Defferential Evolution (Schwefel Func. 2.22)

```
import numpy as np
import random
from array import *
import matplotlib.pyplot as plt
import statistics as st
import math


D = 10 # Dimensi dari permasalahan
#maxit = 99 #Max iterasi
maxit = 495 #Max iterasi
ukuranPopulasi = 10
LB = -10 #Batas Bawah
UB = 10 #Batas Atas
CR = 0.9
F = 0.5
```

```
import random
def initPopulasi(ukuranPopulasi, D, UB, LB): #Inisialisasi setiap baris kolom
    P = np.empty((ukuranPopulasi,D))
    for i in range(ukuranPopulasi):
        for j in range(D):
            P[i][j]= (random.random()*(UB-LB))+LB
    return P

P = initPopulasi(ukuranPopulasi, D, UB, LB)
print(P)
```

```
[[-4.04301875  0.53884136  8.00538886 -3.18897556 -7.86131553 -1.33503336
  -1.82652845 -2.3703909  -6.09920286  5.43886746]
 [ 3.26844945 -2.57427516 -4.52919619 -4.28008403  0.0772137  -5.11631276
  -1.26177483 -3.16590835  1.52185851 -5.34046065]
 [ 3.93840673  5.98670356 -5.01539932  1.44669619 -9.52649048  9.85866649
   8.69204064  8.47641365  6.29342124  3.17994647]
 [ 4.59670209  2.72171256 -4.17595002  6.65078724  0.04273397 -0.24894272
  -2.05950487 -2.38839401 -5.87268808 -6.56585697]
 [ 3.16352711 -0.80191217  8.34511496  2.21464723  5.55000968  0.68912663
   6.6629669  -2.61657411  6.27443675 -4.8049696 ]
 [-8.81458304 -1.41652709  0.85303005 -5.8910459  -8.86689798  2.17723232
  -6.81872751 -5.33849818  8.44511107 -8.05558786]
 [-3.38133462  4.02782274  2.52367092 -4.38898674  5.03965094  3.28683238
  -6.32539395  6.36993378 -3.74536687 -7.49984454]
 [-3.11465176 -3.81676883  6.8295952   1.09234056  2.97402781  2.2807242
   1.27052054  1.4797594   2.26163101 -2.16090941]
 [ 2.17601568  7.8755492   9.59599919  0.12987194  4.05252536 -0.51013306
   3.92332676  1.74298907  4.46762636 -0.60371282]
 [ 8.50170775 -4.45343165 -2.78230591  3.12361584  4.53377593  8.03079509
   9.04568654  3.90613225 -2.63956582  8.6829121 ]]
```

```
def Fitness_Func(populasi):
    ukuran = populasi.shape
    ukuranPopulasi = ukuran[0]
    dimensi = ukuran[1]

    Fobj = np.empty(ukuranPopulasi)

    for i in range (ukuranPopulasi):
        d = 0
        for j in range (dimensi):
            #d = d + populasi[i][j]**2
            d = d + np.sum(np.abs(populasi[i][j])) + np.product(np.abs(populasi[i][j])) #s
            # d = d + np.sum(100*(populasi[i][j]+1)-(populasi[i][j]))**2 + (populasi[i][j]
            # d =  d + np.sum( (populasi[i][j]**2) - (10*math.cos(2*math.pi*populasi[i][j]
        Fobj[i] = d
    return Fobj

Fobj = Fitness_Func(P)
print(Fobj)
```

```
 [ 81.41512616  62.27106726 124.82836952  70.64654504  82.24657027
  113.354482    93.17767495  54.56185742  70.15549886 111.39985777]
```

```
def crossOver(populasi,F,j):
    ukuran = populasi.shape
    ukuranPopulasi = ukuran[0]
    Dimensi = ukuran[1]

    r1 = random.randint(0, ukuranPopulasi-1)
    r2 = random.randint(0, ukuranPopulasi-1)
    r3 = random.randint(0, ukuranPopulasi-1)

    while r1 == r2:
        r2 = (r2 + 1)%ukuranPopulasi

    while r3 == r1 or r3 == r2:
        r3 = (r3 + 1)%ukuranPopulasi

    v = P[r3][j] + F*(P[r1][j] - P[r2][j])
    return v

v = crossOver(P,F,1)
print(v)
```

```
 10.632741705609988
```

```
P = initPopulasi(ukuranPopulasi, D, UB, LB)
print("Inisialisasi Populasi :\n",P)
print(P)

Fobj = Fitness_Func(P)
print(Fobj)
```

```
    print("\nFungsi Objective :\n",Fobj)

    U = np.empty((1,D))
    bestFobj = np.empty((maxit+1))
    bestFobj[0] = Fobj.min()

    for it in range(maxit):
        for i in range(ukuranPopulasi):
            for j in range(D):
                U[0][j] = P[i][j]

            jrand=random.randint(0,D)

            for j in range(D):
                if random.random() < CR or j == jrand:
                    v = crossOver(P,F,j)
                    U[0][j] = v

            FobjU = Fitness_Func(U)

            if FobjU < Fobj[i]:
                Fobj[i] = FobjU
                for j in range(D):
                    P[i][j] = U[0][j]

        bestFobj[it+1] = Fobj.min()

        print("\nNilai Optimal : \n",bestFobj)

        for i in range(30):
            bestFobj
```

```
  9.34427137e-05 8.18760683e-05 8.18760683e-05 8.18760683e-05
  6.20741709e-05 6.20741709e-05 5.83182966e-05 5.83182966e-05
  3.45171810e-05 3.45171810e-05 3.45171810e-05 2.44959076e-05
  2.35304804e-05 2.35304804e-05 1.49241608e-05 1.49241608e-05
  1.49241608e-05 1.49241608e-05 1.49241608e-05 1.49241608e-05
  1.49241608e-05 1.49241608e-05 1.49241608e-05 1.49241608e-05
  1.49241608e-05 1.49241608e-05 1.06051146e-05 1.06051146e-05
  1.06051146e-05 1.06051146e-05 1.06051146e-05 1.03260298e-05
  1.03260298e-05 5.60773710e-06 5.60773710e-06 5.60773710e-06
  5.60773710e-06 5.60773710e-06 5.59497051e-06 5.59497051e-06
  5.43082478e-06 4.21557408e-06 4.04045984e-06 4.04045984e-06
  2.62848647e-06 2.62848647e-06 2.62848647e-06 2.62848647e-06
  2.26387683e-06 2.26387683e-06 2.26387683e-06 1.91623375e-06
  1.64870827e-06 1.64870827e-06 1.64870827e-06 1.34915320e-06
  1.34915320e-06 1.34915320e-06 1.34915320e-06 1.34915320e-06
  1.34915320e-06 1.34915320e-06 8.96715243e-07 8.96715243e-07
  8.96715243e-07 8.96715243e-07 8.96715243e-07 6.51112549e-07
  5.08752856e-07 4.25847853e-07 4.25847853e-07 4.25847853e-07
  4.25847853e-07 4.25847853e-07 4.25847853e-07 3.97564291e-07
  3.97564291e-07 2.80754778e-07 1.99690762e-07 1.99690762e-07
  1.99690762e-07 1.71965660e-07 1.71965660e-07 1.65795730e-07
  1.46518114e-07 1.46518114e-07 1.10089293e-07 1.10089293e-07
  1.10089293e-07 1.10089293e-07 1.10089293e-07 9.26188178e-08
  9.26188178e-08 9.26188178e-08 5.56623318e-08 5.56623318e-08
  5.56623318e-08 5.56623318e-08 5.56623318e-08 4.89979233e-08
  2.71065090e-08 2.71065090e-08 2.71065090e-08 2.71065090e-08
```

```
2.71065099e-08 2.71065099e-08 2.71065099e-08 2.71065099e-08
2.71065099e-08 2.71065099e-08 2.71065099e-08 2.21822048e-08
2.05507286e-08 1.96595203e-08 1.77442341e-08 1.77442341e-08
1.77442341e-08 1.47381667e-08 1.47381667e-08 1.47381667e-08
1.36449447e-08 8.18477746e-09 8.18477746e-09 8.18477746e-09
8.18477746e-09 8.18477746e-09 7.33737469e-09 6.10886479e-09
6.05956661e-09 6.05956661e-09 6.05956661e-09 3.61287171e-09
3.61287171e-09 3.61287171e-09 3.61287171e-09 3.32336673e-09
3.32336673e-09 3.21904335e-09 2.64952222e-09 2.64952222e-09
2.64952222e-09 2.64952222e-09 2.64952222e-09 1.47368215e-09
1.47368215e-09 1.47368215e-09 1.47368215e-09 1.47368215e-09
1.47368215e-09 1.47368215e-09 1.47368215e-09 1.47368215e-09
1.37965671e-09 1.35723167e-09 9.90022370e-10 9.90022370e-10
9.90022370e-10 9.90022370e-10 9.56396424e-10 9.49356522e-10
8.02135465e-10 6.28112576e-10 6.28112576e-10 6.28112576e-10
3.91273824e-10 3.66927813e-10 3.66927813e-10 3.66927813e-10
3.66927813e-10 3.66927813e-10 3.66927813e-10 3.35723627e-10
3.35723627e-10 2.93260698e-10 2.93260698e-10 2.93260698e-10
2.93260698e-10 2.93260698e-10 2.93260698e-10 2.51498876e-10
1.97697402e-10 1.97697402e-10 1.97697402e-10 1.97697402e-10
1.97697402e-10 1.69538636e-10 1.69538636e-10 1.69538636e-10
1.46427257e-10 1.40771051e-10 1.19312924e-10 6.10686869e-11
6.10686869e-11 6.10686869e-11 6.10686869e-11 6.10686869e-11
6.10686869e-11 6.10686869e-11 6.10686869e-11 6.10686869e-11
6.10686869e-11 6.10686869e-11 5.51541835e-11 4.32770856e-11
4.32770856e-11 3.93848802e-11 3.93848802e-11 3.07441428e-11
3.07441428e-11 2.34374542e-11 2.34374542e-11 2.34374542e-11
2.34374542e-11 1.94750744e-11 1.94750744e-11 1.49687313e-11
1.49687313e-11 1.33218360e-11 1.33218360e-11 8.67870253e-12
8.67870253e-12 8.67870253e-12 7.82852902e-12 7.82852902e-12
7.44336002e-12 7.44336002e-12 7.44336002e-12 7.44336002e-12
5.61664143e-12 5.28599158e-12 5.28599158e-12 3.78225993e-12
3.78225993e-12 3.78225993e-12 3.78225993e-12 3.21910287e-12
3.21910287e-12 3.21910287e-12 3.21910287e-12 3.15402471e-12
```

```
import statistics as st
print("Nilai Mean : ",st.mean(bestFobj))
print("Nilai Standard Deviation : ",st.stdev(bestFobj))
print("Nilai Minimal : ",np.min(bestFobj))
```

```
Nilai Mean :  2.1820240155997133
Nilai Standard Deviation :  9.959848743443919
Nilai Minimal :  5.620433788795094e-15
```

```
x = np.linspace(0, 1, maxit+1)
plt.plot(x, bestFobj, label= 'f=Fobj')
plt.show()
```

Genetic Algorithm


```
import numpy as np
import random
from array import *
import matplotlib.pyplot as plt
import statistics as st
import math


N = 3
rows, cols = (N, 4)
induk = [[0 for i in range(cols)] for j in range(rows)]
print (induk)

N_Anak = 6
rows, cols = (N_Anak, 4)
anak = [[0 for i in range(cols)] for j in range(rows)]
print(anak)
```

```
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

```
#Inisialiasi Parameter
Pc = 0.9
Pm = 0.1

for i in range(N):
    a = random.randint(0, 15)
    induk[i] = [int (x) for x in '{:04b}'.format(a)]
    print(a, induk[i])
```

```
 0 [0, 0, 0, 0]
 0 [0, 0, 0, 0]
 11 [1, 0, 1, 1]
```

```
#Function cross over
def crossover(ind1=[], ind2=[]):
    anak1 = [0, 0, 0, 0]
    anak2 = [0, 0, 0, 0]
```

```
    anak1[0] = ind1[0]
    anak1[1] = ind1[1]
    anak1[2] = ind2[2]
    anak1[3] = ind2[3]

    anak2[0] = ind2[0]
    anak2[1] = ind2[1]
    anak2[2] = ind1[2]
    anak2[3] = ind1[3]

    return anak1, anak2



#Function  mutasi
def mutasi(ind=[]):
    ind[1] = 1 - ind[1]
    return ind



#Function hitung Int
def hitungInt(ind = []):
    a = ind[3] * 1
    a = a + ind[2] * 2
    a = a + ind[1] * 4
    a = a + ind[0] * 8
    return a



#Buat function untuk melakukan hitung fitness
def hitungFitness(ind = []):
    a = hitungInt(ind)
    #y = a**2 #sphere
    y = np.sum(np.abs(a)) + np.product(np.abs(a)) #schwefel 2.22
    #y = np.sum(100*(a+1)-(a))**2 + (a-1)**2 #Generalize rosenbrock
    # y = np.sum( (a**2) - (10*math.cos(2*math.pi*a)) + 10 ) #rastrigins

    return y


i = 1
epochs = 495
#epochs = 99
MaxFitness = [0 for i in range(epochs)]
angkaFitness = [0 for i in range(epochs)]

#nilai fitness induk
for j in range(N):
    if (hitungFitness(induk[j])>MaxFitness[0]):
        MaxFitness[0] = hitungFitness(induk[j])



#'epochs' kali
```

```
  # epochs = kali
  for i in range(30):
    while i < epochs:
      print("Iterasi: ", i+1)
      #Reproduksi
      a = random.random()
      if (a<Pc):
          #Cross over
          anak[0], anak[1] = crossover(induk[0], induk[1])
          anak[2], anak[3] = crossover(induk[0], induk[2])
          anak[4], anak[5] = crossover(induk[1], induk[2])
      else:
          #Mutasi
          for j in range(N):
              induk [j] = mutasi(induk[j])
      #Elistism
      minFitness = 10
      idx = 0
      idxanak = 0
      #minFitness (induk terburuk)
      for j in range(len(induk)):
          if (hitungFitness(induk[j])<minFitness):
              minFitness = hitungFitness(induk[j])
              idx = j

      #cari maxFit (Anak terbaik)
      maxFit = -10
      for j in range(len(anak)):
          if (hitungFitness(anak[j])>maxFit):
              maxFit = hitungFitness(anak[j])
              idxanak = j
      #Individual replacement
      if (minFitness < maxFit):
          induk[idx] = anak[idxanak]

      #Populasi induk yang paling baik
      for j in range(N):
          if (hitungFitness(induk[j])>MaxFitness[i]):
              MaxFitness[i] = hitungFitness(induk[j])
      i+=1

      print(MaxFitness)
```

```
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  386
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  387
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  388
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  389
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  390
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  391
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  392
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
```

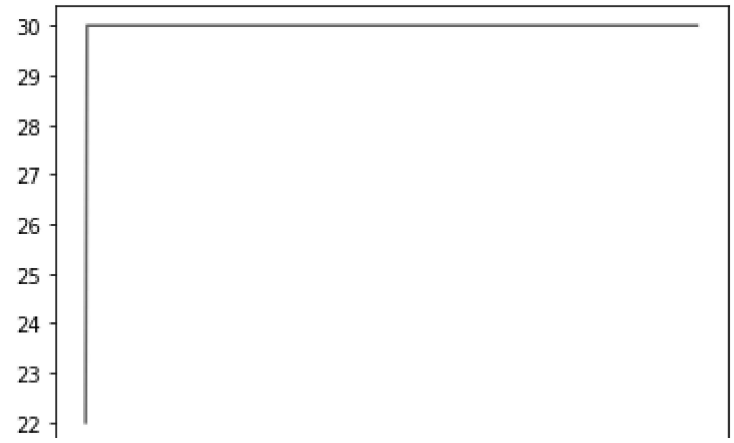
```
Iterasi:  393
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  394
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  395
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  396
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  397
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  398
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  399
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  400
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  401
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  402
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  403
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  404
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  405
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  406
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  407
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  408
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  409
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  410
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  411
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  412
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  413
[22, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
Iterasi:  414
```

```
print("Nilai Mean : ", st.mean(MaxFitness))
print("Nilai Standard Deviation : ", st.stdev(MaxFitness))
print("Nilai Minimal : ",np.min(MaxFitness))
```

```
Nilai Mean :  29
Nilai Standard Deviation :  0.0
Nilai Minimal :  22
```

```
x = [i for i in range(epochs)]
plt.plot(x, MaxFitness)
```

```
[<matplotlib.lines.Line2D at 0x7fbf2dc6d310>]
```



✓ 0s completed at 7:33 AM