Tarea 3

Robótica aérea

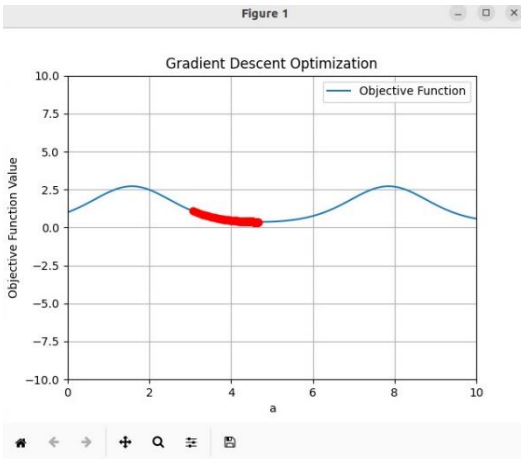
Integrantes:

- Fabián Omar Bolaño Estupiñan ID: 000429409
- Andrés Felipe Pérez Sierra ID: 000429432
- Oscar Stiven Quintero Urrea ID: 000337185

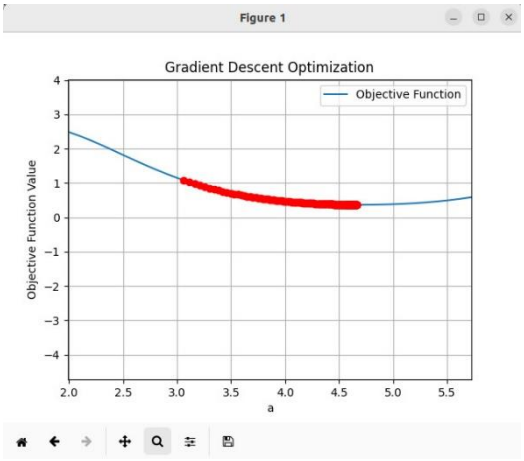
Para la tarea de algoritmo de optimización se utilizó la fórmula $e^{\sin x}$ con ayuda de la librería de cálculo llamada Numpy, dando como resultado la siguiente línea en el código

```
5   def Objective(a):
6       return np.exp(np.sin(a))
```

Para la primera gráfica que es la que se hizo sin optimizar aún, se graficó en x de 0 a 10 y en y de -10 a 10 teniendo como parámetro de inicial $x = 3$, dando como resultado



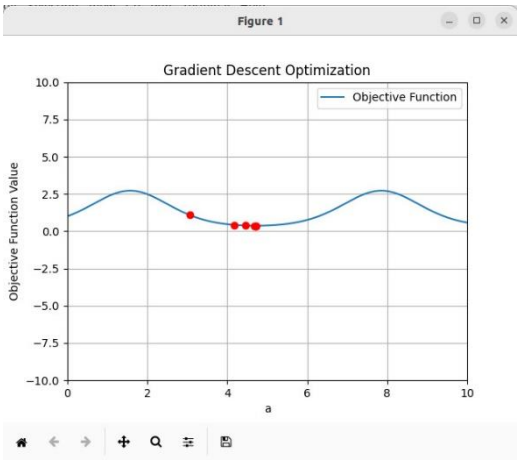
Graficando así una cantidad de puntos que es incontable a simple vista humana incluso si se le hace un acercamiento



Por otro lado, al utilizar el algoritmo de optimización con la fórmula $\frac{|(x_n - x_{n-1})^T[\nabla F(x_n) - \nabla F(x_{n-1})]|}{\|\nabla F(x_n) - \nabla F(x_{n-1})\|^2}$ llamada tasa de aprendizaje adaptada para una función de una sola variable, la cual es una ecuación que a medida que itera el step del código va siendo cambiado; a diferencia del código sin tasa de aprendizaje el cual tiene un step fijo. Se implementa de la siguiente manera en el código

```
27  # Gradient Descent
28  while error > tol:
29      grad_n1= (Objective(x[k]) - Objective(x[k] - h)) / h
30      x.append(x[k] - step * grad_n1)
31      grad_n= (Objective(x[k + 1])- Objective(x[k + 1]- h))/h
32      error = abs(x[k + 1] - x[k])  # to stop algorithm
33      step = abs(((x[k + 1])- x[k])*(grad_n - grad_n1))/(abs((grad_n - grad_n1))**2)
34      k += 1
35
```

Da como resultado la graficación de los puntos en la función de la siguiente manera



Que al hacer zoom se notará que la cantidad de puntos es reducida a 5 dando un resultado satisfactorio el uso del algoritmo