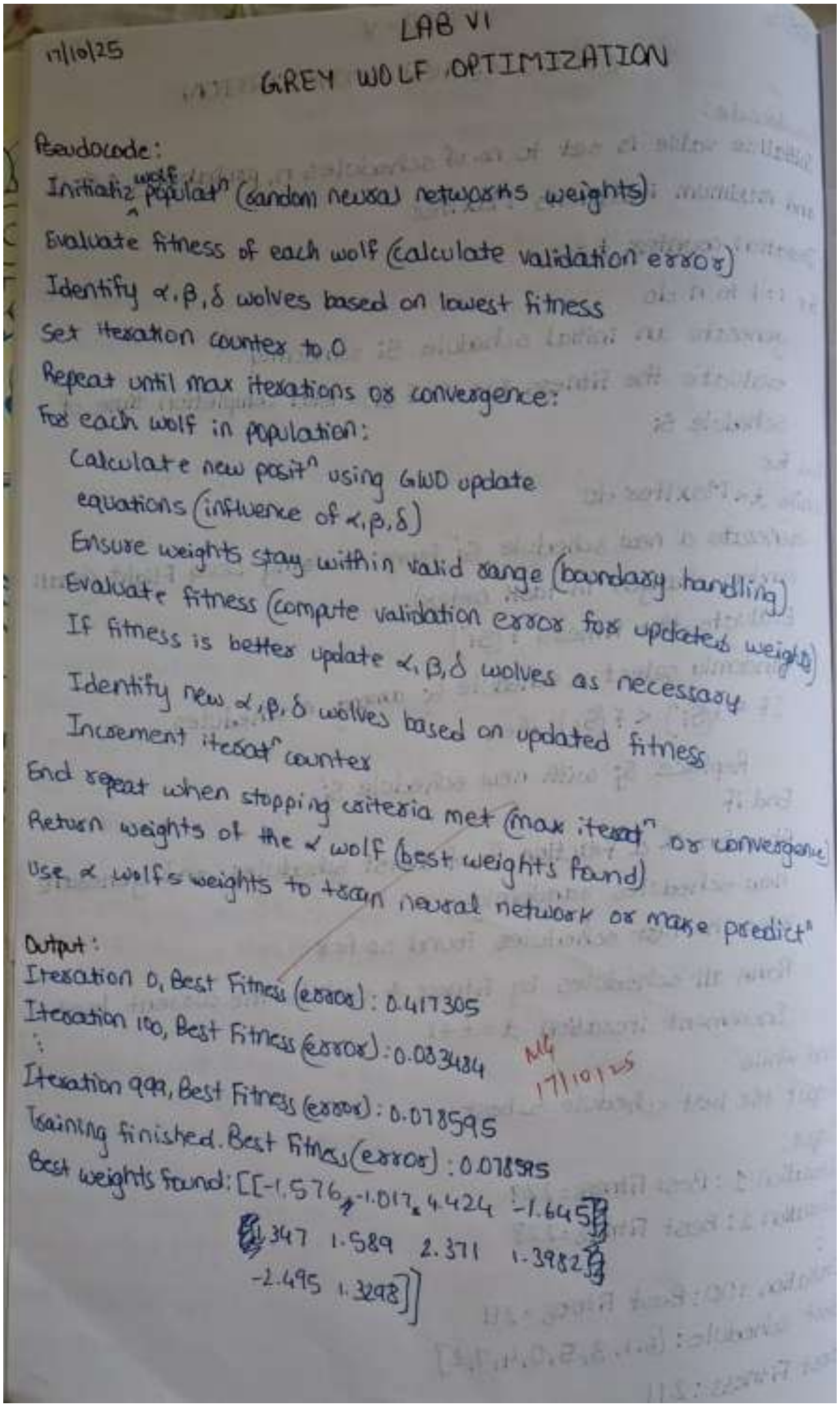
Lab 6

Grey Wolf Optimization



17/10/25

LAB VI

GREY WOLF OPTIMIZATION

Pseudocode:

```
Initialize wolf populatn (random neural networks weights)
Evaluate fitness of each wolf (calculate validation error)
Identify α, β, δ wolves based on lowest fitness
Set iteration counter to 0
Repeat until max iterations or convergence:
For each wolf in population:
    Calculate new positn using GWO update
    equations (influence of α, β, δ)
    Ensure weights stay within valid range (boundary handling)
    Evaluate fitness (compute validation error for updated weights)
    If fitness is better update α, β, δ wolves as necessary
    Identify new α, β, δ wolves based on updated fitness
    Increment iteratn counter
End repeat when stopping criteria met (max iteratn or convergence)
Return weights of the α wolf (best weights found)
Use α wolfs weights to train neural network or make predictn
```

Output:

```
Iteration 0, Best Fitness (error): 0.417305
Iteration 100, Best Fitness (error): 0.083484
:
Iteration 999, Best Fitness (error): 0.078595
Training finished. Best fitness (error): 0.078595
Best weights found: [[-1.576, -1.017, 4.424 -1.645]
                     1.347  1.589  2.371  1.3982]
                     -2.495  1.3298]]
```

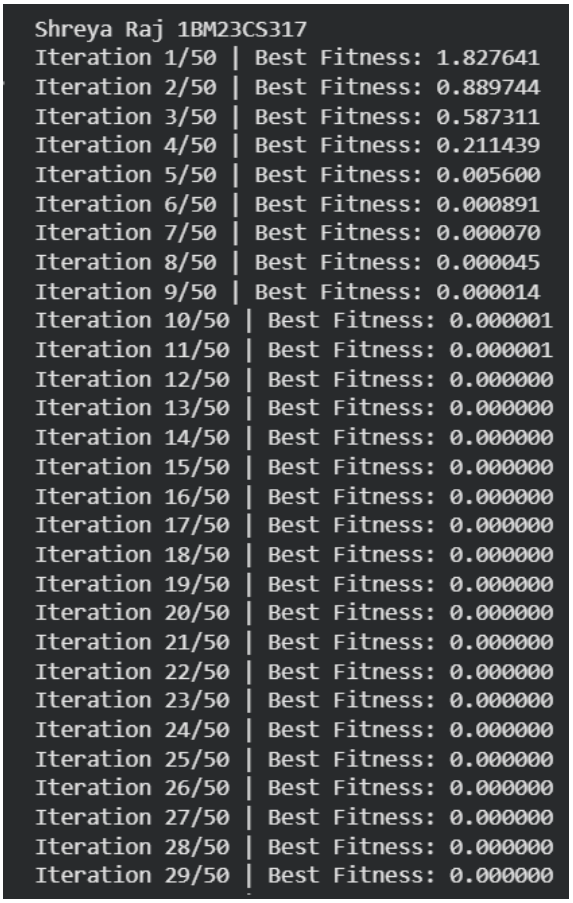
MG 17/10/25

Code:

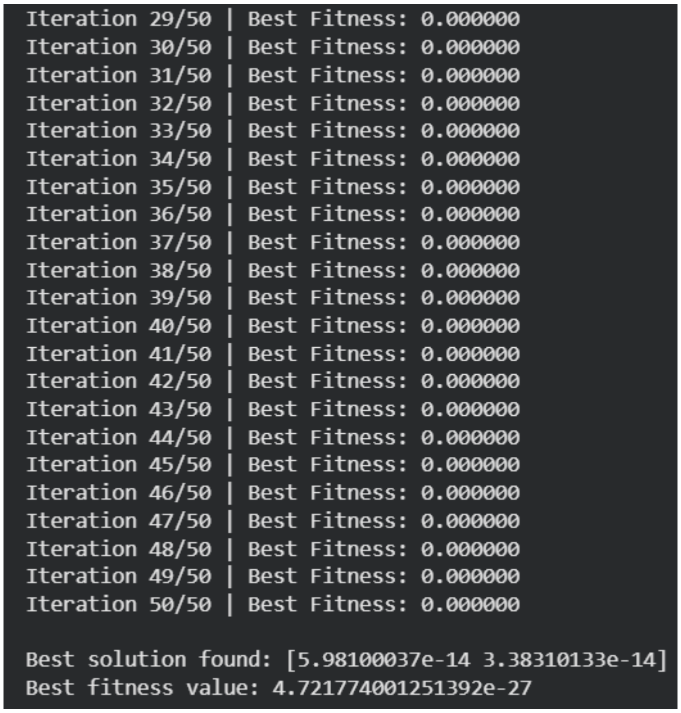
```
import numpy as np
print('Shreya Raj 1BM23CS317')
def objective_function(x):
    return np.sum(x**2)


def grey_wolf_optimizer(num_wolves=30, dim=2, max_iter=50, lower_bound=-10, upper_bound=10):
    wolves = np.random.uniform(lower_bound, upper_bound, (num_wolves, dim))


    Alpha_pos = np.zeros(dim)
    Beta_pos = np.zeros(dim)
    Delta_pos = np.zeros(dim)


    Alpha_score = float("inf")
    Beta_score = float("inf")
    Delta_score = float("inf")


    for t in range(max_iter):
        for i in range(num_wolves):
            wolves[i] = np.clip(wolves[i], lower_bound, upper_bound)


            fitness = objective_function(wolves[i])


            if fitness < Alpha_score:
                Delta_score = Beta_score
                Delta_pos = Beta_pos.copy()
                Beta_score = Alpha_score
                Beta_pos = Alpha_pos.copy()
                Alpha_score = fitness
                Alpha_pos = wolves[i].copy()
            elif fitness < Beta_score:
                Delta_score = Beta_score
                Delta_pos = Beta_pos.copy()
                Beta_score = fitness
                Beta_pos = wolves[i].copy()
            elif fitness < Delta_score:
                Delta_score = fitness
                Delta_pos = wolves[i].copy()

        a = 2 - t * (2 / max_iter)

        for i in range(num_wolves):
            for j in range(dim):
                r1 = np.random.rand()
                r2 = np.random.rand()

                A1 = 2 * a * r1 - a
                C1 = 2 * r2
                D_alpha = abs(C1 * Alpha_pos[j] - wolves[i][j])
                X1 = Alpha_pos[j] - A1 * D_alpha
```

```
                r1 = np.random.rand()
                r2 = np.random.rand()
                A2 = 2 * a * r1 - a
                C2 = 2 * r2
                D_beta = abs(C2 * Beta_pos[j] - wolves[i][j])
                X2 = Beta_pos[j] - A2 * D_beta

                r1 = np.random.rand()
                r2 = np.random.rand()
                A3 = 2 * a * r1 - a
                C3 = 2 * r2
                D_delta = abs(C3 * Delta_pos[j] - wolves[i][j])
                X3 = Delta_pos[j] - A3 * D_delta

                wolves[i][j] = (X1 + X2 + X3) / 3

        print(f"Iteration {t+1}/{max_iter} | Best Fitness: {Alpha_score:.6f}")

    return Alpha_pos, Alpha_score

best_position, best_score = grey_wolf_optimizer()
print("\nBest solution found:", best_position)
print("Best fitness value:", best_score)
```

Output:



```
Shreya Raj 1BM23CS317
Iteration 1/50 | Best Fitness: 1.827641
Iteration 2/50 | Best Fitness: 0.889744
Iteration 3/50 | Best Fitness: 0.587311
Iteration 4/50 | Best Fitness: 0.211439
Iteration 5/50 | Best Fitness: 0.005600
Iteration 6/50 | Best Fitness: 0.000891
Iteration 7/50 | Best Fitness: 0.000070
Iteration 8/50 | Best Fitness: 0.000045
Iteration 9/50 | Best Fitness: 0.000014
Iteration 10/50 | Best Fitness: 0.000001
Iteration 11/50 | Best Fitness: 0.000001
Iteration 12/50 | Best Fitness: 0.000000
Iteration 13/50 | Best Fitness: 0.000000
Iteration 14/50 | Best Fitness: 0.000000
Iteration 15/50 | Best Fitness: 0.000000
Iteration 16/50 | Best Fitness: 0.000000
Iteration 17/50 | Best Fitness: 0.000000
Iteration 18/50 | Best Fitness: 0.000000
Iteration 19/50 | Best Fitness: 0.000000
Iteration 20/50 | Best Fitness: 0.000000
Iteration 21/50 | Best Fitness: 0.000000
Iteration 22/50 | Best Fitness: 0.000000
Iteration 23/50 | Best Fitness: 0.000000
Iteration 24/50 | Best Fitness: 0.000000
Iteration 25/50 | Best Fitness: 0.000000
Iteration 26/50 | Best Fitness: 0.000000
Iteration 27/50 | Best Fitness: 0.000000
Iteration 28/50 | Best Fitness: 0.000000
Iteration 29/50 | Best Fitness: 0.000000
```



```
Iteration 29/50 | Best Fitness: 0.000000
Iteration 30/50 | Best Fitness: 0.000000
Iteration 31/50 | Best Fitness: 0.000000
Iteration 32/50 | Best Fitness: 0.000000
Iteration 33/50 | Best Fitness: 0.000000
Iteration 34/50 | Best Fitness: 0.000000
Iteration 35/50 | Best Fitness: 0.000000
Iteration 36/50 | Best Fitness: 0.000000
Iteration 37/50 | Best Fitness: 0.000000
Iteration 38/50 | Best Fitness: 0.000000
Iteration 39/50 | Best Fitness: 0.000000
Iteration 40/50 | Best Fitness: 0.000000
Iteration 41/50 | Best Fitness: 0.000000
Iteration 42/50 | Best Fitness: 0.000000
Iteration 43/50 | Best Fitness: 0.000000
Iteration 44/50 | Best Fitness: 0.000000
Iteration 45/50 | Best Fitness: 0.000000
Iteration 46/50 | Best Fitness: 0.000000
Iteration 47/50 | Best Fitness: 0.000000
Iteration 48/50 | Best Fitness: 0.000000
Iteration 49/50 | Best Fitness: 0.000000
Iteration 50/50 | Best Fitness: 0.000000

Best solution found: [5.98100037e-14 3.38310133e-14]
Best fitness value: 4.721774001251392e-27
```