Algoritmos Evolutivos
TP2
Juan Pablo Schamun

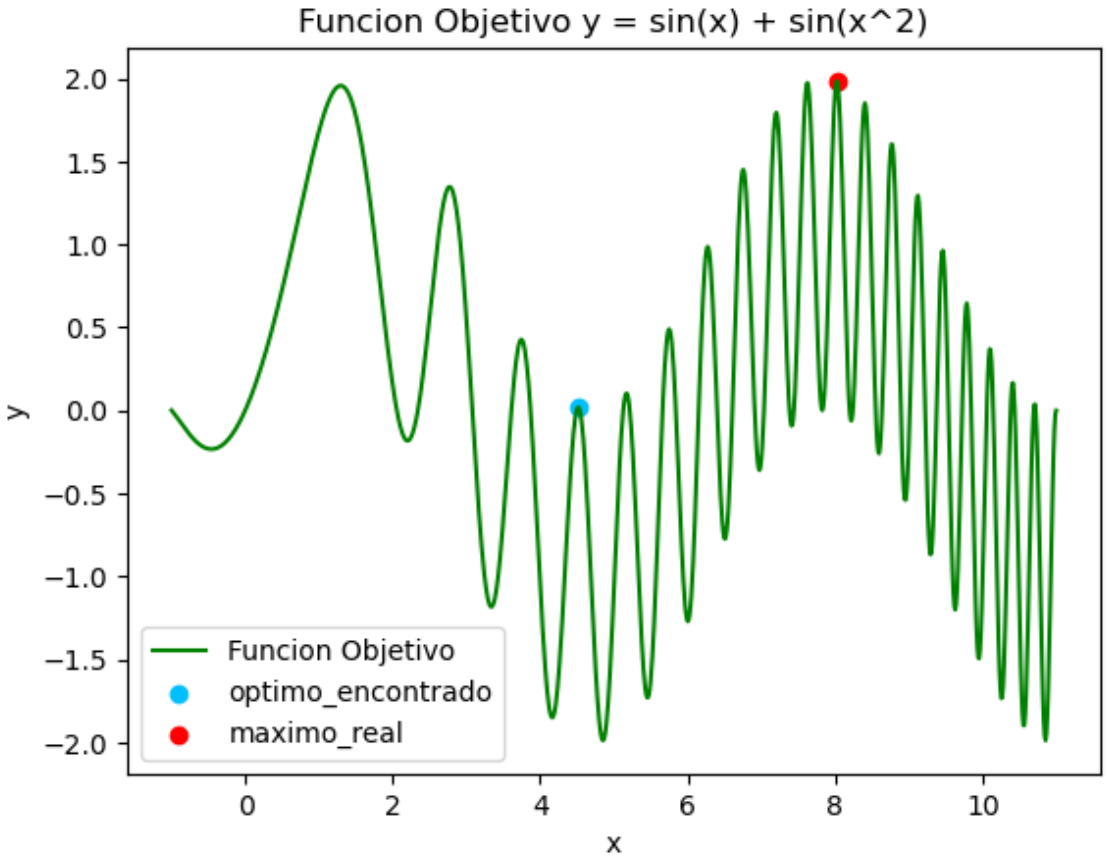
Ejercicio1

a)

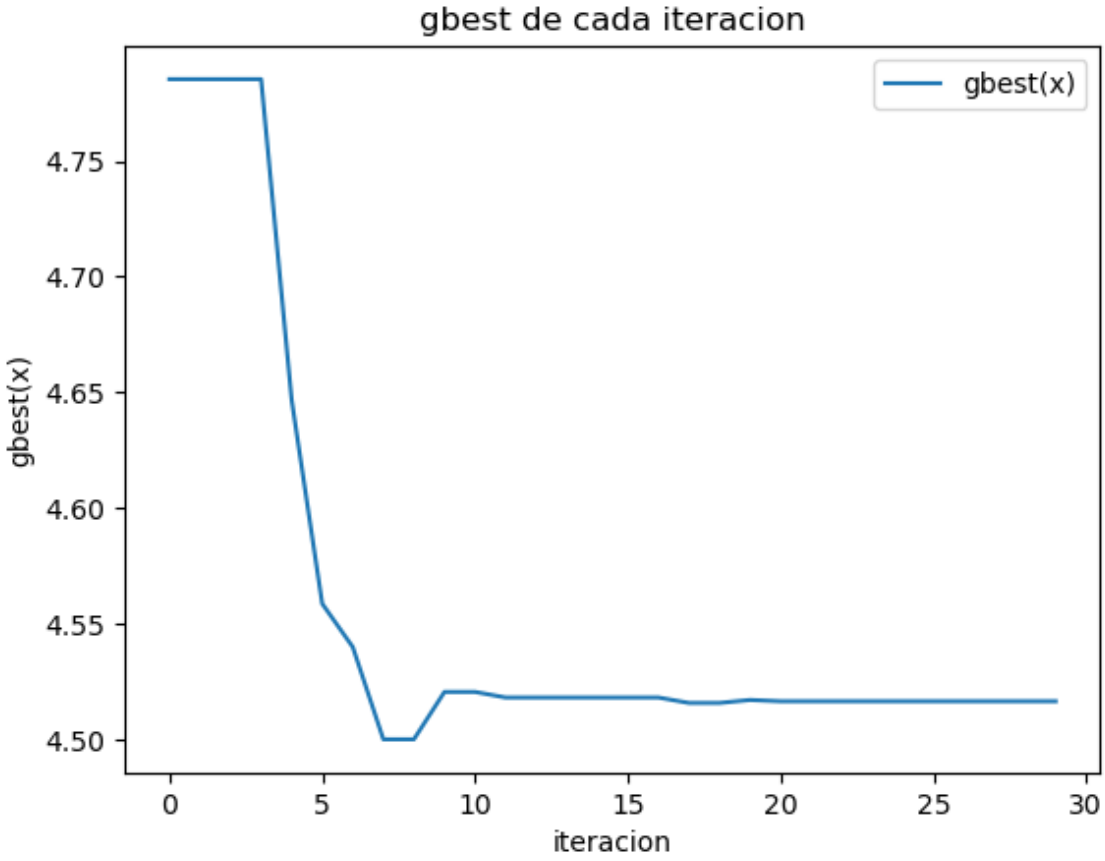
- Solución óptima: x = 4.516
- Valor óptimo: y = 0.019

b) URL GitHub:
https://github.com/juanpsch/AEIV/blob/main/TP2/TP2_1.ipynb

c) Grafico:



d) Gráfico de gbst en cada iteración:



e) Se observa que el algoritmo tiende a estancarse en óptimos locales. Es muy dependiente de la posición inicial de las partículas. Esto puede estar relacionado con una baja inercia, pocas partículas iniciales y una función objetivo multimodal con muchos máximos y alta frecuencia.

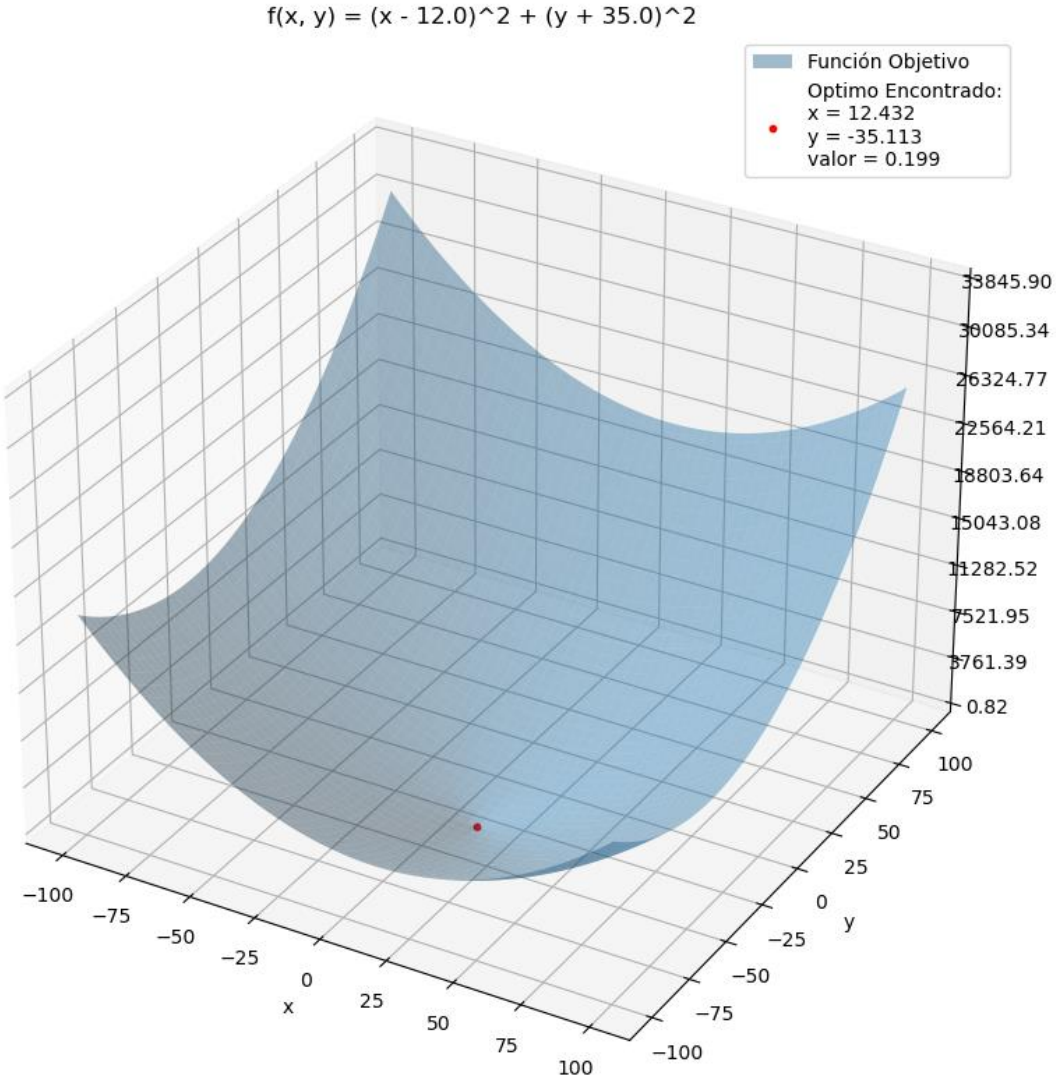
Ejercicio2

a) Introduciendo a=12 y b=35 queda la función
f(x, y) = (x - 12)^2 + (y + 35)^2 para minimizar
   a) Solución óptima: **x = 12.432**; **y = -35.113**
   b) Valor óptimo**: f(x,y) = 0.119**

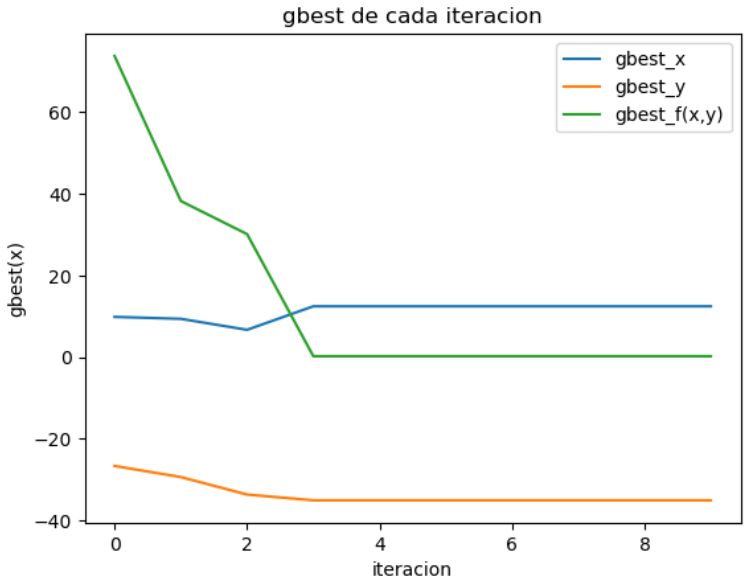
b) URL GitHub:

https://github.com/juanpsch/AEIV/blob/main/TP2/TP2_2.ipynb

c) Gráfico de función objetivo



d) Gráfico de gbest:



e) Cambiando w=0 la función

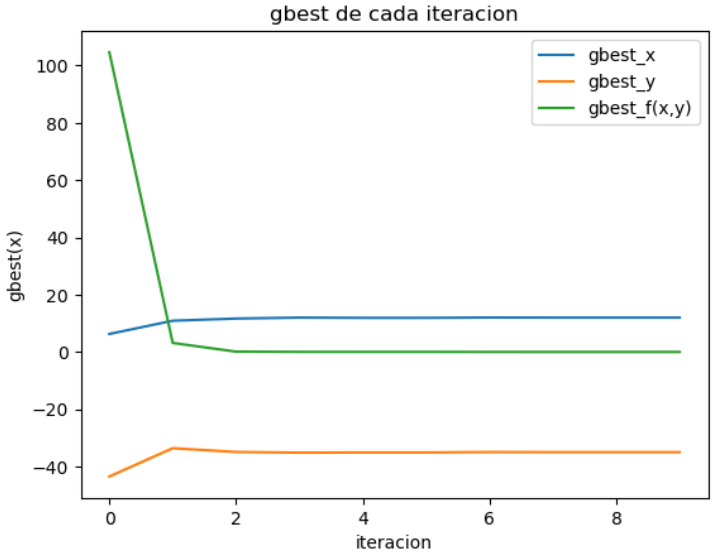
f(x, y) = (x - 12)^2 + (y + 35)^2 para minimizar tiene

a) Solución óptima: **x = 11.97**; **y = -35.02**

Valor óptimo**: f(x,y) = 0.0008**

Se obtiene un óptimo mejor de manera sistemática. Al parecer al ser una función unimodal la inercia no ayuda a converger más rápidamente, ya que no hay mínimos locales en donde pueda quedare estancada la partícula.

Se observa en la siguiente figura, como converge mucho más rápidamente



f) Repetir con pyswarm:

a) Introduciendo a=12 y b=35 queda la función

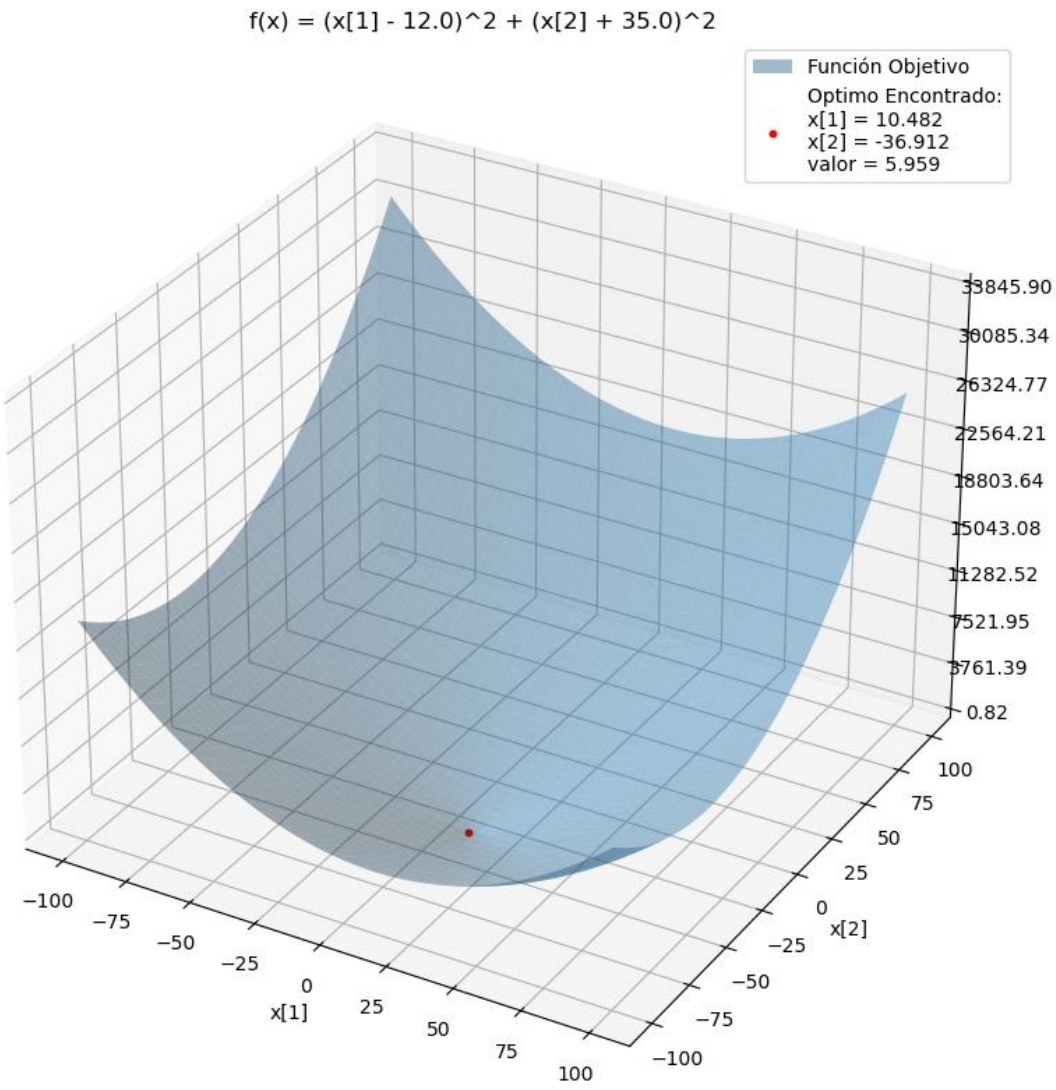
f(x, y) = (x1 - 12)^2 + (x2 + 35)^2 para minimizar

- Solución óptima: **x1 = 10.492**; **x2 = -36.912**
- Valor óptimo**: f(x,y) = 5.959**

b) URL GitHub:

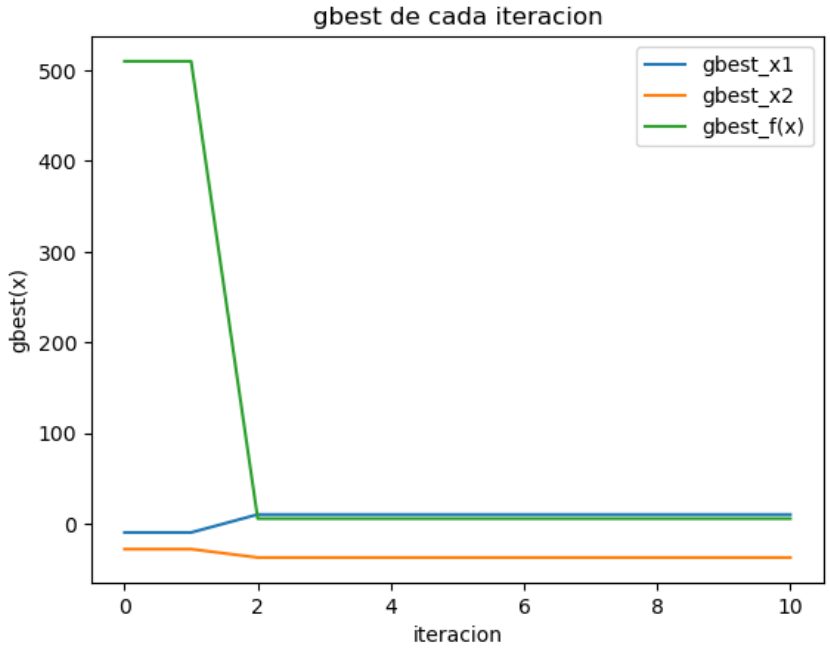
https://github.com/juanpsch/AEIV/blob/main/TP2/TP2_2.ipynb

c) Gráfico de función objetivo:



d) Gráfico de gbest:

e) Cambiando w=0 la función
f(xy) = (x1 - 12)^2 + (x2 + 35)^2 para minimizar tiene
Solución óptima: **x1 = 11.981**; **x2 = -34.969**



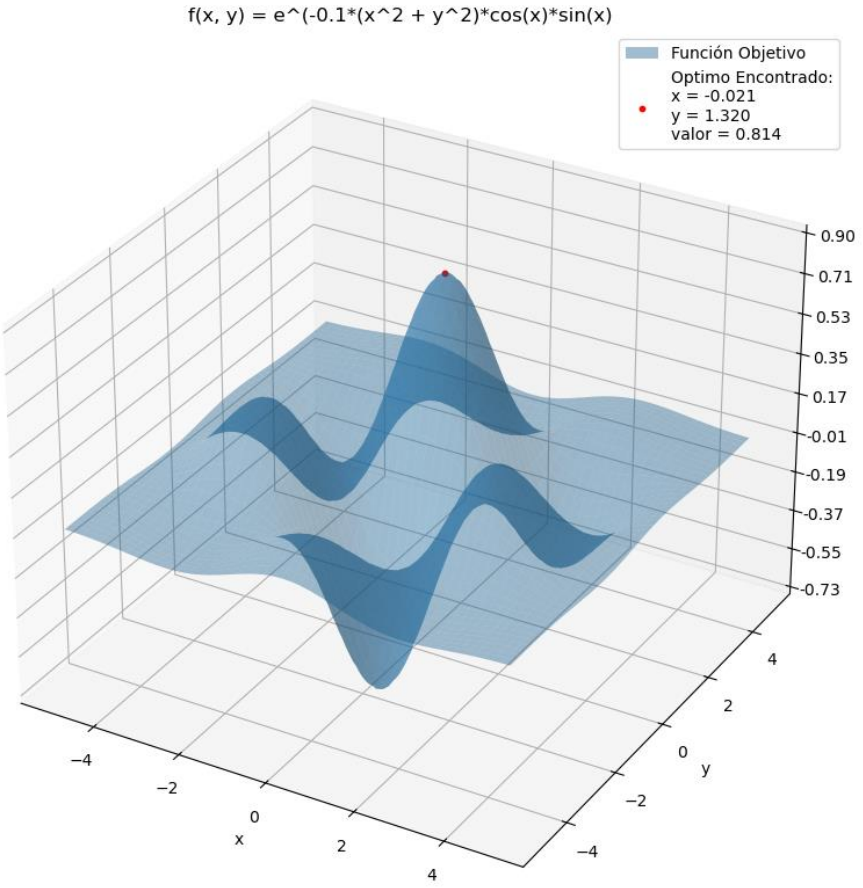
g) Utilizando pyswarm, se tarda más iteraciones en llegar a un óptimo de valores similares al algoritmo casero. Al poner la inercia en cero, esta diferencia ya no es significativa.

Ejercicio3

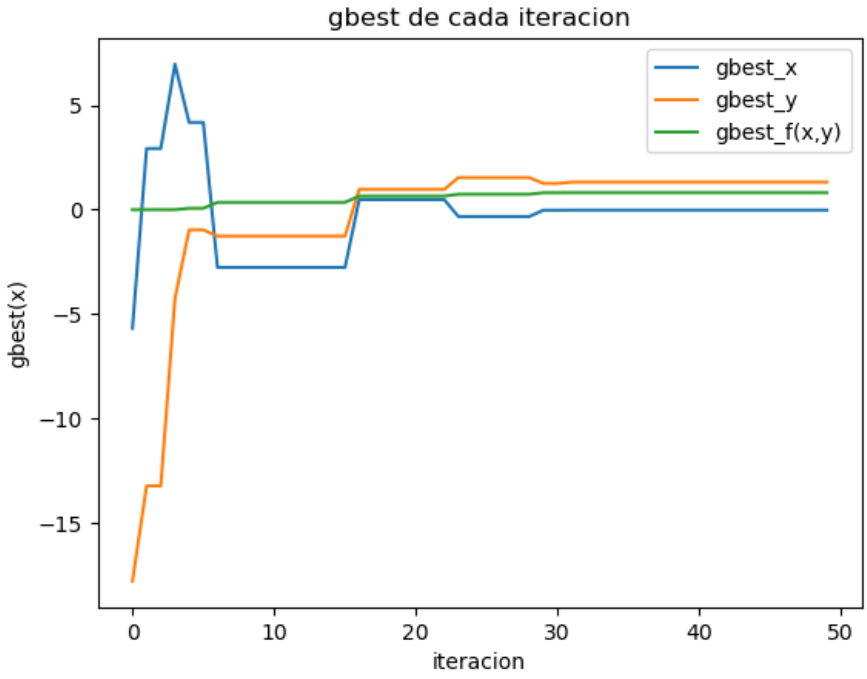
a) Solución óptima: **x = -0.021**; **y = 1.320**
Valor óptimo**: f(x,y) = 0.814**
b) URL GitHub:

https://github.com/juanpsch/AEIV/blob/main/TP2/TP2_3.ipynb

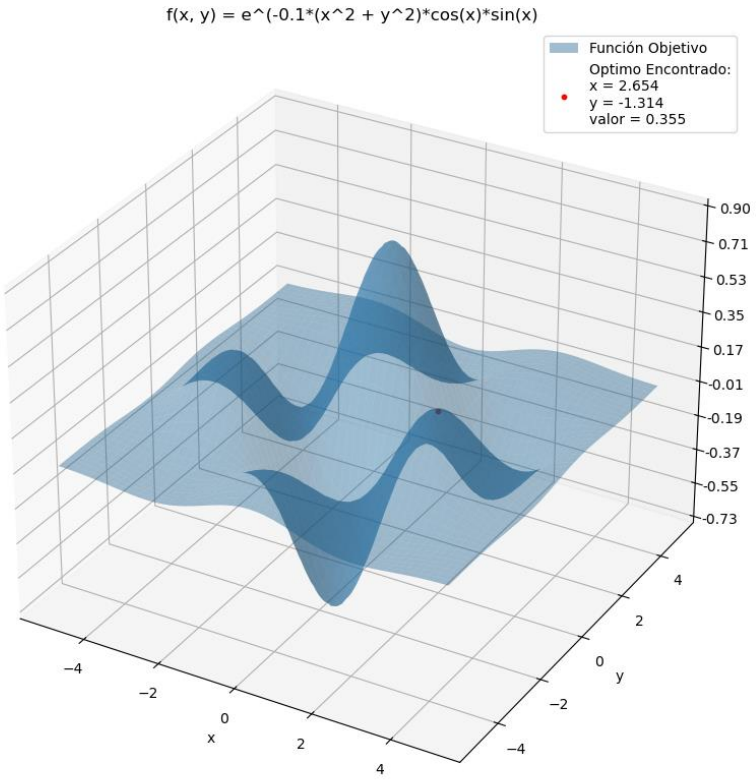
c) Gráfico de función objetivo



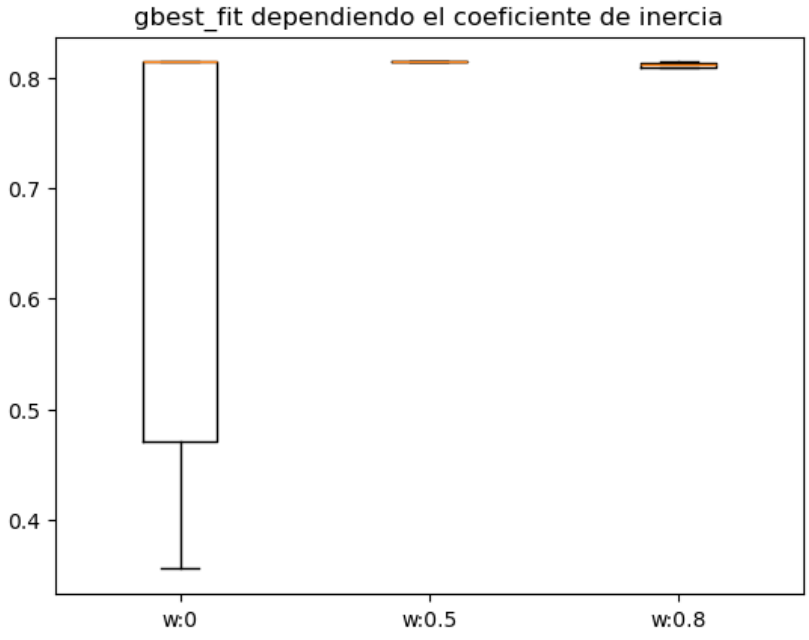
d) Gráfico de gbest:



e) Estableciendo w=0 la solución hallada es muy dependiente a las condiciones iniciales de las partículas, siendo propenso el algoritmo a estancarse en máximos locales.
La función objetivo tiene al menos dos sectores con claros máximos locales cada uno, donde sólo uno es el máximo global
Por ejemplo, una solución hallada fue la siguiente:



f) BoxPlot con diferentes coeficientes de inercia:



g) Se observa que cuando hay inercia es más probable converger al máximo local que cuando no hay inercia.

Ejercicio4

a) Para resolver el sistema, voy intentar minimizar la sumatoria de cada función (despejada para igualar 0) al cuadrado.

De esta manera me queda:

f1(x) = 3x1 + 2x - 9

f2(x) = x1 - 5*x – 4

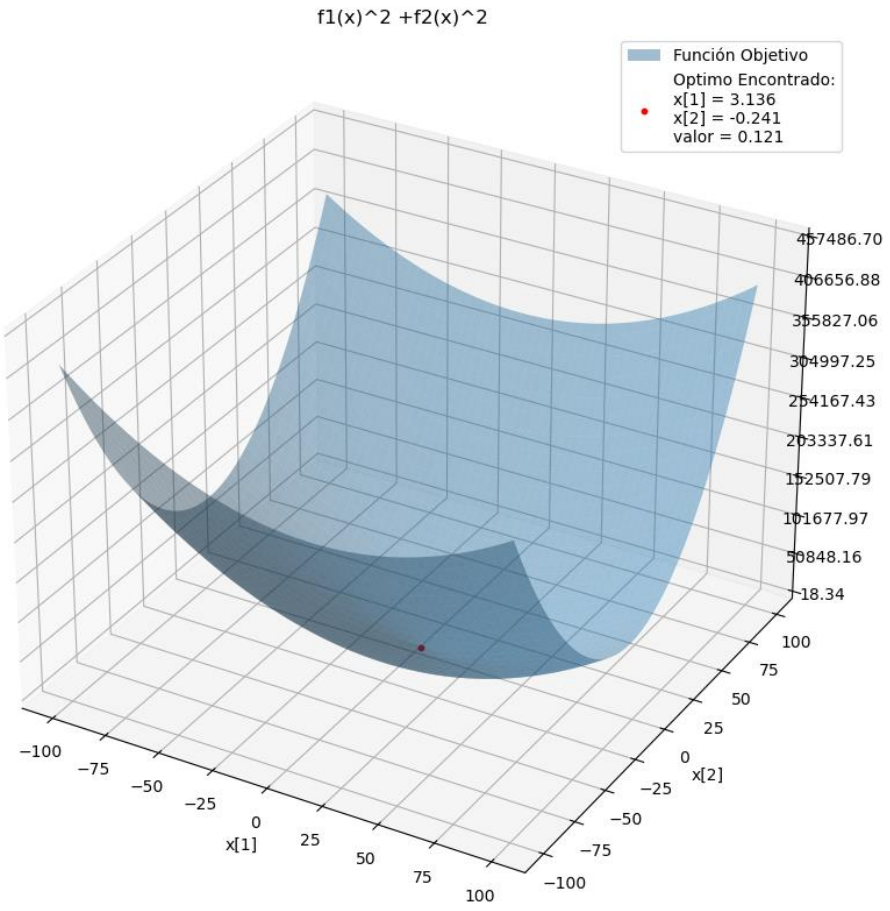
y la función a minimizar es f1^2 + f^2

El código está en el git.

b) Solución al sistema hallada:
x1 = 3.136; x2 = -0.241
Solución Real: x1 = 3.058; x2 = -0.176

c) URL GitHub:
https://github.com/juanpsch/AEIV/blob/main/TP2/TP2_2_4.ipynb

d) La función propuesta a minimizar es unimodal con lo cual con una inercia no muy alta debería poder converger a un mínimo que será cercano a 0, y daría una solución muy acertada. Ver gráfico. Los límites los elegí aleatoriamente entre -100 y 100. En este caso se dio que esos límites incluían al mínimo buscado, pero claramente, dependiendo las funciones del sistema, ese límite puede llegar a excluir a ese mínimo. Esos límites, por tanto, son el parámetro más influyente para lograr encontrar el mínimo. El resto no varía mucho con parámetros estándar.
Para resolver sistemas de n ecuaciones con n incógnitas bastaría con incluirlas como sumatorias del cuadrado de las funciones despejadas, igualadas a 0 en la función objetivo a minimizar. Pero siempre será cada vez más difícil encontrar los límites del universo de búsqueda.
Para sistemas no lineales, sería similar, ya que siempre se está creando una función objetivo como suma de funciones elevadas al cuadrado cuyo mínimo real es 0, con lo cual minimizándola se debería hallar una solución aproximada.


f1(x)^2 +f2(x)^2
Función Objetivo
Optimo Encontrado:
x[1] = 3.136
x[2] = -0.241
valor = 0.121
457486.70
406656.88
355827.06
304997.25
254167.43
203337.61
152507.79
101677.97
50848.16
18.34
x[1]
x[2]