GREY WOLF OPTIMISER

Code:

```
# Grey Wolf

import numpy as np

def obj_fn(x):
    """Objective function to minimize."""
    return np.sum(x**2)  # Example: Sphere function

def gwo(obj_fn, dim, wolves, iters, lb, ub):
    """Grey Wolf Optimm,kkl,k,,lkpppppppppizer (GWO) implementation."""
    # Initialize wolf positions
    pos = np.random.uniform(low=lb, high=ub, size=(wolves, dim))
    a_pos, b_pos, d_pos = np.zeros(dim), np.zeros(dim), np.zeros(dim)
    a_score, b_score, d_score = float("inf"), float("inf"), float("inf")

    for t in range(iters):
        for i in range(wolves):
            fit = obj_fn(pos[i])
            # Update Alpha, Beta, Delta
            if fit < a_score:
                d_score, d_pos = b_score, b_pos.copy()
                b_score, b_pos = a_score, a_pos.copy()
                a_score, a_pos = fit, pos[i].copy()
            elif fit < b_score:
                d_score, d_pos = b_score, b_pos.copy()
                b_score, b_pos = fit, pos[i].copy()
            elif fit < d_score:
```

```
        d_score, d_pos = fit, pos[i].copy()

# Update wolf positions
a = 2 - t * (2 / iters)  # Linearly decreasing factor
for i in range(wolves):
    for j in range(dim):
        r1, r2 = np.random.rand(), np.random.rand()
        A1, C1 = 2 * a * r1 - a, 2 * r2
        D_a = abs(C1 * a_pos[j] - pos[i, j])
        X1 = a_pos[j] - A1 * D_a

        r1, r2 = np.random.rand(), np.random.rand()
        A2, C2 = 2 * a * r1 - a, 2 * r2
        D_b = abs(C2 * b_pos[j] - pos[i, j])
        X2 = b_pos[j] - A2 * D_b

        r1, r2 = np.random.rand(), np.random.rand()
        A3, C3 = 2 * a * r1 - a, 2 * r2
        D_d = abs(C3 * d_pos[j] - pos[i, j])
        X3 = d_pos[j] - A3 * D_d

        # Update position
        pos[i, j] = (X1 + X2 + X3) / 3

    # Keep wolves within bounds
    pos[i] = np.clip(pos[i], lb, ub)

# Print progress
print(f"Iter {t+1}/{iters}, Best Score: {a_score}, Best Pos: {a_pos}")
```

```
    return a_score, a_pos

# Parameters
dim = 5       # Problem dimension
wolves = 20   # Number of wolves
iters = 50   # Number of iterations
lb = -10      # Lower bound
ub = 10       # Upper bound

# Run GWO
best_score, best_pos = gwo(obj_fn, dim, wolves, iters, lb, ub)
print("\nFinal Best Score:", best_score)
print("Final Best Pos:", best_pos)
```

Output:

```
Iter 44/50, Best Score: 3.6384318663945005e-09, Best Pos: [-3.03609398e-05  2.85365921e-05  2.57152534e-05  2.68369104e-05
  2.28284055e-05]
Iter 45/50, Best Score: 3.3320450798049916e-09, Best Pos: [-2.52105791e-05  2.68237075e-05  2.20522287e-05  3.18528402e-05
  2.18187139e-05]
Iter 46/50, Best Score: 3.0629745568651214e-09, Best Pos: [-2.73630476e-05  2.82810426e-05  1.98730573e-05  2.64678645e-05
  2.04678909e-05]
Iter 47/50, Best Score: 2.9639650951638374e-09, Best Pos: [-2.53901778e-05  2.67901836e-05  2.17957463e-05  2.69457229e-05
  2.00115839e-05]
Iter 48/50, Best Score: 2.9009306337631494e-09, Best Pos: [-2.45654095e-05  2.63794204e-05  2.22912615e-05  2.59449946e-05
  2.07738872e-05]
Iter 49/50, Best Score: 2.757516734618361e-09, Best Pos: [-2.43128984e-05  2.64208350e-05  2.11191357e-05  2.47965912e-05
  2.01853995e-05]
Iter 50/50, Best Score: 2.722932571502108e-09, Best Pos: [-2.46416651e-05  2.65663891e-05  2.05249416e-05  2.45481437e-05
  1.96484935e-05]

Final Best Score: 2.722932571502108e-09
Final Best Pos: [-2.46416651e-05  2.65663891e-05  2.05249416e-05  2.45481437e-05
  1.96484935e-05]
```