```
import numpy as np
import math
import random
import matplotlib.pyplot as plt


def initialization(N,Dim,UB,LB):
  X=np.zeros((N,Dim))
  for i in range(N):
    for j in range(Dim):
      X[i,j]=random.random()*(UB-LB)+LB
  return X
def fitness(x):
  fitval=x[0]   # will change beased on requirement
  return fitval


def AOA(N,M_Iter,LB,UB,Dim):
  # Two variables to keep the positions and the fitness value of the best-obtained solution
  Best_P=np.zeros((1,Dim))
  Best_FF=math.inf
  Conv_curve=[]
  # Initialize the positions of solution
  X=initialization(N,Dim,UB,LB)
  Xnew=X
  Ffun=np.zeros((1,X.shape[0])) # (fitness values)
  Ffun_new=np.zeros((1,Xnew.shape[0])) # (fitness values)

  MOP_Max=1;
  MOP_Min=0.2;
  C_Iter=1;
  Alpha=5;
  Mu=0.499;

  for i in range(X.shape[0]):
      Ffun=fitness(X[i,:])  # Calculate the fitness values of solutions
      if Ffun<Best_FF:
          Best_FF=Ffun
          Best_P=X[i,:]
  while C_Iter<M_Iter+1:  # Main loop
    MOP=1-((C_Iter)**(1/Alpha)/(M_Iter)**(1/Alpha));  # Probability Ratio
    MOA=MOP_Min+C_Iter*((MOP_Max-MOP_Min)/M_Iter); # Accelerated function

    # Update the Position of solutions
    for i in range(X.shape[0]):   # if each of the UB and LB has a just value
      for j in range(X.shape[1]):
        r1=random.random();
        if r1<MOA:
          # Exploration phase
           r2=random.random();
           if r2>0.5:
             Xnew[i,j]=Best_P[j]/(MOP+1)*((UB-LB)*Mu+LB)
           else:
            Xnew[i,j]=Best_P[j]*MOP*((UB-LB)*Mu+LB);
        else:
           # Exploitation phase
           r3=random.random();
           if r3>0.5:
             Xnew[i,j]=Best_P[j]-MOP*((UB-LB)*Mu+LB);
           else:
             Xnew[i,j]=Best_P[j]+MOP*((UB-LB)*Mu+LB);

      # Flag_UB=Xnew[i]>UB[0] # check if they exceed (up) the boundaries
      # Flag_LB=Xnew[i,:]<LB[0] # check if they exceed (down) the boundaries
      # Xnew[i,:]=(Xnew[i,:]*(~(Flag_UB+Flag_LB)))+UB*Flag_UB+LB*Flag_LB

      Ffun_new=fitness(Xnew[i,:]); # calculate Fitness function
      if Ffun_new<Ffun:
        X[i,:]=Xnew[i,:]
        Ffun=Ffun_new
      if Ffun<Best_FF:
        Best_FF=Ffun
        Best_P=X[i,:]
    Conv_curve.append(Best_FF)

    print('Iteration - ',str(C_Iter+1),': Best Position',str(Best_P),': Best Fitness',str("%.2f"%Best_FF))

    C_Iter=C_Iter+1;  # incremental iteration

  return [Best_FF,Best_P,Conv_curve]

Solution no=25 # Number of search solutions
```

```
M_Iter=100    # Maximum number of iterations
LB=0
UB=1
Dim=2
[Best_FF,Best_P,Conv_curve]=AOA(Solution_no,M_Iter,LB,UB,Dim) # call the AOA
```

```
Iteration -  2 : Best Position [-2.98733212  0.15004695] : Best Fitness -2.99
Iteration -  3 : Best Position [-5.15377035  0.32736025] : Best Fitness -5.15
Iteration -  4 : Best Position [-7.92058632 -0.0246788 ] : Best Fitness -7.92
Iteration -  5 : Best Position [-9.81556652 -0.08850248] : Best Fitness -9.82
Iteration -  6 : Best Position [-12.06465797  -0.48495566] : Best Fitness -12.06
Iteration -  7 : Best Position [-14.42668965   0.15570514] : Best Fitness -14.43
Iteration -  8 : Best Position [-16.07332627   0.39395128] : Best Fitness -16.07
Iteration -  9 : Best Position [-17.85437768  -0.26309005] : Best Fitness -17.85
Iteration -  10 : Best Position [-19.57083406   0.20569426] : Best Fitness -19.57
Iteration -  11 : Best Position [-21.04405234  -0.39967414] : Best Fitness -21.04
Iteration -  12 : Best Position [-21.93451773  -0.17605538] : Best Fitness -21.93
Iteration -  13 : Best Position [-23.65911468   0.17076056] : Best Fitness -23.66
Iteration -  14 : Best Position [-25.49820682  -0.48074566] : Best Fitness -25.50
Iteration -  15 : Best Position [-27.44503423  -0.09197575] : Best Fitness -27.45
Iteration -  16 : Best Position [-29.02059964   0.43778755] : Best Fitness -29.02
Iteration -  17 : Best Position [-30.85804831  -0.45793725] : Best Fitness -30.86
Iteration -  18 : Best Position [-32.0492598   0.1535518] : Best Fitness -32.05
Iteration -  19 : Best Position [-33.06339359   0.47786391] : Best Fitness -33.06
Iteration -  20 : Best Position [-3.47557078e+01 -2.18409540e-02] : Best Fitness -34.76
Iteration -  21 : Best Position [-35.1677126   -0.12147359] : Best Fitness -35.17
Iteration -  22 : Best Position [-36.773175     0.07130415] : Best Fitness -36.77
Iteration -  23 : Best Position [-37.94654774   0.06812021] : Best Fitness -37.95
Iteration -  24 : Best Position [-39.0902941   -0.12977758] : Best Fitness -39.09
Iteration -  25 : Best Position [-40.08152361   0.12133421] : Best Fitness -40.08
Iteration -  26 : Best Position [-41.28981077  -0.09091793] : Best Fitness -41.29
Iteration -  27 : Best Position [-42.468317    -0.09575381] : Best Fitness -42.47
Iteration -  28 : Best Position [-43.50298217   0.0544152 ] : Best Fitness -43.50
Iteration -  29 : Best Position [-44.17593801   0.27873381] : Best Fitness -44.18
Iteration -  30 : Best Position [-45.48915571   0.12161844] : Best Fitness -45.49
Iteration -  31 : Best Position [-46.45021585  -0.10669354] : Best Fitness -46.45
Iteration -  32 : Best Position [-47.3880507    0.10431519] : Best Fitness -47.39
Iteration -  33 : Best Position [-48.20156305   0.10169547] : Best Fitness -48.20
Iteration -  34 : Best Position [-4.90946899e+01 -8.56640457e-04] : Best Fitness -49.09
Iteration -  35 : Best Position [-49.86942878  -0.08100722] : Best Fitness -49.87
Iteration -  36 : Best Position [-5.08144695e+01 -9.95087268e-05] : Best Fitness -50.81
Iteration -  37 : Best Position [-51.36778132   0.09188831] : Best Fitness -51.37
Iteration -  38 : Best Position [-5.19076820e+01  5.05873644e-06] : Best Fitness -51.91
Iteration -  39 : Best Position [-52.17107016  -0.08779587] : Best Fitness -52.17
Iteration -  40 : Best Position [-52.85630423   0.0826336 ] : Best Fitness -52.86
Iteration -  41 : Best Position [-53.19052803  -0.05319393] : Best Fitness -53.19
Iteration -  42 : Best Position [-53.51652481  -0.07255022] : Best Fitness -53.52
Iteration -  43 : Best Position [-53.91393575   0.08919467] : Best Fitness -53.91
Iteration -  44 : Best Position [-54.22394873  -0.06190007] : Best Fitness -54.22
Iteration -  45 : Best Position [-5.49795565e+01  3.26591842e-02] : Best Fitness -54.98
Iteration -  46 : Best Position [-55.56878308   0.07410811] : Best Fitness -55.57
Iteration -  47 : Best Position [-55.99945992  -0.07158681] : Best Fitness -56.00
Iteration -  48 : Best Position [-56.27921166  -0.07702045] : Best Fitness -56.28
Iteration -  49 : Best Position [-56.75610405  -0.06776491] : Best Fitness -56.76
Iteration -  50 : Best Position [-5.71541858e+01 -1.69093143e-06] : Best Fitness -57.15
Iteration -  51 : Best Position [-5.76709479e+01  2.87104075e-02] : Best Fitness -57.67
Iteration -  52 : Best Position [-5.80481763e+01 -1.23376896e-02] : Best Fitness -58.05
Iteration -  53 : Best Position [-5.84763966e+01  3.69612813e-03] : Best Fitness -58.48
Iteration -  54 : Best Position [-5.88334158e+01 -3.11907536e-07] : Best Fitness -58.83
Iteration -  55 : Best Position [-5.90648442e+01  2.99363897e-03] : Best Fitness -59.06
Iteration -  56 : Best Position [-5.90086090e+01  1.68347897e-04] : Best Fitness -59.06
Iteration -  57 : Best Position [-2.65394288e+01  7.57153421e-05] : Best Fitness -59.06
Iteration -  58 : Best Position [-2.65924897e+01  3.41505704e-05] : Best Fitness -59.06
Iteration -  59 : Best Position [-1.20281037e+01  1.54467147e-05] : Best Fitness -59.06
```

```
Conv_curve=np.array(Conv_curve)
print("\nBest solution found:\n")
print('Best fitness :',Best_FF)
print('Best position :',Best_P)
plt.plot(Conv_curve)
plt.xlabel('Iteration')
plt.ylabel('fitness value')
plt.title('Convergence curve')
plt.show()
```

```
Best solution found:

Best fitness : -59.064844193054405
Best position : [5.00006080e-04 6.68432893e-14]
```