# BIS LAB

# LAB 6

# Grey Wolf Optimizer for Function Optimization

Use Grey Wolf Optimization to find the best solution for a mathematical function by mimicking the hunting behavior of grey wolves.

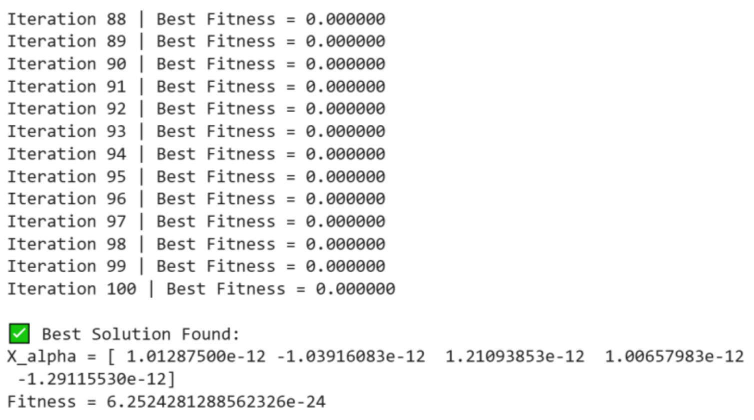
PYTHON CODE:

```
import numpy as np


# 1. Define the Objective Function (to be minimized)

def objective_function(x):

    # Example: Sphere function (minimum at x = 0)

    return np.sum(x ** 2)


# 2. Initialize the Grey Wolf Optimizer

def grey_wolf_optimizer(f, n=20, dim=5, MaxIter=50, lb=-10, ub=10):

    # Initialize the population of wolves randomly within [lb, ub]

    X = np.random.uniform(lb, ub, (n, dim))


    # Evaluate fitness

    fitness = np.array([f(x) for x in X])


    # Identify alpha, beta, delta wolves

    idx = np.argsort(fitness)

    X_alpha, X_beta, X_delta = X[idx[0]], X[idx[1]], X[idx[2]]

    f_alpha, f_beta, f_delta = fitness[idx[0]], fitness[idx[1]],
fitness[idx[2]]


    # Main optimization loop

    for t in range(MaxIter):

        a = 2 - 2 * (t / MaxIter)  # linearly decreases from 2 to 0
```

```
for i in range(n):
    for j in range(dim):
        # ---- Alpha influence ----
        r1, r2 = np.random.rand(), np.random.rand()
        A1 = 2 * a * r1 - a
        C1 = 2 * r2
        D_alpha = abs(C1 * X_alpha[j] - X[i][j])
        X1 = X_alpha[j] - A1 * D_alpha

        # ---- Beta influence ----
        r1, r2 = np.random.rand(), np.random.rand()
        A2 = 2 * a * r1 - a
        C2 = 2 * r2
        D_beta = abs(C2 * X_beta[j] - X[i][j])
        X2 = X_beta[j] - A2 * D_beta

        # ---- Delta influence ----
        r1, r2 = np.random.rand(), np.random.rand()
        A3 = 2 * a * r1 - a
        C3 = 2 * r2
        D_delta = abs(C3 * X_delta[j] - X[i][j])
        X3 = X_delta[j] - A3 * D_delta

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

    # Enforce boundary limits
    X[i] = np.clip(X[i], lb, ub)

# Recalculate fitness
```

```python
        fitness = np.array([f(x) for x in X])

        # Update alpha, beta, delta
        idx = np.argsort(fitness)
        if fitness[idx[0]] < f_alpha:
            X_alpha, f_alpha = X[idx[0]], fitness[idx[0]]
        if fitness[idx[1]] < f_beta:
            X_beta, f_beta = X[idx[1]], fitness[idx[1]]
        if fitness[idx[2]] < f_delta:
            X_delta, f_delta = X[idx[2]], fitness[idx[2]]

        # Display progress
        print(f"Iteration {t+1} | Best Fitness = {f_alpha:.6f}")

    # Return best solution
    return X_alpha, f_alpha


# 3. Run the optimizer
best_position, best_score = grey_wolf_optimizer(objective_function,
n=30, dim=5, MaxIter=100, lb=-5, ub=5)


print("\n✅ Best Solution Found:")
print("X_alpha =", best_position)
print("Fitness =", best_score)
```

Output:



```
Iteration 88 | Best Fitness = 0.000000
Iteration 89 | Best Fitness = 0.000000
Iteration 90 | Best Fitness = 0.000000
Iteration 91 | Best Fitness = 0.000000
Iteration 92 | Best Fitness = 0.000000
Iteration 93 | Best Fitness = 0.000000
Iteration 94 | Best Fitness = 0.000000
Iteration 95 | Best Fitness = 0.000000
Iteration 96 | Best Fitness = 0.000000
Iteration 97 | Best Fitness = 0.000000
Iteration 98 | Best Fitness = 0.000000
Iteration 99 | Best Fitness = 0.000000
Iteration 100 | Best Fitness = 0.000000

✅ Best Solution Found:
X_alpha = [ 1.01287500e-12 -1.03916083e-12  1.21093853e-12  1.00657983e-12
 -1.29115530e-12]
Fitness = 6.2524281288562326e-24
```