parámetros:

(50 iteraciones con 10 partículas)



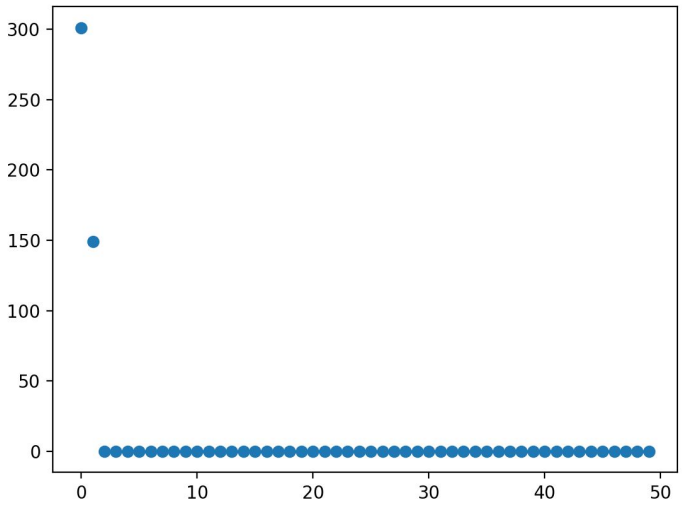
a1 ≈ 4.000227
a2 ≈ 5.197692
a3 ≈ 2.727444
Evaluación - 0.098583

(100 iteraciones con 10 partículas)



a1 ≈ 3.981412
a2 ≈ -4.19261
a3 ≈ -3.177937
Evaluación - 1.441316

(50 iteraciones con 20 partículas)



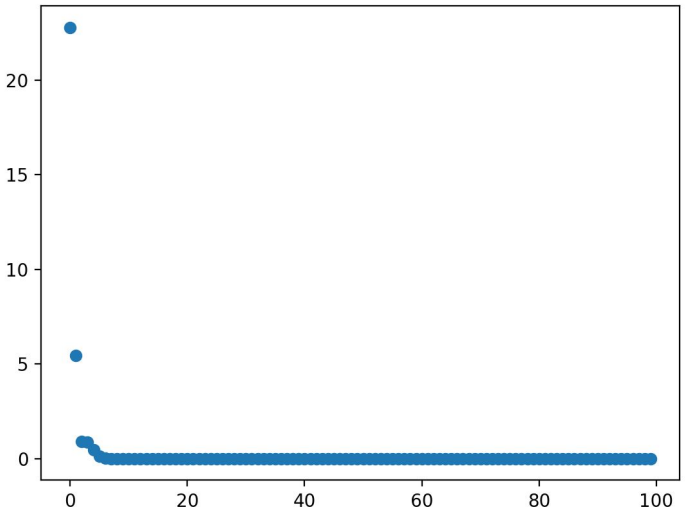
a1 ≈ 4.000000
a2 ≈ 10.700083
a3 ≈ 5.130605
Evaluación - 2.774739e-07

Cabe mencionar que con cada ejecución del programa, para cada variación de parámetros, se obtuvieron resultados muy distintos. Esto probablemente debido a la baja cantidad de iteraciones o partículas que se pueden quedar a medias o atorados en mínimos locales. Se encontró que duplicar el número de partículas resultaba en un aumento mínimo de estabilidad, pero seguían surgiendo soluciones distintas con cada iteración. En varias ejecuciones se obtuvo una solución con el valor evaluación 1.441316, y este se presentó con más frecuencia a la hora de elevar las iteraciones a diferencia de aumentar las partículas. De igual manera parece que tener menos partículas limita el espacio de búsqueda y puede generar una convergencia en un mínimo local de manera precipitada.

Gráfica con la curva de mejor encontrado:

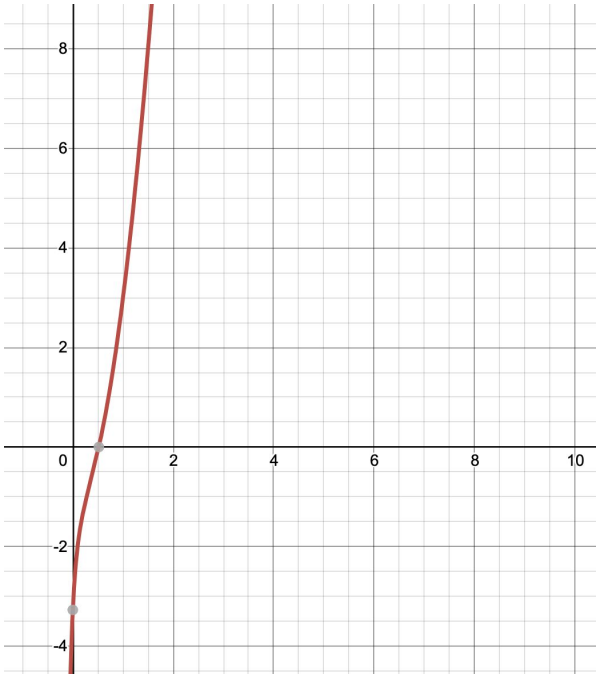
Después de aumentar ambas iteraciones y partículas, incrementó la estabilidad del programa y mejoraron las soluciones. Esta es una de las mejores que surgieron:

(100 iteraciones con 20 partículas)



a1 ≈ 4.000000
a2 ≈ 10.699900
a3 ≈ 5.399993
Evaluación - 1.769361e-10

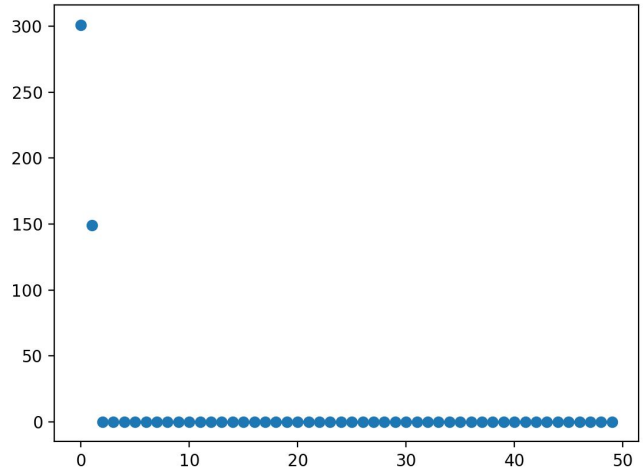
Tomando en cuenta los coeficientes, se graficó la siguiente función: $4\,x^{2} - 10.7^{\frac{(e^{-5.4*x})}{2}}$



Lo cual dió resultados prácticamente idénticos a los resultados meta.

Conclusiones y retos encontrados:

Como el resto de los algoritmos de optimización, en diferentes ejecuciones se pudieron obtener diferentes resultados, pero parece que se puede mejorar este aspecto ajustando parámetros que permitan más iteraciones en un mayor espacio de búsqueda a cambio de tiempo de cómputo. Como visto en las gráficas, este algoritmo puede generar significativamente mejores soluciones con cada iteración al inicio y hacer ajustes pequeños posteriormente.



Durante el desarrollo del programa tuve algunos warnings arrojados indicando overflow de valores escalares a la hora de correr la función de evaluación ya que en casos resultaban valores muy grandes (inf) o muy chicos (nan). Sin embargo estos warnings no afectaron los resultados al aumentar las iteraciones, ya que el error fue disminuyendo y los valores fueron siendo cada vez más razonables.