Page

aim: Grey wolf optimiser:

Algorithm:

Initialize the population of wolves (positions) randomly within the search space
Define the maximum no. of iterations (T) and population size (N)
Define the fitness function to evaluate solutions.

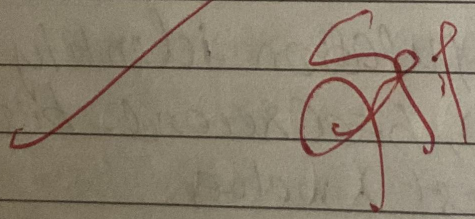
Evaluate the fitness of each wolf in the population identify the alpha (best solution), beta (second_best) and delta (third best) wolves

For t = 1 to T:
  for each wolf i in the population
    for each dimension d
      A1 = 2 * a * rand() - a
      C1 = 2 * rand()
      D_alpha = [C1 * x_alpha[d] - x_i[d]]
      x1 = x_alpha[d] - A1 * D_alpha

      A2 = 2 * a * rand() - a
      C2 = 2 * rand()
      D_beta = C2 * x_beta[d] - x_i[d]
      x2 = x_beta[d] - A2 * D_beta

A3 = 2*a*rand() - a
C3 = 2*rand()
D_delta = C3 * X_delta[d] - X_i[d]
X3 = X_delta[d] - A3 * D_delta

X_i[d] = (X1 + X2 + X3) / 3
End For
End For
a = 2 - (2 * t / T)
Update alpha, beta & delta values based on fitness
End For



```
import numpy as np.

def objective_function(x):
    return sum(x**2)

def initialize_population(dim, n_wolves, bounds)
    return np.random.uniform(bounds[0],
        bounds[1], (n_wolves, dim))

def gwo(objective_function, bounds, dim,
    n_wolves, n_iterations):
    wolves = initialize_pop(dim, n_wolves, bounds)
    fitness = np.apply_along_axis(objective_
        function, 1, wolves)

# Initialize alpha, beta, delta
alpha, beta, delta = np.argsort(fitness)[:3]
alpha_pos, alpha_score = wolves[alpha],
    fitness[alpha]
beta_pos, beta_score = wolves[beta], fitness
    [beta]
delta_pos, delta_score = wolves[delta],
    fitness[delta]

# Main optimization loop
for iteration in range(n_iterations):
    a = 2 - 2*(iteration/n_iterations)
```

```
for i in range(n_wolves):
    for j in range(dim):
# update each wolf's position
        r1, r2 = np.random.rand(), np.
            random.rand()
        A1, C1 = 2*a*r1 - a, 2*r2
        D_alpha = abs(C1*alpha_pos[j] -
            wolves[i,j])
        X1 = alpha_pos[j] - A1*D_alpha

        r1, r2 = np.random.rand(),
        np.random.rand()
        A2, C2 = 2*a*r1 - a, 2*r2
        D_beta = abs(C2*delta_pos[j] -
            wolves[i,j])
        X3 = delta_pos[j] - A3*D_delta

        # Avg position update
        wolves[i,j] = (X1 + X2 + X3)/3.0

# enforce bounds
wolves[i,:] = np.clip(wolves[i,:], bounds[0],
    bounds[1])
# Calculate fitness & update alpha, beta, delta
fitness = np.apply_along_axis(obj_fun,
wolves)
sorted_indices = np.argsort(fitness)
alpha, beta, delta = sorted_indices[:3]
beta_pos, beta_score = wolves[beta],
    fitness[beta]
```

```
    delta_pos, delta_score = wolves[delta],
        fitness[delta]

Return alpha_pos, alpha_score

# example usage
dim = 5
bounds = (-10, 10)
n_wolves = 30
n_iterations = 100

best_solution, best_score = gwo(objective_function,
    bounds, dim, n_wolves, n_iterations)

print(f"Best solution: {best_solution}")
print(f"Best score: {best_score}")
```

Output:

```
Best solution: [1.11961142e-11 1.39239509e-11
-1.60198045e-11 -1.40895814e-11
-1.19655093e-11]

Best score: 9.17557965435404e-22
```