Divisi Komputer
Himpunan Mahasiswa Elektroteknik
Institut Teknologi Bandung
Basement Labtek 8, ITB
Jalan Ganesha No. 10, Bandung 40132
Email: hme.divkom@gmail.com
Website: hme.ee.itb.ac.id/divisi-komputer

01 Mei 2019

Nama Proyek : Prediksi Harga Saham dengan Algoritma Genetika

Waktu Pengerjaan : 19 April 2019 – 01 Mei 2019

DESKRIPSI

Dalam mengimplementasikan algoritma genetika, terdapat beberapa bagian, yaitu inisiasi populasi yang terdiri dari empat buah gen menggunakan *real-coded genetic algorithm,* evaluasi dengan menghitung *fitness* dan *error*, *crossover* menggunakan *extended intermediate*, mutasi menggunakan *random mutation*, seleksi dan regenerasi menggunakan *replacement selection*, dan membuat populasi baru dari hasil seleksi dan regenerasi.

IMPLEMENTASI FUNGSI

1. Fungsi create_data
   Membuat dan menyimpan data hasil prediksi saham ke dalam format .xlsx.

```
function data = create_data
        T = readtable('proyek.xlsx', 'Sheet', 1, 'Range', 'A1:E11');
        Yt = T.Yt;
        Yt_1 = T.Yt_1;
        Yt_2 = T.Yt_2;
        Et = T.Et;
        Et_1 = T.Et_1;
        data = T;
end
```

2. Fungsi create_population
   Menginisiasi populasi yang terdiri dari empat buah gen. Di dalam fungsi ini terdapat fungsi create_data, create_genes, dan calculate_fitness. Jumlah populasi bergantung dari *user*.

```
function population = create_population(population_size)
        data = create_data;
        population = struct.empty(population_size,0);
        for i=1:population_size
                genes = create_genes;
                fitness = calculate_fitness(genes,data);
                population(i).genes = genes;
                population(i).fitness = fitness;
        end
end
```

3. Fungsi create_genes
   Membentuk nilai gen menggunakan nilai *random* pada interval [-100,100]. Hasil nilai *random* tersebut adalah koefisien regresi. Tujuan pembentukan koefisien regresi adalah untuk mendapatkan hasil prediksi yang akurat.

```
function genes = create_genes
        a = -100;
        b = 100;
        x = a + (b-a) * rand(1,4);
        genes =(x);
end
```

4. Fungsi calculate_fitness

   Menghitung nilai *fitness* dengan menghitung prediksi harga saham dengan fungsi regresi serta menghitung *error*.

```
function fitness = calculate_fitness(genes,data)
   e = 0;
    for i = 1:10
      y = genes(1)+(1+genes(2))*data.Yt_1(i)-genes(3)*data.Yt_2(i)+data.Et(i)-genes(4)*data.Et_1(i);
      yi = data.Yt(i);
      e = e + immse(y,yi);
    end
    error = sqrt(e)/10;
    fitness = 1/error;
end
```

5. Fungsi crossover

   Menghasilkan individu atau populasi baru (anak dari populasi awal) dengan gen yang berbeda dari populasi awal. Banyaknya individu yang dihasilkan bergantung pada nilai *crossover rate* (*cr*) dari *user*.

```
function pops_co = crossover(parent1,parent2,cr,banyak_pops)
  data = create_data;
  child1 = parent1;
  child2 = parent2;

  %tentukan banyak anak
  many_anak = 2*(round(cr*banyak_pops)/2);
  counter_anak = 0;

  %buat anak
  if (many_anak==0)
    alpha= rand(1,1);
    for j=1:4
      child1.genes(j) = child1.genes(j) + alpha*(child2.genes(j)- child1.genes(j));
    end
    pops_co.genes = child1.genes;
    pops_co.fitness = calculate_fitness(pops_co.genes,data);
  else
    for i=1:(many_anak/2)
      alpha= rand(1,1);
      for j=1:4
        child1.genes(j) = child1.genes(j) + alpha*(child2.genes(j)- child1.genes(j));
      end

      counter_anak= counter_anak+1;
      pops_co(counter_anak).genes = child1.genes;
      pops_co(counter_anak).fitness = calculate_fitness(pops_co(counter_anak).genes,data);

      alpha= rand(1,1);
      for j=1:4
        child2.genes(j)= child2.genes(j) + alpha*(child1.genes(j)- child2.genes(j));
      end

      counter_anak = counter_anak+1;
      pops_co(counter_anak).genes = child2.genes;
      pops_co(counter_anak).fitness = calculate_fitness(pops_co(counter_anak).genes,data);
  end

end
```

6. Fungsi mutation

   Menghasilkan individu atau populasi baru dengan gen yang berbeda dari hasil *crossover*. Populasi atau *parent* dipilih secara *random*. Banyaknya individu yang dihasilkan bergantung pada nilai *mutation rate* (*mr*) dari *user*.

```
function[pops_mu] = mutation(pops,mutation_rate)
  data = create_data;

  %Tentukan banyak mutan
  many_mutant= round(length(pops)*mutation_rate);
  if (many_mutant==0)
    many_mutant=1;
  end

  mutant.genes(1:4) = 0;
  counter_mutant=0;

  for i=1:many_mutant
    % Pilih parent
    idx_parent= randi([1,length(pops)], 1, 'single');
    parent= pops(idx_parent);

    for j=1:4
      % Pilih r
      r= rand(1,1,'single');
      % Buat mutan
      mutant.genes(j)= parent.genes(j)+ r*200;
    end

    counter_mutant= counter_mutant+1;
    pops_mu(counter_mutant).genes= mutant.genes;
    pops_mu(counter_mutant).fitness = calculate_fitness(mutant.genes,data);
  end

end
```

7. Fungsi selection

   Menjamin populasi yang terbaik selalu lolos dengan memperhatikan nilai *fitness*.

```
function [best1, best2] = selection(population)

        fitness = zeros(1,length(population));
        for i=1:length(population)
            fitness(i) = population(i).fitness;
        end

        [~,index] = max(fitness);
        best1 = population(index);

        population(index) = [];
        fitness(index) = [];

        [~,index] = max(fitness);
        best2 = population(index);

end
```

8. Fungsi regeneration

   Membuat populasi baru dari hasil selection.

```
function [new_population, maxf, idx_max] = regeneration(pops_co,pops_mu,population)
    data = create_data;
    fitness = zeros(1,(length(pops_co)+ length(pops_mu)));
    fitness_pops = zeros(1,(length(population)));
    batas = length(pops_co);

    %-----------------------------------------------
    %Isi array pops
    for i=1:length(population)
        fitness_pops= population(i).fitness;
    end
```

```
    %-------------------------------------------------
    % Isi array fitness
    for i=1:batas
        fitness(i) = calculate_fitness(pops_co(i).genes,data);
    end

    for i= (batas+1) : length(fitness)
        fitness(i) = calculate_fitness(pops_mu(i- batas).genes,data);
    end

    %-------------------------------------------------
    %Tentukan populasi baru dari yang terbaik
    for i=1:batas
        [~,index] = min(fitness_pops);
        if (fitness(i)> population(index).fitness)
            %Berarti ganti populasi terjelek dengan anaknya
            population(index)= pops_co(i);
            fitness_pops(index)= pops_co(i).fitness;
        end
    end

    for i=(batas+1): length(fitness)
        [~,index] = min(fitness_pops);
        if (fitness(i)> population(index).fitness)
            %Berarti ganti populasi terjelek dengan anaknya
            population(index)= pops_mu(i- batas);
            fitness_pops(index)= pops_mu(i- batas).fitness;
        end
    end

    new_population= population;
    [maxf, idx_max] = max(fitness_pops);

end
```

9. Main Program

   Implementasi seluruh fungsi

```
clc
clear
warning('off','MATLAB:table:ModifiedVarnames');

p = input('jumlah populasi (integer) : ');
mr = input('CrossOver Rate (0-1) : ');
cr = input('mutation rate (0-1) : ');

maxc = -1000;
population = create_population(p);
for i = 1:10
    [best1, best2] = selection(population);
    pops_co = crossover(best1,best2,cr,p);
    pops_mu = mutation(population,mr);
    [population, max, idx_max] = regeneration(pops_co,pops_mu,population);
    if(max>maxc)
        maxc=max;
    end
    disp(maxc);
    save(i)=max;
    savemax(i)=maxc;
end

disp(population(idx_max));

subplot(1,2,1);
plot(save), xlabel('Jumlah Loop'),ylabel('Fitness tiap loop');
subplot(1,2,2);
plot(savemax), xlabel('Jumlah Loop'),ylabel('Fitness maksimum');
```

HASIL PENGUJIAN

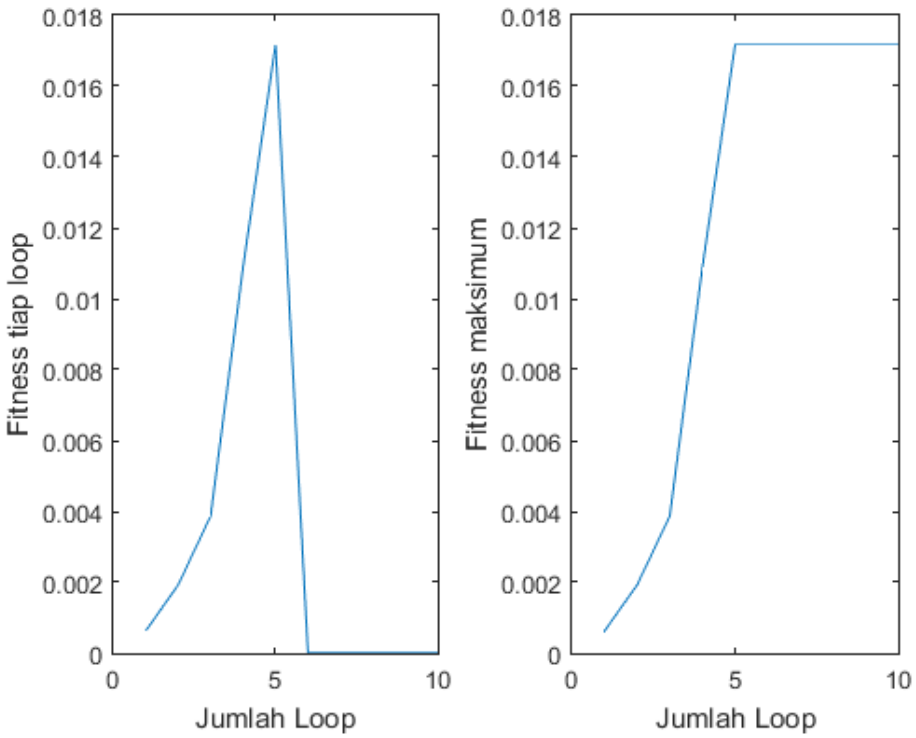
Setiap hasil uji coba memiliki *workspace*, *command window*, grafik. *Workspace* berisi seluruh variabel yang ada di dalam program. *Command window* berisi masukan *user*, hasil *fitness* maksimum di setiap *loop*, hasil *genes* terbaik, dan nilai *fitness* terbaik. Grafik terdiri dari dua bagian, yaitu grafik *fitness* terbaik dari setiap *loop* dan grafik *fitness* maksimum dari setiap *loop*.

1. Uji Coba Populasi
   Ukuran populasi yang diujikan adalah pada interval [200, 1400] dengan kelipatan 200. Kombinasi *crossover rate* (*cr*) dan *mutation rate* (*mr*) yang digunakan adalah 0.5 : 0.5. Masing-masing percobaan diulang 10 kali.
   a. Populasi 200

```
jumlah populasi (integer) : 200
CrossOver Rate (0-1) : 0.5
mutation rate (0-1) : 0.5
   6.2724e-04

    0.0019

    0.0039

    0.0109

    0.0172

    0.0172

    0.0172

    0.0172

    0.0172

    0.0172

  genes: [30.2108 -2.2802 -2.2667 -0.6574]
fitness: 0.0172
```

MATLAB Workspace
May 1, 2019

Name	Value
best1	*1x1 struct*
best2	*1x1 struct*
cr	0.5000
i	10
idx_max	1
max	4.6523e-06
maxc	0.0172
mr	0.5000
p	200
pops_co	*1x100 struct*
pops_mu	*1x100 struct*
population	*1x200 struct*
save	[6.2724e-04,0.0019,0.0039,0.0109,0.0...
savemax	[6.2724e-04,0.0019,0.0039,0.0109,0.0...

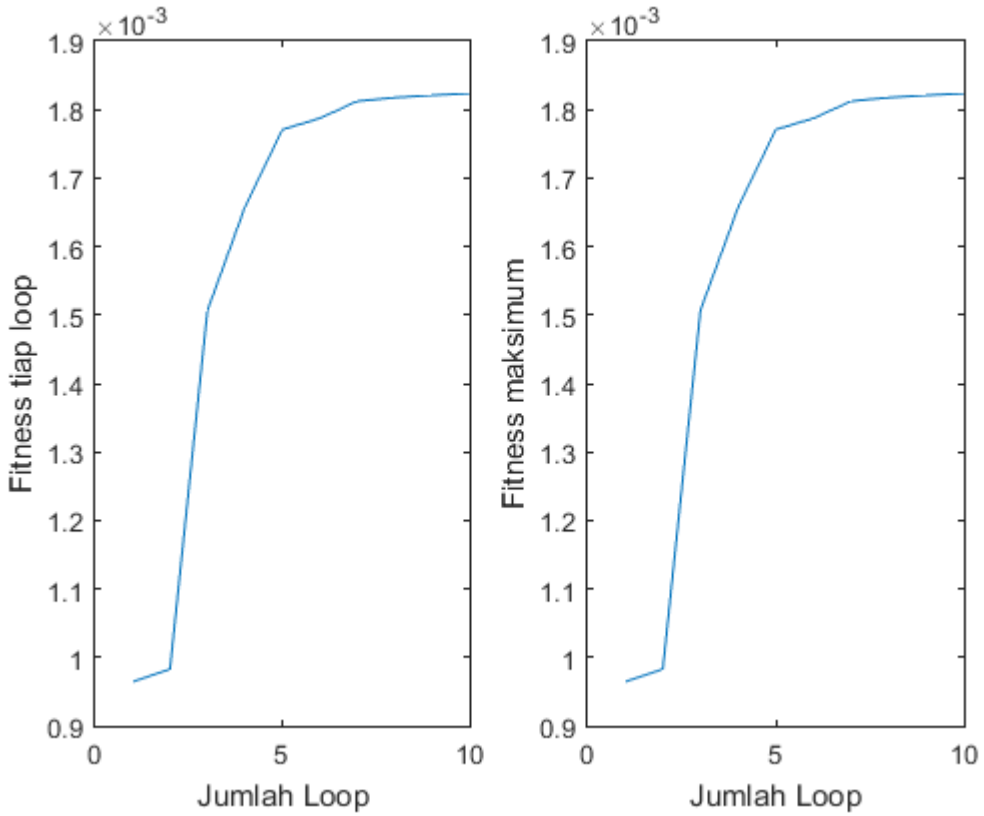


b. Populasi 400

```
jumlah populasi (integer) : 400
CrossOver Rate (0-1) : 0.5
mutation rate (0-1) : 0.5
   9.6427e-04

   9.8275e-04

    0.0015

    0.0017

    0.0018

    0.0018

    0.0018

    0.0018

    0.0018

    0.0018

  genes: [66.9425 18.8228 18.8090 23.7691]
fitness: 0.0018
```

MATLAB Workspace
May 1, 2019

Name	Value
best1	*1x1 struct*
best2	*1x1 struct*
cr	0.5000
i	10
idx_max	1
max	0.0018
maxc	0.0018
mr	0.5000
p	400
pops_co	*1x200 struct*
pops_mu	*1x200 struct*
population	*1x400 struct*
save	[9.6427e-04,9.8275e-04,0.0015,0.001...
savemax	[9.6427e-04,9.8275e-04,0.0015,0.001...

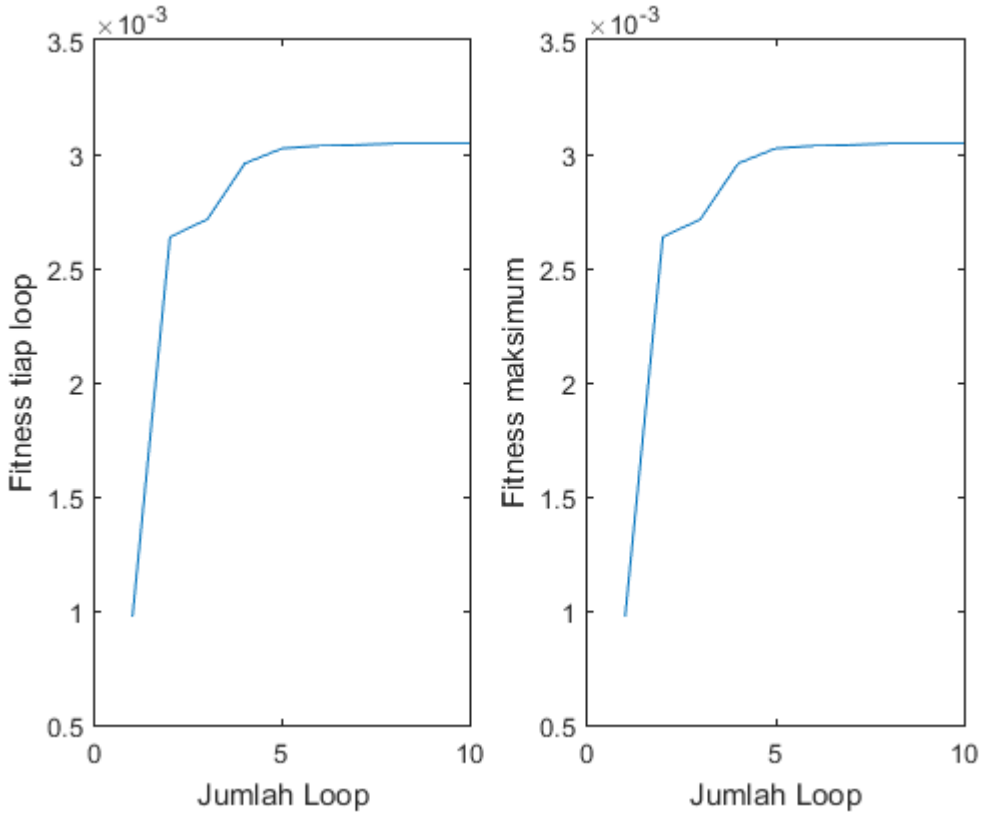


c. Populasi 600

```
jumlah populasi (integer) : 600
CrossOver Rate (0-1) : 0.5
mutation rate (0-1) : 0.5
   9.7767e-04

    0.0026

    0.0027

    0.0030

    0.0030

    0.0030

    0.0030

    0.0030

    0.0030

    0.0030

  genes: [-21.2528 -12.3760 -12.3495 -57.8541]
fitness: 0.0030
```

MATLAB Workspace
Apr 30, 2019

Name	Value
best1	*1x1 struct*
best2	*1x1 struct*
cr	0.5000
i	10
idx_max	1
max	0.0030
maxc	0.0030
mr	0.5000
p	600
pops_co	*1x300 struct*
pops_mu	*1x300 struct*
population	*1x600 struct*
save	[9.7767e-04,0.0026,0.0027,0.0030,0.0...
savemax	[9.7767e-04,0.0026,0.0027,0.0030,0.0...

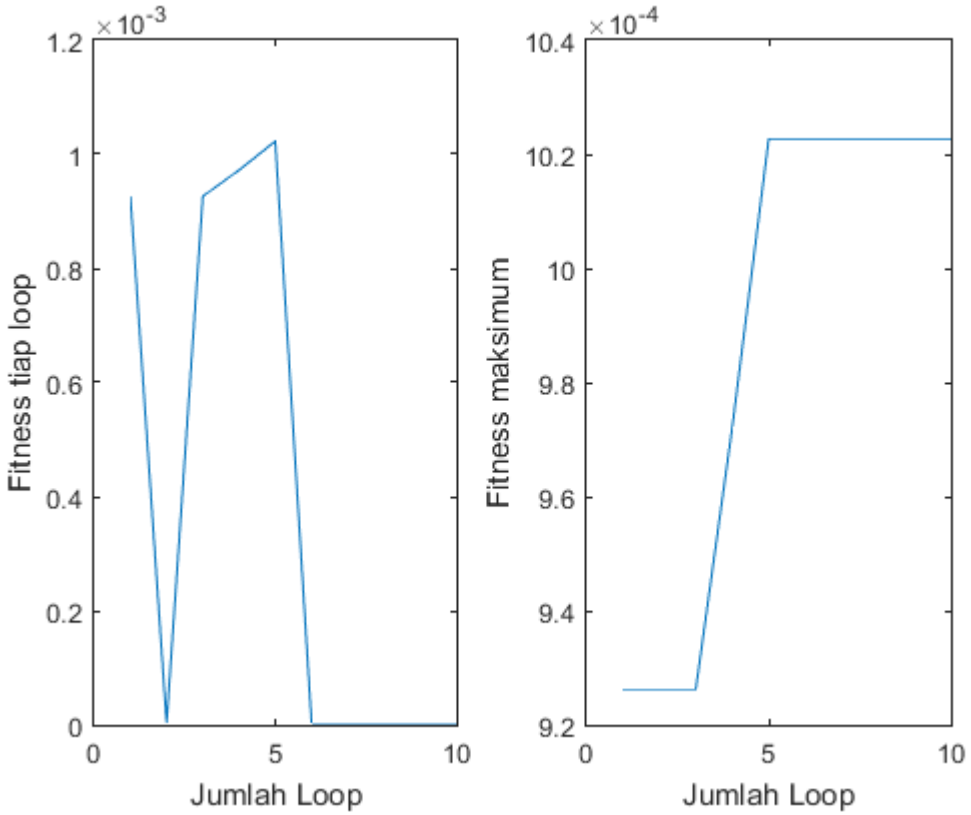


d. Populasi 800

```
jumlah populasi (integer) : 800
CrossOver Rate (0-1) : 0.5
mutation rate (0-1) : 0.5
   9.2628e-04

   9.2628e-04

   9.2629e-04

   9.7218e-04

    0.0010

    0.0010

    0.0010

    0.0010

    0.0010

    0.0010

     genes: [26.0078 33.4748 33.4464 13.3871]
   fitness: 0.0010
```

MATLAB Workspace
May 1, 2019

Name	Value
best1	*1x1 struct*
best2	*1x1 struct*
cr	0.5000
i	10
idx_max	1
max	3.2930e-06
maxc	0.0010
mr	0.5000
p	800
pops_co	*1x400 struct*
pops_mu	*1x400 struct*
population	*1x800 struct*
save	[9.2628e-04,3.2930e-06,9.2629e-04,9...
savemax	[9.2628e-04,9.2628e-04,9.2629e-04,9...

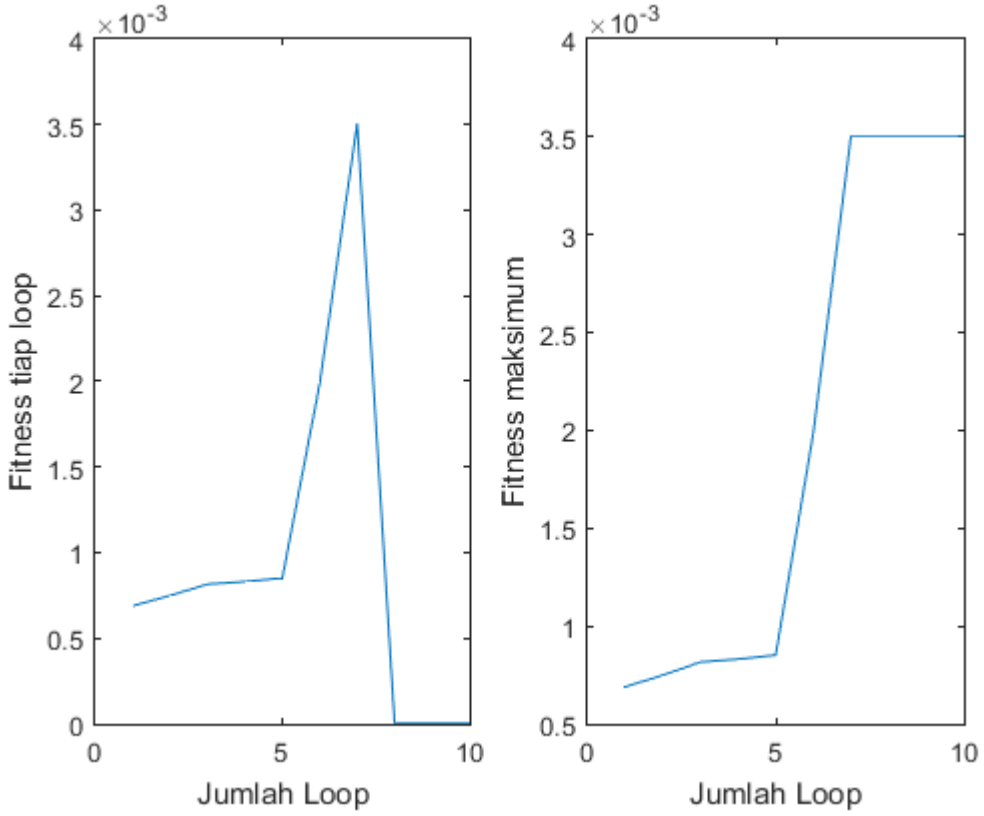


e. Populasi 1000

```
jumlah populasi (integer) : 1000
CrossOver Rate (0-1) : 0.5
mutation rate (0-1) : 0.5
   6.8897e-04

   7.4902e-04

   8.1580e-04

   8.3064e-04

   8.5159e-04

    0.0020

    0.0035

    0.0035

    0.0035

    0.0035

      genes: [31.6749 10.4565 10.4359 78.7185]
    fitness: 0.0035
```

MATLAB Workspace
May 1, 2019

Name	Value
best1	*1x1 struct*
best2	*1x1 struct*
cr	0.5000
i	10
idx_max	1
max	2.2825e-06
maxc	0.0035
mr	0.5000
p	1000
pops_co	*1x500 struct*
pops_mu	*1x500 struct*
population	*1x1000 struct*
save	[6.8897e-04,7.4902e-04,8.1580e-04,8...
savemax	[6.8897e-04,7.4902e-04,8.1580e-04,8...

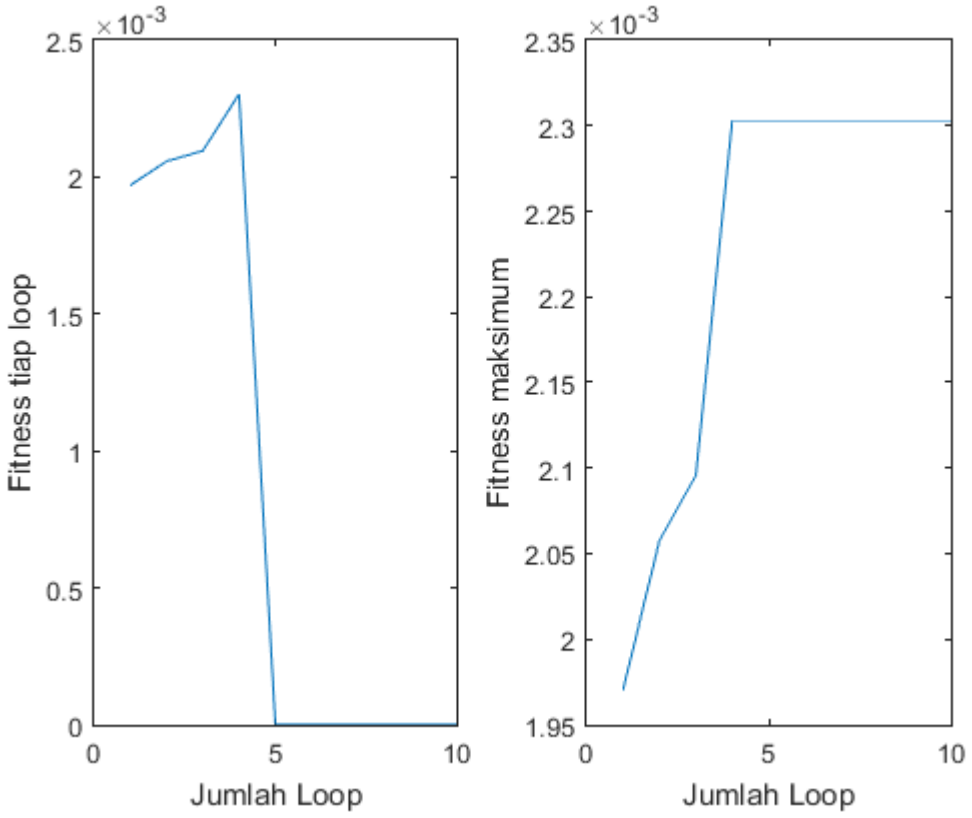


f. Populasi 1200

```
jumlah populasi (integer) : 1200
CrossOver Rate (0-1) : 0.5
mutation rate (0-1) : 0.5
    0.0020

    0.0021

    0.0021

    0.0023

    0.0023

    0.0023

    0.0023

    0.0023

    0.0023

    0.0023

  genes: [-44.0629 15.4528 15.4269 40.5965]
fitness: 0.0023
```

MATLAB Workspace
May 1, 2019

Name	Value
best1	*1x1 struct*
best2	*1x1 struct*
cr	0.5000
i	10
idx_max	1
max	6.1886e-06
maxc	0.0023
mr	0.5000
p	1200
pops_co	*1x600 struct*
pops_mu	*1x600 struct*
population	*1x1200 struct*
save	[0.0020,0.0021,0.0021,0.0023,6.1886e...
savemax	[0.0020,0.0021,0.0021,0.0023,0.0023,...

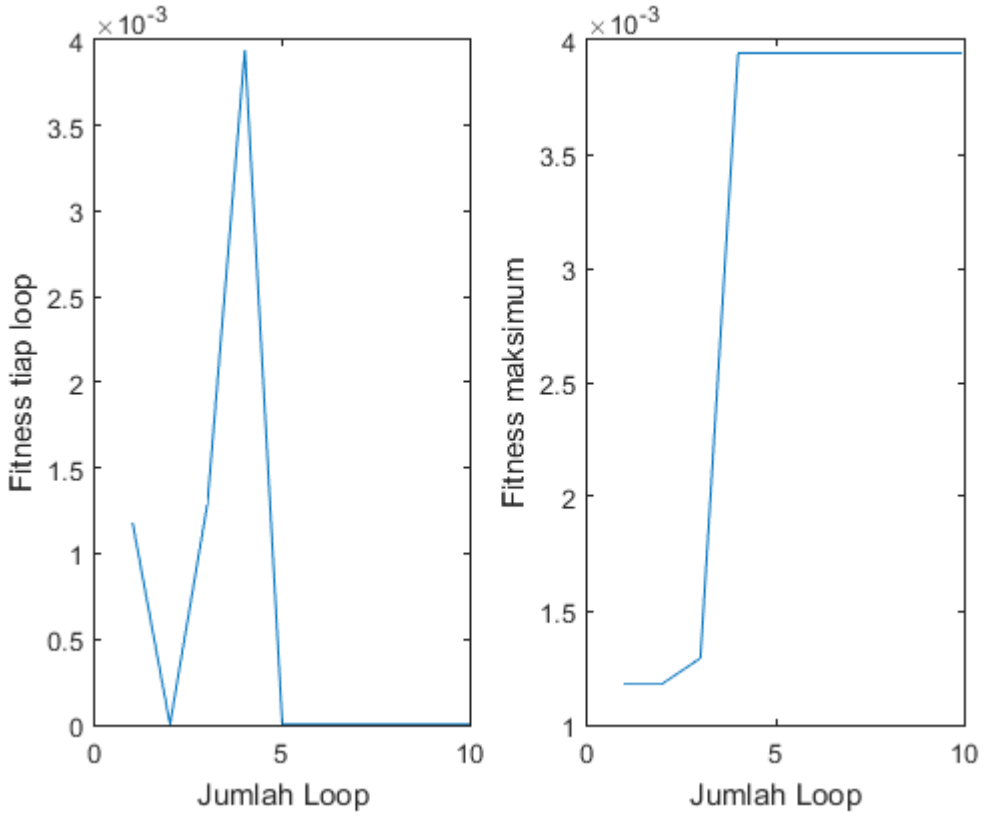


g. Populasi 1400

```
jumlah populasi (integer) : 1400
CrossOver Rate (0-1) : 0.5
mutation rate (0-1) : 0.5
    0.0012

    0.0012

    0.0013

    0.0039

    0.0039

    0.0039

    0.0039

    0.0039

    0.0039

    0.0039

  genes: [123.1763 -10.0031 -9.9541 228.4878]
fitness: 0.0039
```

MATLAB Workspace
May 1, 2019

Name	Value
best1	*1x1 struct*
best2	*1x1 struct*
cr	0.5000
i	10
idx_max	1
max	2.5778e-06
maxc	0.0039
mr	0.5000
p	1400
pops_co	*1x700 struct*
pops_mu	*1x700 struct*
population	*1x1400 struct*
save	[0.0012,2.5778e-06,0.0013,0.0039,2.5...
savemax	[0.0012,0.0012,0.0013,0.0039,0.0039,...

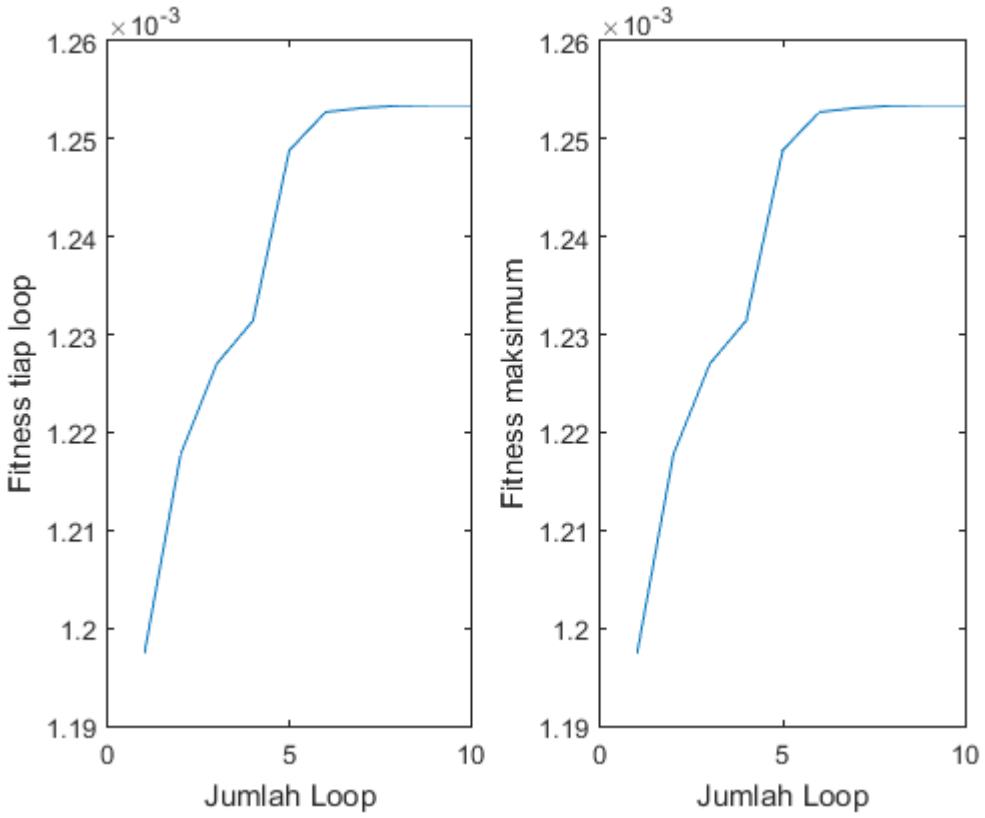


2. Uji Coba *crossover rate* (*cr*) dan *mutation rate* (*mr*)
   Ukuran populasi yang digunakan adalah 600. Kombinasi *crossover rate* (*cr*) dan *mutation rate* (*mr*) yang digunakan adalah antara 0.2 dan 0.8 dengan kelipatan 2. Masing-masing percobaan diulang 10 kali.
   a. *Cr : Mr* = 0.2 : 0.8

```
jumlah populasi (integer) : 600
CrossOver Rate (0-1) : 0.2
mutation rate (0-1) : 0.8
    0.0012

    0.0012

    0.0012

    0.0012

    0.0012

    0.0013

    0.0013

    0.0013

    0.0013

    0.0013

  genes: [12.5614 16.9141 16.7147 -3.4198]
fitness: 0.0013
```

MATLAB Workspace
Apr 30, 2019

Name	Value
best1	*1x1 struct*
best2	*1x1 struct*
cr	0.8000
i	10
idx_max	1
max	0.0013
maxc	0.0013
mr	0.2000
p	600
pops_co	*1x480 struct*
pops_mu	*1x120 struct*
population	*1x600 struct*
save	[0.0012,0.0012,0.0012,0.0012,0.0012,...
savemax	[0.0012,0.0012,0.0012,0.0012,0.0012,...

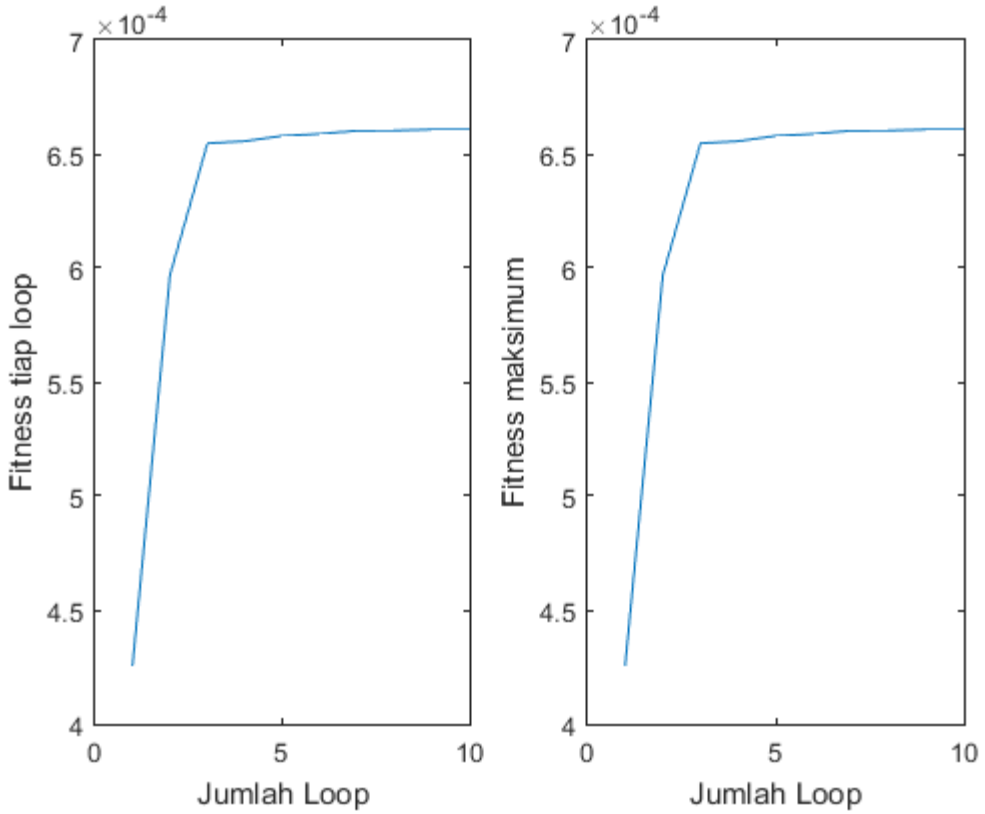


b. *Cr : Mr* = 0.4 : 0.6

```
jumlah populasi (integer) : 600
CrossOver Rate (0-1) : 0.4
mutation rate (0-1) : 0.6
   4.2587e-04

   5.9683e-04

   6.5467e-04

   6.5561e-04

   6.5795e-04

   6.5878e-04

   6.6022e-04

   6.6028e-04

   6.6074e-04

   6.6076e-04

    genes: [-34.1876 55.7402 55.6344 -17.7948]
  fitness: 6.6076e-04
```

MATLAB Workspace
Apr 30, 2019

Name	Value
best1	*1x1 struct*
best2	*1x1 struct*
cr	0.6000
i	10
idx_max	1
max	6.6076e-04
maxc	6.6076e-04
mr	0.4000
p	600
pops_co	*1x360 struct*
pops_mu	*1x240 struct*
population	*1x600 struct*
save	[4.2587e-04,5.9683e-04,6.5467e-04,6...
savemax	[4.2587e-04,5.9683e-04,6.5467e-04,6...

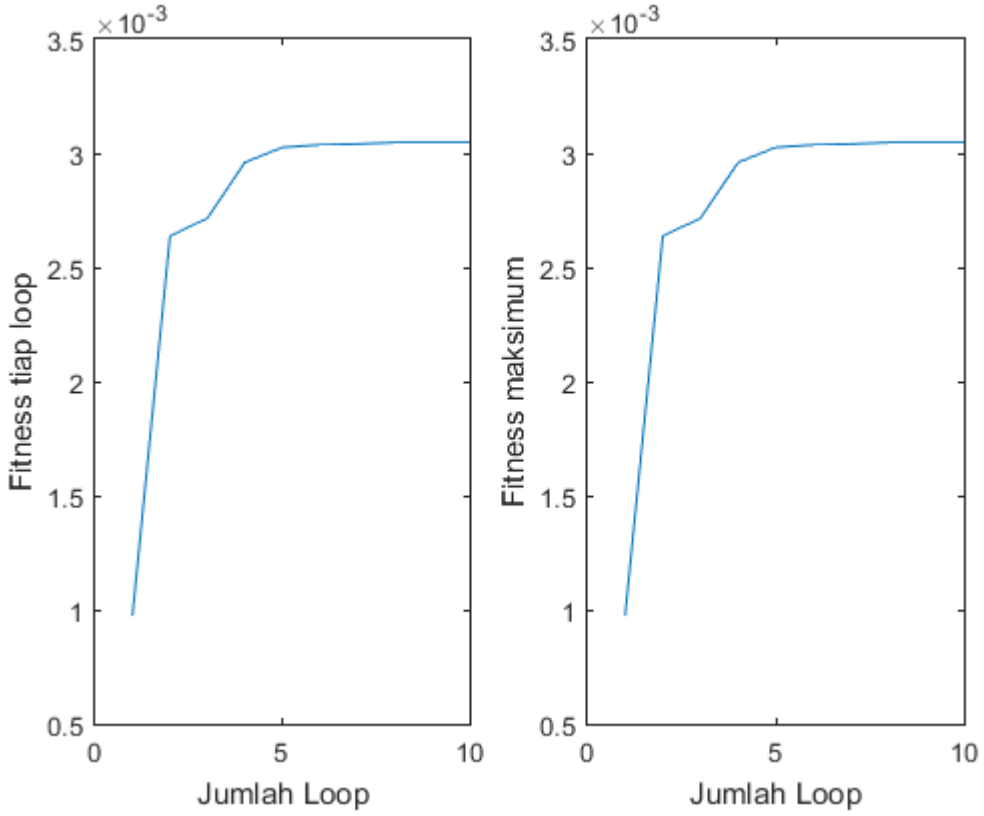


c. *Cr : Mr* = 0.5 : 0.5

```
jumlah populasi (integer) : 600
CrossOver Rate (0-1) : 0.5
mutation rate (0-1) : 0.5
   9.7767e-04

    0.0026

    0.0027

    0.0030

    0.0030

    0.0030

    0.0030

    0.0030

    0.0030

    0.0030

  genes: [-21.2528 -12.3760 -12.3495 -57.8541]
fitness: 0.0030
```

MATLAB Workspace
Apr 30, 2019

Name	Value
best1	*1x1 struct*
best2	*1x1 struct*
cr	0.5000
i	10
idx_max	1
max	0.0030
maxc	0.0030
mr	0.5000
p	600
pops_co	*1x300 struct*
pops_mu	*1x300 struct*
population	*1x600 struct*
save	[9.7767e-04,0.0026,0.0027,0.0030,0.0...
savemax	[9.7767e-04,0.0026,0.0027,0.0030,0.0...

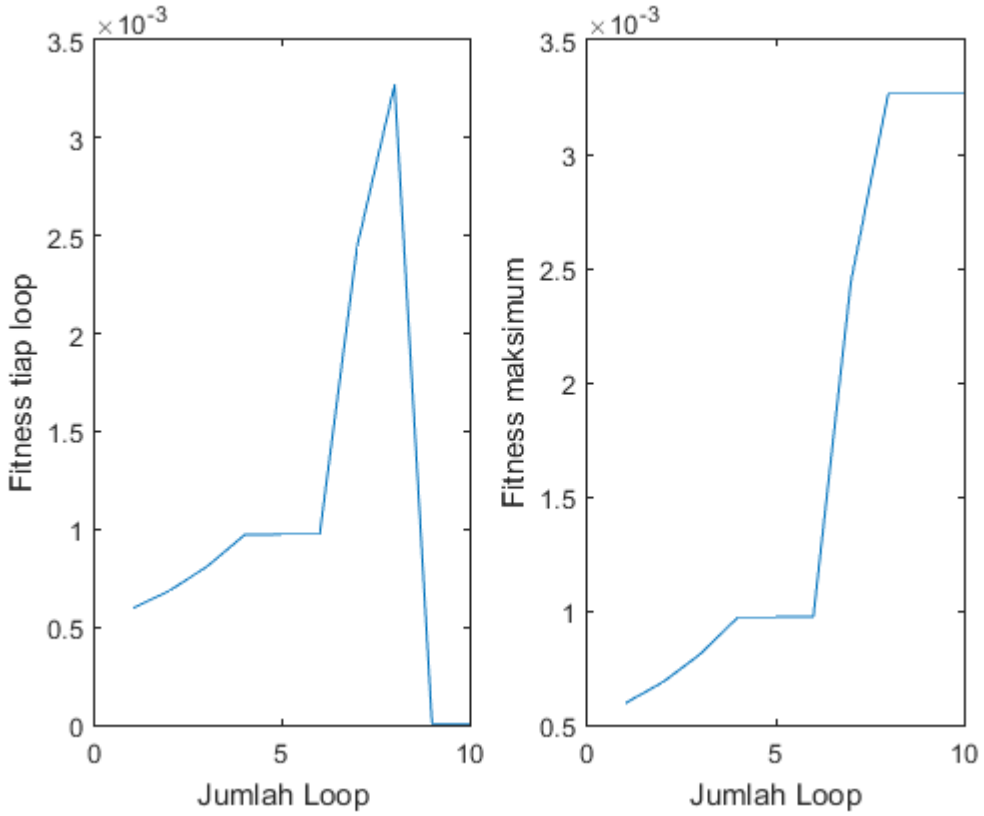


d. *Cr : Mr* = 0.6 : 0.4

```
jumlah populasi (integer) : 600
CrossOver Rate (0-1) : 0.6
mutation rate (0-1) : 0.4
   5.9451e-04

   6.8717e-04

   8.1063e-04

   9.7214e-04

   9.7312e-04

   9.7367e-04

    0.0024

    0.0033

    0.0033

    0.0033

     genes: [14.2076 2.4897 2.5399 150.7066]
   fitness: 0.0033
```

MATLAB Workspace
May 1, 2019

Name	Value
best1	*1x1 struct*
best2	*1x1 struct*
cr	0.4000
i	10
idx_max	1
max	2.6217e-06
maxc	0.0033
mr	0.6000
p	600
pops_co	*1x240 struct*
pops_mu	*1x360 struct*
population	*1x600 struct*
save	[5.9451e-04,6.8717e-04,8.1063e-04,9...
savemax	[5.9451e-04,6.8717e-04,8.1063e-04,9...



e. *Cr : Mr* = 0.8 : 0.2

```
jumlah populasi (integer) : 600
CrossOver Rate (0-1) : 0.8
mutation rate (0-1) : 0.2
   5.3644e-04

   5.4606e-04

   5.4606e-04

   6.1946e-04

   6.1946e-04

   6.1946e-04

   6.1946e-04

   6.1946e-04

   6.1946e-04

   6.3734e-04

    genes: [181.7039 -29.2792 -28.8713 165.9578]
  fitness: 6.3734e-04
```

MATLAB Workspace
May 1, 2019

Name	Value
best1	*1x1 struct*
best2	*1x1 struct*
cr	0.2000
i	10
idx_max	1
max	6.3734e-04
maxc	6.3734e-04
mr	0.8000
p	600
pops_co	*1x120 struct*
pops_mu	*1x480 struct*
population	*1x600 struct*
save	[5.3644e-04,5.4606e-04,8.8275e-06,6...
savemax	[5.3644e-04,5.4606e-04,5.4606e-04,6...