```python
import numpy as np
import matplotlib.pyplot as plt

# Step 1: Define the Problem (a mathematical function to optimize)
def objective_function(x):
    return np.sum(x**2)  # Example: Sphere function (minimize sum of squares)

# Step 2: Initialize Parameters
num_wolves = 5  # Number of wolves in the pack
num_dimensions = 2  # Number of dimensions (for the optimization problem)
num_iterations = 100  # Number of iterations
lb = -10  # Lower bound of search space
ub = 10  # Upper bound of search space

# Step 3: Initialize Population (Generate initial positions randomly)
wolves = np.random.uniform(lb, ub, (num_wolves, num_dimensions))

# Initialize alpha, beta, delta wolves
alpha_pos = np.zeros(num_dimensions)
beta_pos = np.zeros(num_dimensions)
delta_pos = np.zeros(num_dimensions)

alpha_score = float('inf')  # Best (alpha) score
beta_score = float('inf')   # Second best (beta) score
delta_score = float('inf')  # Third best (delta) score

# To store the alpha score over iterations for graphing
alpha_score_history = []

# Step 4: Evaluate Fitness and assign Alpha, Beta, Delta wolves
def evaluate_fitness():
    global alpha_pos, beta_pos, delta_pos, alpha_score, beta_score, delta_score

    for wolf in wolves:
        fitness = objective_function(wolf)

        # Update Alpha, Beta, Delta wolves based on fitness
        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 = wolf.copy()
        elif fitness < beta_score:
            delta_score = beta_score
            delta_pos = beta_pos.copy()

            beta_score = fitness
            beta_pos = wolf.copy()
        elif fitness < delta_score:
            delta_score = fitness
            delta_pos = wolf.copy()

# Step 5: Update Positions
def update_positions(iteration):
    a = 2 - iteration * (2 / num_iterations)  # a decreases linearly from 2 to 0

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

            # Position update based on alpha
            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

            # Position update based on beta
            r1 = np.random.random()
            r2 = np.random.random()
            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

            # Position update based on delta
            r1 = np.random.random()
            r2 = np.random.random()
            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

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

            # Apply boundary constraints
            wolves[i, j] = np.clip(wolves[i, j], lb, ub)

# Step 6: Iterate (repeat evaluation and position updating)
for iteration in range(num_iterations):
    evaluate_fitness()  # Evaluate fitness of each wolf
    update_positions(iteration)  # Update positions based on alpha, beta, delta

    # Record the alpha score for this iteration
    alpha_score_history.append(alpha_score)

    # Optional: Print current best score
    print(f"Iteration {iteration+1}/{num_iterations}, Alpha Score: {alpha_score}")

# Step 7: Output the Best Solution
print("Best Solution:", alpha_pos)
print("Best Solution Fitness:", alpha_score)

# Plotting the convergence graph
plt.plot(alpha_score_history)
plt.title('Convergence of Grey Wolf Optimizer')
plt.xlabel('Iteration')
plt.ylabel('Alpha Fitness Score')
plt.grid(True)
plt.show()
```

```
Iteration 1/100, Alpha Score: 37.80280316995745
Iteration 2/100, Alpha Score: 17.222974015848603
Iteration 3/100, Alpha Score: 17.222974015848603
Iteration 4/100, Alpha Score: 7.277347771846025
Iteration 5/100, Alpha Score: 1.7778124914272804
Iteration 6/100, Alpha Score: 1.7778124914272804
Iteration 7/100, Alpha Score: 0.9538850298398175
Iteration 8/100, Alpha Score: 0.5128288223581519
Iteration 9/100, Alpha Score: 0.22134410277724406
Iteration 10/100, Alpha Score: 0.07636704819101871
Iteration 11/100, Alpha Score: 0.07636704819101871
Iteration 12/100, Alpha Score: 0.037697664739239904
Iteration 13/100, Alpha Score: 0.024031801537290558
Iteration 14/100, Alpha Score: 0.005988914644250488
Iteration 15/100, Alpha Score: 0.005054476899430858
Iteration 16/100, Alpha Score: 0.0026955904778654485
Iteration 17/100, Alpha Score: 0.0026955904778654485
Iteration 18/100, Alpha Score: 0.002367052840406579
Iteration 19/100, Alpha Score: 0.002367052840406579
Iteration 20/100, Alpha Score: 0.00141901463620052
Iteration 21/100, Alpha Score: 0.00141901463620052
Iteration 22/100, Alpha Score: 0.0014009812931264163
Iteration 23/100, Alpha Score: 0.0005049397768075314
Iteration 24/100, Alpha Score: 0.0002560222815757512
Iteration 25/100, Alpha Score: 0.0002560222815757512
Iteration 26/100, Alpha Score: 2.292543291423311e-05
Iteration 27/100, Alpha Score: 2.292543291423311e-05
Iteration 28/100, Alpha Score: 3.0553279773985726e-06
Iteration 29/100, Alpha Score: 3.0553279773985726e-06
Iteration 30/100, Alpha Score: 2.5861949155018664e-06
Iteration 31/100, Alpha Score: 2.543637375210279e-06
Iteration 32/100, Alpha Score: 2.239266657323426e-06
Iteration 33/100, Alpha Score: 1.2573747467434092e-06
Iteration 34/100, Alpha Score: 6.131192848997858e-07
Iteration 35/100, Alpha Score: 2.0141060469699903e-07
Iteration 36/100, Alpha Score: 2.0141060469699903e-07
Iteration 37/100, Alpha Score: 1.6041811079889588e-07
Iteration 38/100, Alpha Score: 5.130521341207807e-08
Iteration 39/100, Alpha Score: 5.130521341207807e-08
Iteration 40/100, Alpha Score: 5.130521341207807e-08
Iteration 41/100, Alpha Score: 2.2866962886494455e-08
Iteration 42/100, Alpha Score: 2.2866962886494455e-08
Iteration 43/100, Alpha Score: 1.325713878607976e-08
Iteration 44/100, Alpha Score: 1.2560119939260002e-08
Iteration 45/100, Alpha Score: 1.2043056759141376e-08
Iteration 46/100, Alpha Score: 8.919440061069608e-09
Iteration 47/100, Alpha Score: 8.058662312347624e-09
Iteration 48/100, Alpha Score: 3.545210657130187e-09
Iteration 49/100, Alpha Score: 1.025593631304138e-09
Iteration 50/100, Alpha Score: 1.025593631304138e-09
Iteration 51/100, Alpha Score: 6.445439535203631e-10
Iteration 52/100, Alpha Score: 3.171546494902945e-10
Iteration 53/100, Alpha Score: 2.942268153195744e-10
Iteration 54/100, Alpha Score: 2.942268153195744e-10
Iteration 55/100, Alpha Score: 1.922860220716823e-10
Iteration 56/100, Alpha Score: 1.922860220716823e-10
Iteration 57/100, Alpha Score: 1.606071592027489e-10
Iteration 58/100, Alpha Score: 1.193918695494025e-10
Iteration 59/100, Alpha Score: 4.0724096260106506e-11
Iteration 60/100, Alpha Score: 4.0724096260106506e-11
Iteration 61/100, Alpha Score: 2.7877330933001823e-11
Iteration 62/100, Alpha Score: 2.741734004070573e-11
Iteration 63/100, Alpha Score: 2.2120540460280262e-11
Iteration 64/100, Alpha Score: 2.004239967865558e-11
Iteration 65/100, Alpha Score: 1.6509330039311565e-11
Iteration 66/100, Alpha Score: 1.6509330039311565e-11
Iteration 67/100, Alpha Score: 1.6509330039311565e-11
Iteration 68/100, Alpha Score: 1.2351157442463385e-11
Iteration 69/100, Alpha Score: 1.1965572029721486e-11
Iteration 70/100, Alpha Score: 1.1727306216902872e-11
Iteration 71/100, Alpha Score: 9.391516608916525e-12
Iteration 72/100, Alpha Score: 8.89229264034819e-12
Iteration 73/100, Alpha Score: 7.976972376792823e-12
Iteration 74/100, Alpha Score: 7.781690786469272e-12
Iteration 75/100, Alpha Score: 7.781690786469272e-12
Iteration 76/100, Alpha Score: 6.876826733055706e-12
Iteration 77/100, Alpha Score: 5.614750344576231e-12
Iteration 78/100, Alpha Score: 5.614750344576231e-12
Iteration 79/100, Alpha Score: 4.443525611894963e-12
Iteration 80/100, Alpha Score: 4.4291097332990095e-12
Iteration 81/100, Alpha Score: 4.27816481109228e-12
Iteration 82/100, Alpha Score: 4.017763289759196e-12
Iteration 83/100, Alpha Score: 3.9604033994671554e-12
Iteration 84/100, Alpha Score: 3.714639765090552e-12
Iteration 85/100, Alpha Score: 3.2801754236886643e-12
Iteration 86/100, Alpha Score: 2.9570889433216277e-12
Iteration 87/100, Alpha Score: 2.9570889433216277e-12
Iteration 88/100, Alpha Score: 2.88517459706165e-12
Iteration 89/100, Alpha Score: 2.7943528546246316e-12
Iteration 90/100, Alpha Score: 2.5355464347920247e-12
Iteration 91/100, Alpha Score: 2.3991784645364 63e-12
Iteration 92/100, Alpha Score: 2.3813767863051294e-12
```

```
Iteration 93/100, Alpha Score: 2.2812645409647317e-12
Iteration 94/100, Alpha Score: 2.1799234914939267e-12
Iteration 95/100, Alpha Score: 2.0646808689642623e-12
Iteration 96/100, Alpha Score: 2.0646808689642623e-12
Iteration 97/100, Alpha Score: 2.0602785261591698e-12
Iteration 98/100, Alpha Score: 1.9913821821002276e-12
Iteration 99/100, Alpha Score: 1.9913821821002276e-12
Iteration 100/100, Alpha Score: 1.9913821821002276e-12
Best Solution: [-1.00873564e-06  9.86830574e-07]
Best Solution Fitness: 1.9913821821002276e-12
```