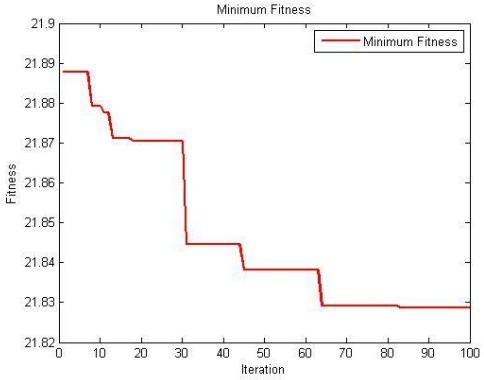
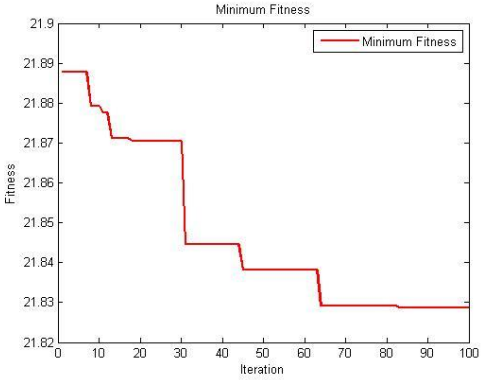
LAMPIRAN

Lampiran 1 Grafik Konvergensi Simulasi

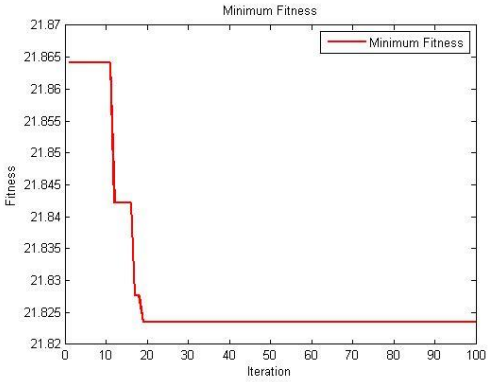
a. Simulasi Penambahan Kapasitor Bank Pada Sistem Standar IEEE 30 Bus



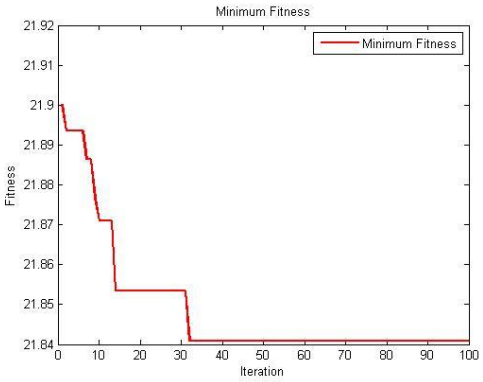
(1)



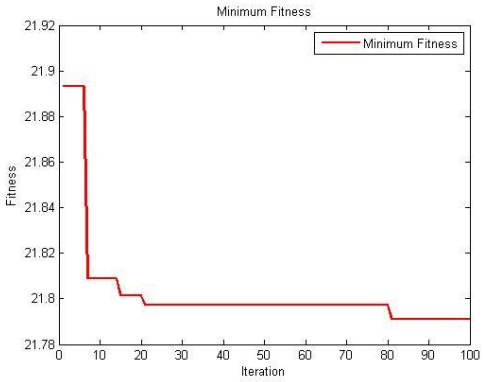
(2)



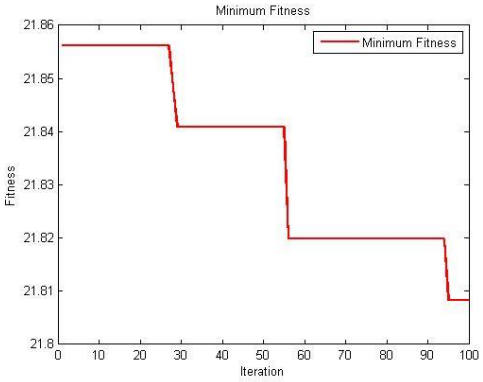
(3)



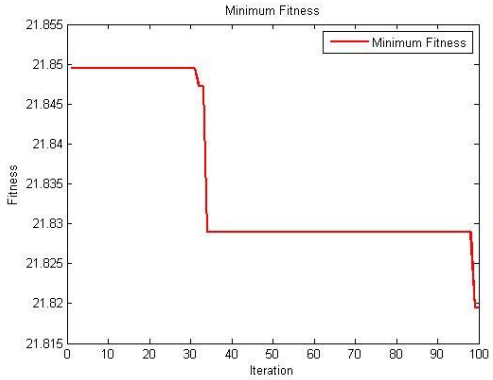
(4)



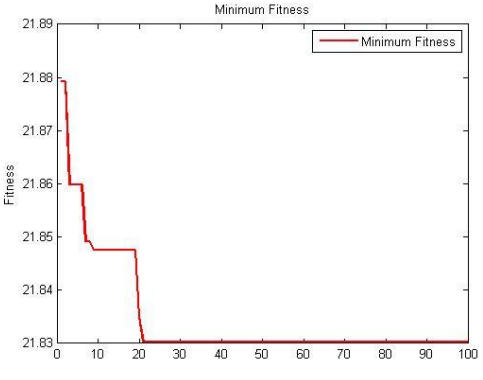
(5)



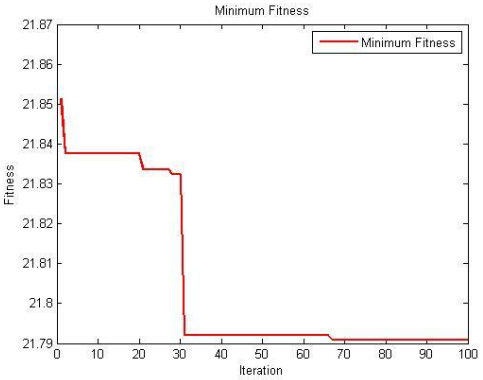
(6)



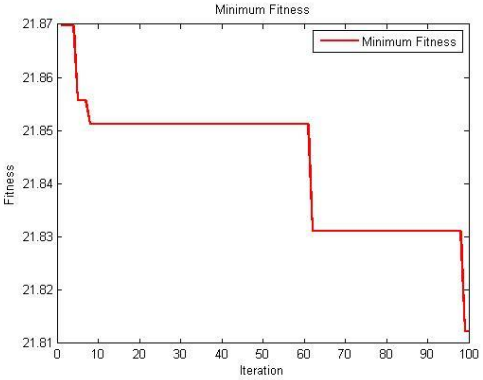
(7)



(8)

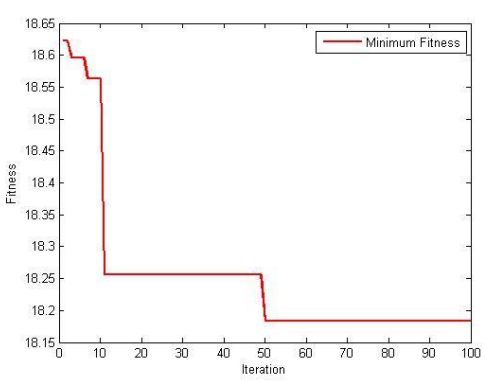


(9)

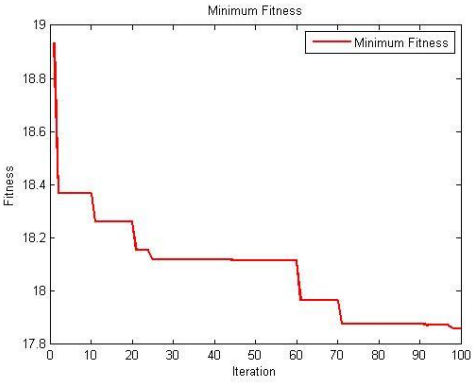


(10)

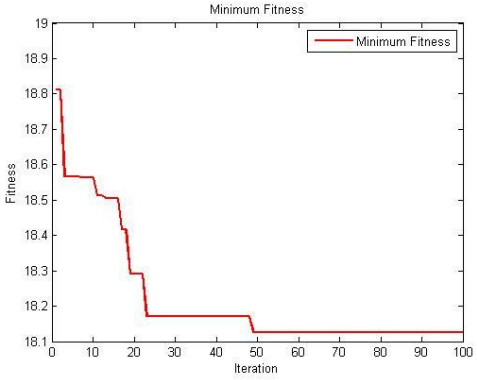
b. Simulasi Penambahan UPFC Pada Sistem Standar IEEE 30 Bus



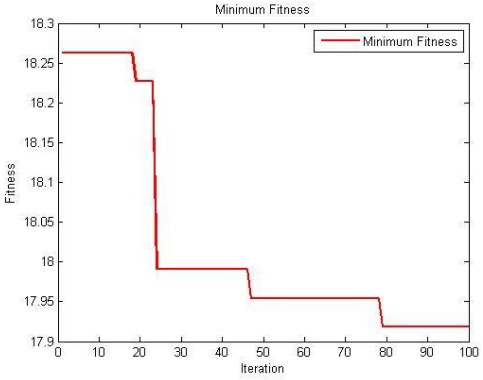
(1)



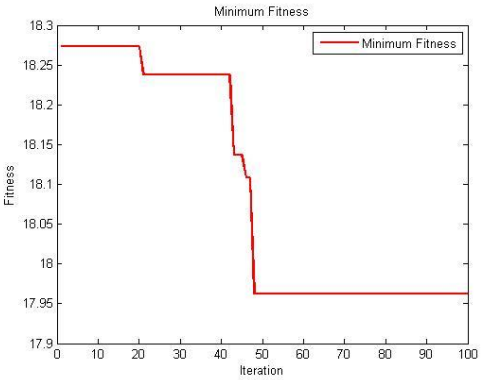
(2)


Minimum Fitness
Minimum Fitness
Fitness
Iteration

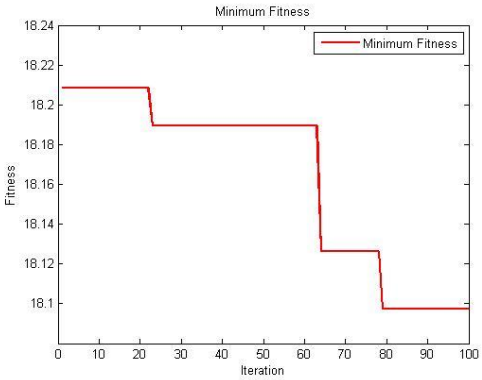
(3)


Minimum Fitness
Minimum Fitness
Fitness
Iteration

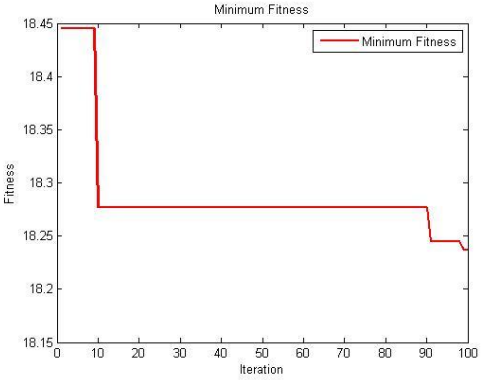
(4)


Minimum Fitness
Minimum Fitness
Fitness
Iteration

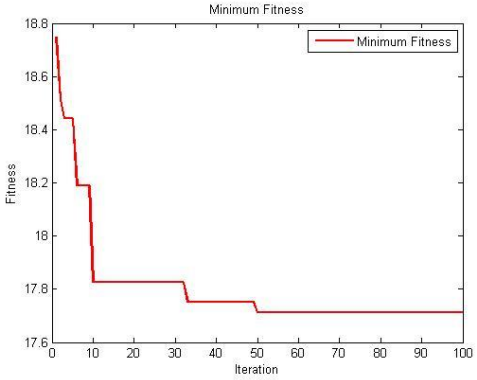
(5)


Minimum Fitness
Minimum Fitness
Fitness
Iteration

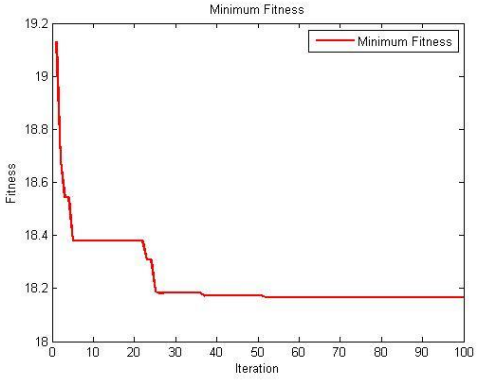
(6)


Minimum Fitness
Minimum Fitness
Fitness
Iteration

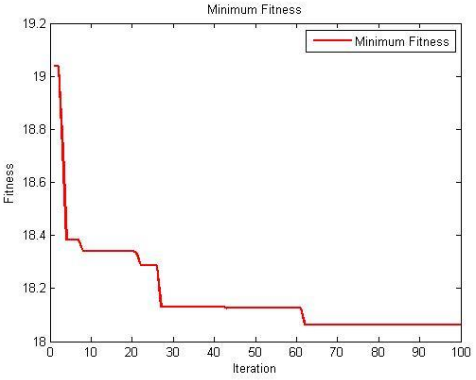
(7)


Minimum Fitness
Minimum Fitness
Fitness
Iteration

(8)

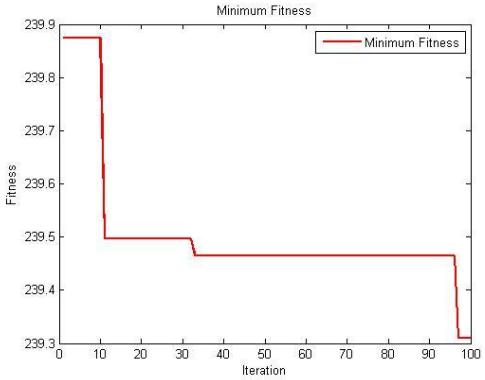


(9)

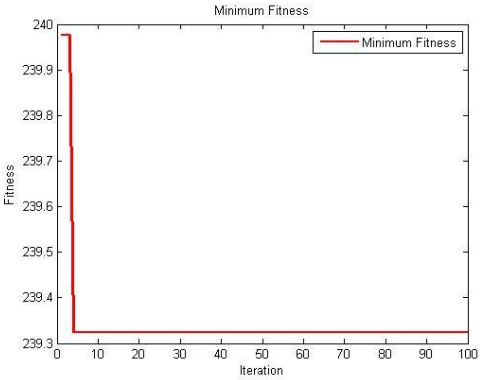


(10)

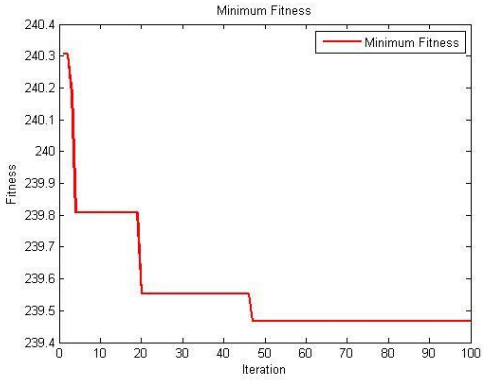
c. Simulasi Penambahan Kapasitor Bank Pada Sistem Tenaga Listrik 500 kV



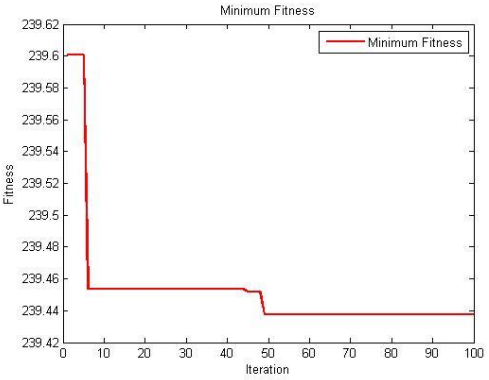
(1)



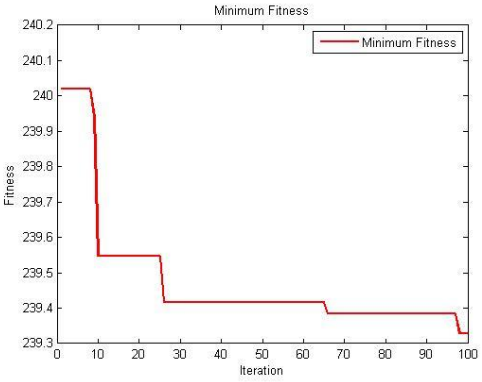
(2)



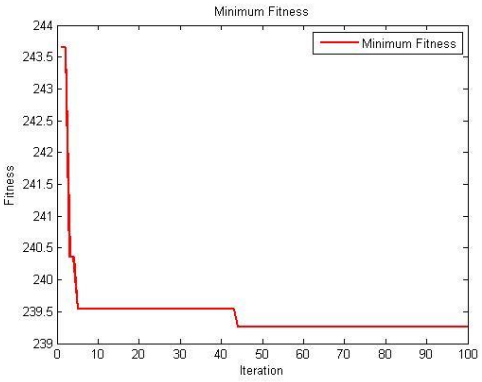
(3)



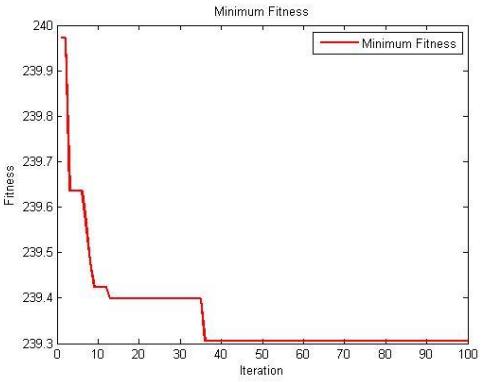
(4)


Minimum Fitness
Minimum Fitness
Fitness
Iteration

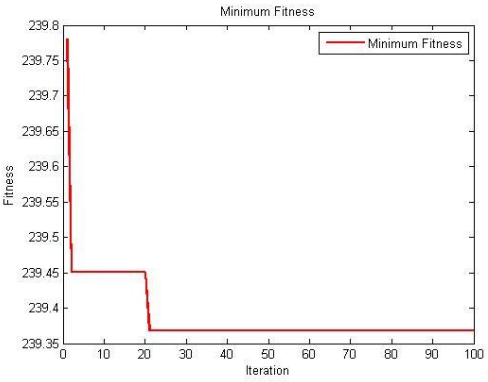
(5)


Minimum Fitness
Minimum Fitness
Fitness
Iteration

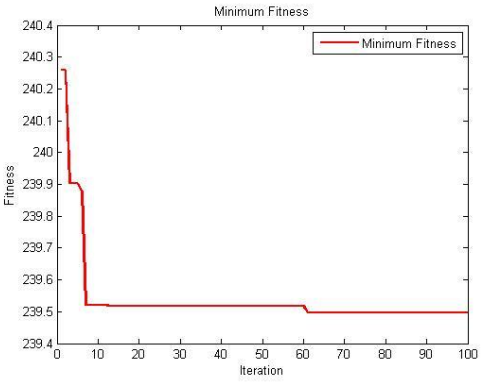
(6)


Minimum Fitness
Minimum Fitness
Fitness
Iteration

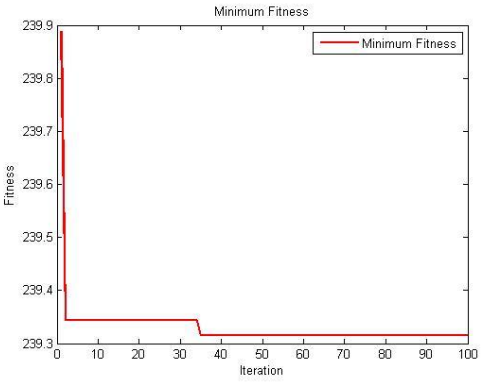
(7)


Minimum Fitness
Minimum Fitness
Fitness
Iteration

(8)

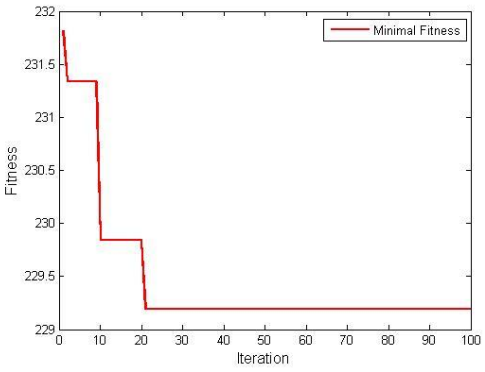

Minimum Fitness
Minimum Fitness
Fitness
Iteration

(9)

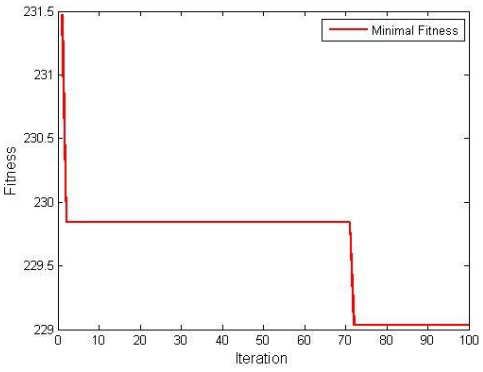

Minimum Fitness
Minimum Fitness
Fitness
Iteration

(10)

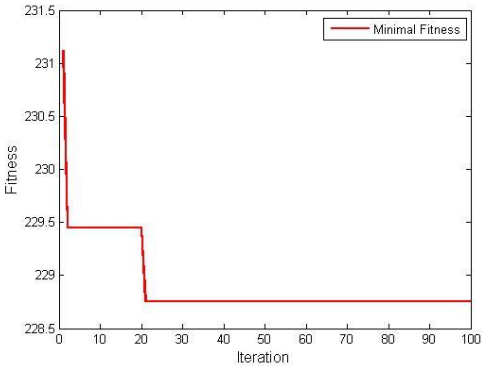
d. Simulasi Penambahan UPFC Pada Sistem Tenaga Listrik 500 kV



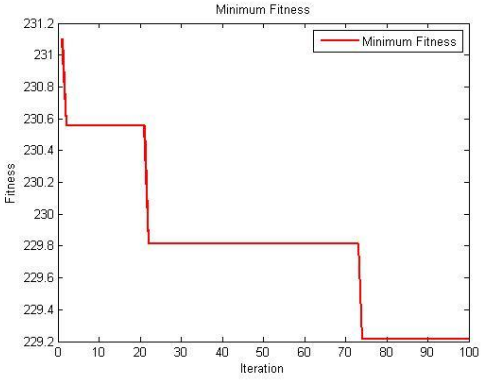
(1)



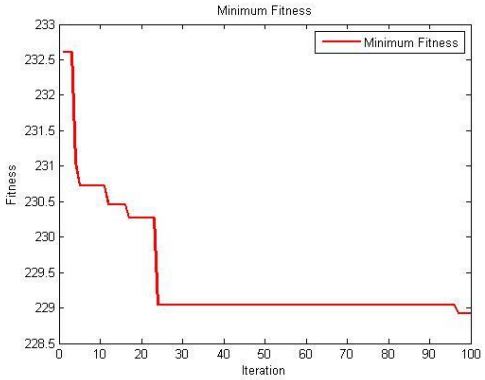
(2)



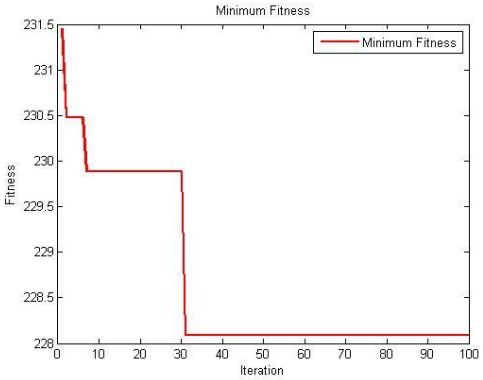
(3)



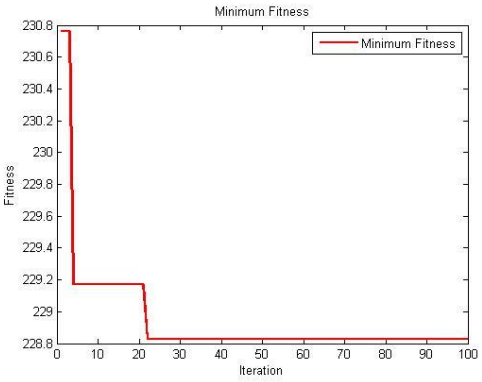
(4)



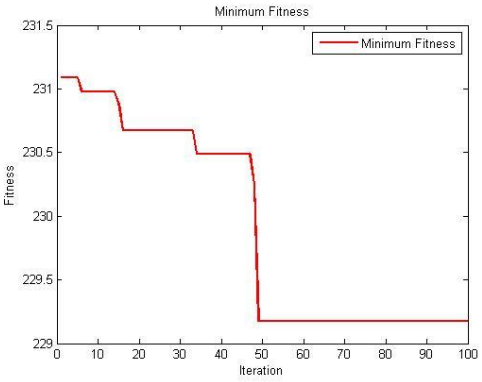
(5)



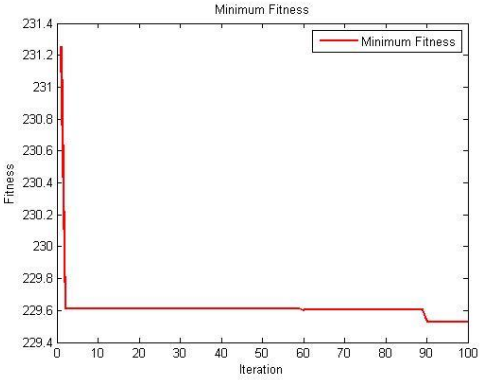
(6)



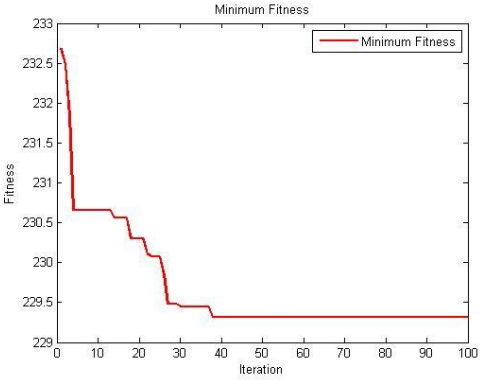
(7)



(8)



(9)



(10)

Lampiran 2 Listing Program

a. Optimasi Kapasitor Bank Menggunakan Metode ICA

```
%% Problem Statement
ProblemParams.nKB = 3;
ProblemParams.nCandidate = max(busdata(:,1));
ProblemParams.QMin = 0;
ProblemParams.QMax = 75;
ProblemParams.SearchSpaceSize = max(busdata(:,1));

%% Algorithmic Parameter Setting
AlgorithmParams.NumOfCountries = 200;
AlgorithmParams.NumOfInitialImperialists = 15;
AlgorithmParams.NumOfAllColonies = AlgorithmParams.NumOfCountries -
AlgorithmParams.NumOfInitialImperialists;
AlgorithmParams.NumOfDecades = 100;
AlgorithmParams.RevolutionRate = 0.6;
AlgorithmParams.AssimilationCoefficient = .5;
AlgorithmParams.AssimilationAngleCoefficient = .5;
AlgorithmParams.Zeta = 0.02;
AlgorithmParams.DampRatio = 0.99;
AlgorithmParams.StopIfJustOneEmpire = false;
AlgorithmParams.UnitingThreshold = 0.02;
```

```
zarib = 1.05;
alpha = 0.1;

%% Display Setting
DisplayParams.PlotCost = true;     % "true" to plot. "false"
if DisplayParams.PlotCost
    DisplayParams.CostFigureHandle = figure('Name','Plot of
Fitness','NumberTitle','off');
    DisplayParams.CostAxisHandle = axes;
end

ColorMatrix = [1 0 0; 0 1 0; 0 0 1; 1 1 0; 1 0 1; 0 1 1; 1 1 1;
               0.5 0.5 0.5; 0 0.5 0.5; 0.5 0 0.5; 0.5 0.5 0; 0.5 0 0;
               0 0.5 0; 0 0 0.5; 1 0.5 1; 0.1*[1 1 1]; 0.2*[1 1 1];
               0.3*[1 1 1]; 0.4*[1 1 1]; 0.5*[1 1 1]; 0.6*[1 1 1]];

DisplayParams.ColorMatrix = [ColorMatrix ; sqrt(ColorMatrix)];

%% Creation of Initial Empires
InitialCountries =
GenerateNewCountry(AlgorithmParams.NumOfCountries,ProblemParams);

% Calculates the cost of each country. The less the cost is, the more is
the power.
for iii = 1:AlgorithmParams.NumOfCountries
    InitialCost(iii,1) =
CostFunction(busdata,linedata,InitialCountries.Candidate(iii,:),InitialCo
untries.Q(iii,:),ProblemParams);
end

[InitialCost,SortInd] = sort(InitialCost);
InitialCountries.Candidate = InitialCountries.Candidate(SortInd,:);
InitialCountries.Q = InitialCountries.Q(SortInd,:);

Empires =
CreateInitialEmpires(InitialCountries,InitialCost,AlgorithmParams,
ProblemParams);

%% Main Loop
MinFitness = repmat(nan,AlgorithmParams.NumOfDecades,1);
% MeanFitness = repmat(nan,AlgorithmParams.NumOfDecades,1);

if DisplayParams.PlotCost
    axes(DisplayParams.CostAxisHandle);
    if any(findall(0)==DisplayParams.CostFigureHandle)

h_MinLosPlot =
plot(MinFitness,'r','LineWidth',1.5,'YDataSource','MinFitness');
% hold on;
%h_MeanLosPlot =
plot(MeanFitness,'k:','LineWidth',1.5,'YDataSource','MeanFitness');
title ('Minimum Fitness')
xlabel('Decade');
ylabel ('Fitness');
legend('Minimum Fitness');
% hold off;
pause(0.05);
     end
end
```

```
counter=1;
for Decade = 1:AlgorithmParams.NumOfDecades
    AlgorithmParams.RevolutionRate = AlgorithmParams.DampRatio *
AlgorithmParams.RevolutionRate;

    Remained = AlgorithmParams.NumOfDecades - Decade;
    for ii = 1:numel(Empires)
%% Assimilation;  Movement of Colonies Toward Imperialists (Assimilation
Policy)
Empires(ii) =
AssimilateColonies(Empires(ii),AlgorithmParams,ProblemParams);

%% Revolution;  A Sudden Change in the Socio-Political Characteristics
Empires(ii) = RevolveColonies(Empires(ii),AlgorithmParams,ProblemParams);

%% New Cost Evaluation
   for iii = 1:size(Empires(ii).ColoniesCandidate,1)
Empires(ii).ColoniesCost(iii,1) =
CostFunction(busdata,linedata,Empires(ii).ColoniesCandidate(iii,:),Empire
s(ii).ColoniesQ(iii,:),ProblemParams);
    end

%% Empire Possession  (****** Power Possession, Empire Possession)
Empires(ii) = PossesEmpire(Empires(ii));

%% Computation of Total Cost for Empires
Empires(ii).TotalCost = Empires(ii).ImperialistCost +
AlgorithmParams.Zeta * mean(Empires(ii).ColoniesCost);

    end

%% Uniting Similiar Empires
Empires = UniteSimilarEmpires(Empires,AlgorithmParams,ProblemParams);

%% Imperialistic Competition
Empires = ImperialisticCompetition(Empires);

    if numel(Empires) == 1 && AlgorithmParams.StopIfJustOneEmpire
        break
    end

%% Displaying the Results

    ImperialistCosts = [Empires.ImperialistCost];
    MinFitness(Decade) = min(ImperialistCosts);
%    MeanFitness(Decade) = mean(ImperialistCosts);

    if DisplayParams.PlotCost
        refreshdata(h_MinLosPlot);
%         refreshdata(h_MeanLosPlot);
        drawnow;
        pause(0.01);
    end
    fprintf('Dekade ke %d selesai \n',counter);
    counter = counter + 1;
end
```

b. Optimasi UPFC Menggunakan Metode ICA

```
%% Problem Statement
ProblemParams.nUPFC = 3;
ProblemParams.nCandidate = length(linedata(:,1));
ProblemParams.PMin = -250;
ProblemParams.PMax = 0;
ProblemParams.QMin = -150;
ProblemParams.QMax = 100;
ProblemParams.SearchSpaceSize = length(linedata(:,1));

%% Algorithmic Parameter Setting
AlgorithmParams.NumOfCountries = 200;
AlgorithmParams.NumOfInitialImperialists = 15;
AlgorithmParams.NumOfAllColonies = AlgorithmParams.NumOfCountries -
AlgorithmParams.NumOfInitialImperialists;
AlgorithmParams.NumOfDecades = 3;
AlgorithmParams.RevolutionRate = 0.6;
AlgorithmParams.AssimilationCoefficient = .5;
AlgorithmParams.AssimilationAngleCoefficient = .5;
AlgorithmParams.Zeta = 0.02;
AlgorithmParams.DampRatio = 0.99;
AlgorithmParams.StopIfJustOneEmpire = false;
AlgorithmParams.UnitingThreshold = 0.02;
zarib = 1.05;
alpha = 0.1;

%% Display Setting
DisplayParams.PlotCost = true;    % "true" to plot. "false"

if DisplayParams.PlotCost
   DisplayParams.CostFigureHandle = figure('Name','Plot of
Fitness','NumberTitle','off');
    DisplayParams.CostAxisHandle = axes;
end

ColorMatrix = [1 0 0; 0 1 0; 0 0 1; 1 1 0; 1 0 1; 0 1 1; 1 1 1;
               0.5 0.5 0.5; 0 0.5 0.5; 0.5 0 0.5; 0.5 0.5 0; 0.5 0 0;
               0 0.5 0; 0 0 0.5; 1 0.5 1; 0.1*[1 1 1]; 0.2*[1 1 1];
               0.3*[ 1 1  1]; 0.4*[1 1 1]; 0.5*[1 1 1]; 0.6*[1 1 1]];
DisplayParams.ColorMatrix = [ColorMatrix ; sqrt(ColorMatrix)];

%% Creation of Initial Empires
InitialCountries =
GenerateNewCountry(AlgorithmParams.NumOfCountries,ProblemParams);

% Calculates the cost of each country. The less the cost is, the more is
the power.
for iii = 1:AlgorithmParams.NumOfCountries
    InitialCost(iii,1) =
CostFunction(busdata,linedata,InitialCountries.Candidate(iii,:),InitialCo
untries.PQ(iii,:),ProblemParams);
end

[InitialCost,SortInd] = sort(InitialCost);
InitialCountries.Candidate = InitialCountries.Candidate(SortInd,:);
InitialCountries.PQ = InitialCountries.PQ(SortInd,:);
```

```
Empires =
CreateInitialEmpires(InitialCountries,InitialCost,AlgorithmParams,
ProblemParams);

%% Main Loop
Fitness = repmat(nan,AlgorithmParams.NumOfDecades,1);
MeanLosses = repmat(nan,AlgorithmParams.NumOfDecades,1);

if DisplayParams.PlotCost
    axes(DisplayParams.CostAxisHandle);
    if any(findall(0)==DisplayParams.CostFigureHandle)

h_MinLosPlot=plot(Fitness,'r','LineWidth',1.5,'YDataSource','Fitness');
        title ('Minimum Fitness')
        xlabel('Decade');
        ylabel ('Fitness');
        legend('Minimum Fitness');

%       hold on;
%
h_MeanLosPlot=plot(MeanLosses,'k:','LineWidth',1.5,'YDataSource','MeanLos
ses');
%       hold off;
        pause(0.05);
    end
end
counter = 1;
for Decade = 1:AlgorithmParams.NumOfDecades
    AlgorithmParams.RevolutionRate = AlgorithmParams.DampRatio *
AlgorithmParams.RevolutionRate;

    Remained = AlgorithmParams.NumOfDecades - Decade;
    for ii = 1:numel(Empires)
        %% Assimilation;  Movement of Colonies Toward Imperialists
(Assimilation Policy)
        Empires(ii) =
AssimilateColonies(Empires(ii),AlgorithmParams,ProblemParams);

        %% Revolution;  A Sudden Change in the Socio-Political
Characteristics
        Empires(ii) =
RevolveColonies(Empires(ii),AlgorithmParams,ProblemParams);

        %% New Cost Evaluation
        for iii = 1:size(Empires(ii).ColoniesCandidate,1)
            Empires(ii).ColoniesCost(iii,1) =
CostFunction(busdata,linedata,Empires(ii).ColoniesCandidate(iii,:),Empire
s(ii).ColoniesPQ(iii,:),ProblemParams);
        end

        %% Empire Possession  (****** Power Possession, Empire
Possession)
        Empires(ii) = PossesEmpire(Empires(ii));

        %% Computation of Total Cost for Empires
        Empires(ii).TotalCost = Empires(ii).ImperialistCost +
AlgorithmParams.Zeta * mean(Empires(ii).ColoniesCost);

    end
```

```
    %% Uniting Similiar Empires
    Empires = UniteSimilarEmpires(Empires,AlgorithmParams,ProblemParams);

    %% Imperialistic Competition
    Empires = ImperialisticCompetition(Empires);

    if numel(Empires) == 1 && AlgorithmParams.StopIfJustOneEmpire
        break
    end

   %% Displaying the Results

    ImperialistCosts = [Empires.ImperialistCost];
    Fitness(Decade) = min(ImperialistCosts);
%    MeanLosses(Decade) = mean(ImperialistCosts);

    if DisplayParams.PlotCost
        refreshdata(h_MinLosPlot);
%         refreshdata(h_MeanLosPlot);
        drawnow;
        pause(0.01);
    end

    fprintf('Dekade ke %d selesai \n',counter);
    counter = counter + 1;
end
```