Fecha de entrega	Tarea 3	6 de Junio
10 Junio	Introducción al Diseño de Experimentos	2025
2025	Dra. Norma Icoquih Zapata Peñasco	
	Dra. Gabriela Berenice Díaz Cortés	

Ejercicio 1. Ajuste de parámetros para permeabilidades relativas usando el algoritmo de Evolutivo Diferencial

En ingeniería de yacimientos, las permeabilidades relativas (k_r) suelen modelarse como funciones no lineales de la saturación de agua (S_w). Un modelo común es la función de Corey:

$$k_{rw} = k_{rw0}\left(\frac{S_w - S_{wr}}{1 - S_{wr} - S_{or}}\right)^{n_w}, \quad k_{ro} = k_{ro0}\left(\frac{1 - S_w - S_{or}}{1 - S_{wr} - S_{or}}\right)^{n_o} \tag{1}$$

Donde:

- k_{rw0} y k_{ro0} son las permeabilidades relativas máximas para agua y aceite.
- S_{wr} es la saturación residual de agua.
- S_{or} es la saturación residual de aceite.
- n_w y n_o son exponentes que determinan la forma de la curva.

Datos experimentales

S_w	k_{rw}	k_{ro}
0.20	0.02043	0.47018
0.25	0.03725	0.34831
0.30	0.06070	0.25510
0.35	0.09138	0.17839
0.40	0.12527	0.12192
0.45	0.15629	0.07397
0.50	0.20728	0.04302
0.55	0.25201	0.02135
0.60	0.30686	0.008581
0.65	0.37060	0.002643
0.70	0.43427	0.000306
0.75	0.51454	0.000000

Objetivo

Ajustar los parámetros del modelo de Corey para que las curvas generadas se ajusten a los datos dados, utilizando el algoritmo de **Evolución Diferencial** mediante la función `scipy.optimize.differential_evolution`.

Función Objetivo

La función objetivo, llamada `objetivo`, debe:

1. Recibir un vector de parámetros:

$$[k_{rw0}, k_{ro0}, S_{wr}, S_{or}, n_w, n_o] \tag{2}$$

2. Calcular las curvas modeladas de k_{rw} y k_{ro} para los valores de saturación S_w usando las ecuaciones de Corey.

3. Calcular el error cuadrático medio (MSE) entre los valores modelados y los experimentales:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left[\left(k_{rw,\text{modelo}}^{(i)} - k_{rw,\text{datos}}^{(i)}\right)^2 + \left(k_{ro,\text{modelo}}^{(i)} - k_{ro,\text{datos}}^{(i)}\right)^2\right] \tag{3}$$

   Donde:

   - N es el número de puntos de saturación.
   - $k_{rw,\text{modelo}}^{(i)}$ y $k_{ro,\text{modelo}}^{(i)}$ son los valores generados por el modelo.
   - $k_{rw,\text{datos}}^{(i)}$ y $k_{ro,\text{datos}}^{(i)}$ son los valores experimentales.

4. Devolver el valor numérico del error.

Límites para los parámetros

Los rangos usados para los parámetros en el algoritmo de optimización son:

Parámetro	Rango
k_{rw0}	$[0{,}0, 1{,}0]$
k_{ro0}	$[0{,}0, 1{,}0]$
S_{wr}	$[0{,}0, 0{,}4]$
S_{or}	$[0{,}0, 0{,}4]$
n_w	$[1{,}0, 5{,}0]$
n_o	$[1{,}0, 5{,}0]$

$$bounds = [(0,1{,}0), (0,1{,}0), (0{,}0,1), (0{,}0,1), (1{,}0,5{,}0), (1{,}0,5{,}0)]$$

Aplicar Evolución Diferencial

Se usa el siguiente comando para encontrar los parámetros óptimos:

```
from scipy.optimize import differential_evolution
resultado = differential_evolution(objetivo, bounds)
```

El vector `resultado.x` contiene los parámetros óptimos encontrados.

Visualización de resultados

Una vez obtenidos los parámetros óptimos, grafica las curvas generadas por el modelo junto con los datos experimentales para evaluar visualmente la calidad del ajuste.
Documentación oficial de `differential_evolution`:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html

Ejercicio 2. Comparación de algoritmos

Usa la librería `SwarmPackagePy` para comparar tres algoritmos de optimización disponibles en dicha librería.
Selecciona tres algoritmos y una función objetivo para realizar la comparación.
Ejecuta cada algoritmo 30 veces con 25 partículas, dentro del intervalo de $[-10, 10]$, durante 10 iteraciones.
Para cada algoritmo, calcula el promedio, la desviación estándar y el intervalo de confianza de los resultados obtenidos.
Organiza los resultados en una tabla y concluye cuál de los tres algoritmos presenta mejor desempeño para la función objetivo seleccionada.
A continuación se presenta un ejemplo de código que se puede utilizar para realizar el experimento:

```
import SwarmPackagePy
from SwarmPackagePy import testFunctions as tf
from SwarmPackagePy import animation, animation3D

n_pop = 25        # Número de partículas
```

```
n_its = 10        # Número de iteraciones
c_min = -10       # Límite inferior del espacio de búsqueda
c_max = 10        # Límite superior del espacio de búsqueda
n_dim = 2         # Dimensionalidad del problema

# Inicialización y ejecución del algoritmo PSO con parámetros específicos
alh = SwarmPackagePy.pso(n_pop, tf.sphere_function, c_min, c_max, n_dim, n_its,
                         w=0.5, c1=1, c2=1)

# Evaluación del valor de la función objetivo en la mejor posición encontrada
tf.sphere_function(alh.get_Gbest())
```