```
clc;
clear all;
%==========================================================================
%rand('seed',0)
%==========================================================================
%                 Archimedes optimization algorithm (AOA)
%==========================================================================
tic
%==========================================================================
Nb=34;                                                        % No. of buses
%==========================================================================
Materials_no=34;                                     % Number of
Particles
dim=3;                                        % Number of control varaibles
Max_iter=100;                                % Maximum number of iterations
%==========================================================================
% C3=2;C4=.5;                                 %cec and engineering problems
C3=1;C4=2;                                %standard Optimization functions
%==========================================================================
Qc_min=500;                                  % Minimum limit of capacitor
Qc_max=1200;                                 % Maximum limit of capacitor
%==========================================================================
bounds=[Qc_min*ones(Nb,1)  Qc_max*ones(Nb,1)];
%==========================================================================
nv=size(bounds,1);                                      % No. of variables
L=bounds(:,1)';                                       % Lower boundaries
Lmax=L([ones(Materials_no,1)],1:nv);
H=bounds(:,2)';                                       % Upper boundaries
Hmax=H([ones(Materials_no,1)],1:nv);
rang=H-L;
%==========================================================================
% Initial population
%==========================================================================
for k=1:Materials_no;
    for j=1:dim;
        x1(k,j)=L(j)+rand*rang(j);
    end
end
x1;                                                 % Sizing of Capacitors
%==========================================================================
bounds1=[2*ones(Nb,1)  Nb*ones(Nb,1)];
%==========================================================================
nv1=size(bounds1,1);                                      % No. of variables
L1=bounds1(:,1)';                                       % Lower boundaries
Lmax1=L1([ones(Materials_no,1)],1:nv);
```

```
H1=bounds1(:,2)';                                           % Upper boundaries
Hmax1=H1([ones(Materials_no,1)],1:nv);
rang1=H1-L1;
%==========================================================================
% Initial population
%==========================================================================
for k=1:Materials_no;
    for j=1:dim;
        x2(k,j)=L1(j)+rand*rang1(j);
    end
end
x2;                                                    % Sitting of Capacitors
%==========================================================================
%                              Initialization
%==========================================================================
C1=0.5;
C2=0.5;
%==========================================================================
u=.9;                                                   %paramter in Eq. (12)
l=.1;                                                   %paramter in Eq. (12)
X=[x2 x1];                                          %initial positions Eq. (4)
den=rand(Materials_no,2*dim);                                         % Eq. (5)
vol=rand(Materials_no,2*dim);
ss1=2*ones(Materials_no,dim)+rand(Materials_no,dim)*(Nb-2);
ss2=Qc_min*ones(Materials_no,dim)+rand(Materials_no,dim)*(Qc_max-Qc_min);
acc=[ss1 ss2];                                                      % Eq. (6)
%==========================================================================
%                         Objective Function
%==========================================================================
bvar1=round(X(:,1:dim));                % Capacitor Locations (0 off, 1 on)
bvar2=X(:,dim+1:2*dim);                           % Capacitor Sizing (KVAR)
bvar3=[bvar1 X(:,dim+1:2*dim)];
%==========================================================================
%                        DISTRIBUTION LOAD FLOW
%==========================================================================
PL=zeros(Materials_no,1);
Vb=zeros(Materials_no,Nb);
for ii=1:Materials_no
    %======================================================================
    bvar4=zeros(Nb,1);
    %======================================================================
    ss3=bvar3(ii,1:dim);
    ss4=bvar3(ii,dim+1:2*dim);
    bvar4(ss3,1)=ss4;
    Qcc=bvar4;
```

```
    %======================================================================
    backward_forward_sweep               % Backward Forward Load Flow
    %======================================================================
    PL(ii,1)=PLoss;                % Power Losses for each solution (P.u.)
    Vb(ii,:)=V_bus;                % Voltage magnitude at each bus (P.u.)
    %======================================================================
end
PL;
%====================================================================
%                      FIRING CONSTRAINTS
%====================================================================
%constraints
%====================================================================
%                     Vmin = <  Vi  < =  Vmax
%====================================================================
const1=Vb;
for i=1:Materials_no;
    ppp1=find(const1(i,:)<0.85);%find the index of invisible solution
    ppp2=sum(ppp1);
    if ppp2>0
       PL(i,:)=inf;
    end
end
PL;
%====================================================================
%                         Nc < = Nc_max
%====================================================================
% const2=bvar2;
% for i=1:npop;
%     ppp3=find(const2(i,:)>1);  % find the index of invisible solution
%     ppp4=length(ppp3);
%     if ppp4>3
%        PL(i,:)=inf;
%     end
% end
% PL;
%====================================================================
%                      Qc_total < = 3007
%====================================================================
const3=bvar2;
for i=1:Materials_no;
    ppp5=find(const3(i,:)>1);  % find the index of invisible solution
    ppp6=const3(i,ppp5);
    ppp7=sum(ppp6);
    if ppp7>3007
```

```
       PL(i,:)=inf;
    end
end
PL;
%==========================================================================
%                      Initialize the population/solutions
%==========================================================================
Fitness=PL;
%==========================================================================
% Find the current best
sol1=[bvar1 bvar2 Fitness];
sol2=sortrows(sol1,2*dim+1);
bbst=[sol2(1,:)];
gbst=bbst(:,2*dim+1);
%==========================================================================
[Scorebest, Score_index] = min(Fitness);
Xbest=X(Score_index,:);
den_best=den(Score_index,:);
vol_best=vol(Score_index,:);
acc_best=acc(Score_index,:);
acc_norm=acc;
%==========================================================================
%               Start the iterations -- AOA
%==========================================================================
for t=1:Max_iter
    %======================================================================
    TF=exp(((t-Max_iter)/(Max_iter)));                           % Eq. (8)
    if TF>1
        TF=1;
    end
    d=exp((Max_iter-t)/Max_iter)-(t/Max_iter);                   % Eq. (9)
    acc=acc_norm;
    r=rand();
    %======================================================================
    for i=1:Materials_no
        den(i,:)=den(i,:)+r*(den_best-den(i,:));                 % Eq. (7)
        vol(i,:)=vol(i,:)+r*(vol_best-vol(i,:));
        if TF<.45                                               %collision
            mr=randi(Materials_no);
            acc_temp(i,:)=((den(mr,:).*vol(mr,:).*acc(mr,:)))./(den(i,:).*vol(i,:
))*rand;   % Eq. (10)
        else
            acc_temp(i,:)=(den_best.*vol_best.*acc_best)./(den(i,:).*vol(i,:))*ra
nd;        % Eq. (11)
        end
```

```
    end
    acc_norm=((u*(acc_temp-min(acc_temp(:)))).(max(acc_temp(:))-
min(acc_temp(:))))+l;      % Eq. (12)
    %======================================================================
     for i=1:Materials_no
            if TF<.4
                for j=1:size(X,2)
                    mrand=randi(Materials_no);
                    Xnew(i,j)=X(i,j)+C1*rand*acc_norm(i,j).*(X(mrand,j)-
X(i,j))*d;          % Eq. (13)
                end
            else
                for j=1:size(X,2)
                    p=2*rand-
C4;                                                               % Eq. (15)
                    T=C3*TF;
                    if T>1
                        T=1;
                    end
                    if p<.5
                        Xnew(i,j)=Xbest(j)+C2*rand*acc_norm(i,j).*(T*Xbest(j)-
X(i,j))*d;     % Eq. (14)
                    else
                        Xnew(i,j)=Xbest(j)-C2*rand*acc_norm(i,j).*(T*Xbest(j)-
X(i,j))*d;
                    end
                end
            end
     end
    %======================================================================
    %                          Objective Function
    %======================================================================
    bvar1=round(Xnew(:,1:dim));         % Capacitor Locations (0 off, 1 on)
    bvar2=Xnew(:,dim+1:2*dim);                       % Capacitor Sizing (KVAR)
    bvar3=[bvar1 X(:,dim+1:2*dim)];
    %======================================================================
    %                         DISTRIBUTION LOAD FLOW
    %======================================================================
    PL=zeros(Materials_no,1);
    Vb=zeros(Materials_no,Nb);
    for ii=1:Materials_no
        %==================================================================
        bvar4=zeros(Nb,1);
        %==================================================================
        ss3=bvar3(ii,1:dim);
```

```
    ss4=find(ss3(1,:)>Nb);
    ss3(1,ss4)=Nb;
    ss5=find(ss3(1,:)<2);
    ss3(1,ss5)=2;
    ss5=bvar3(ii,dim+1:2*dim);
    bvar4(ss3,1)=ss5;
    Qcc=bvar4;
    %===================================================================
    backward_forward_sweep               % Backward Forward Load Flow
    %===================================================================
    PL(ii,1)=PLoss;              % Power Losses for each solution (P.u.)
    Vb(ii,:)=V_bus;               % Voltage magnitude at each bus (P.u.)
    %===================================================================
end
PL;
%=====================================================================
%                      FIRING CONSTRAINTS
%=====================================================================
%constraints
%=====================================================================
%                     Vmin = <  Vi  < =  Vmax
%=====================================================================
const1=Vb;
for i=1:Materials_no;
    ppp1=find(const1(i,:)<0.85);%find the index of invisible solution
    ppp2=sum(ppp1);
    if ppp2>0
       PL(i,:)=inf;
    end
end
PL;
%=====================================================================
%                         Nc < = Nc_max
%=====================================================================
% const2=bvar2;
% for i=1:npop;
%     ppp3=find(const2(i,:)>1);  % find the index of invisible solution
%     ppp4=length(ppp3);
%     if ppp4>3
%        PL(i,:)=inf;
%     end
% end
% PL;
%=====================================================================
%                      Qc_total < = 3007
```

```
    %====================================================================
    const3=bvar2;
    for i=1:Materials_no;
        ppp5=find(const3(i,:)>1);  % find the index of invisible solution
        ppp6=const3(i,ppp5);
        ppp7=sum(ppp6);
        if ppp7>3007
           PL(i,:)=inf;
        end
    end
    PL;
    %======================================================================
    %                     Initialize the population/solutions
    %======================================================================
    Fitness=PL;
    %======================================================================
    % Find the current best
    sol1=[bvar1 bvar2 Fitness];
    sol2=sortrows(sol1,2*dim+1);
    bbst=[sol2(1,:)];
    gbst=bbst(:,2*dim+1);
    %======================================================================
    [gmin,ind]=min(bbst(:,2*dim+1));
    if gmin<gbst
       gbst=gmin;
    end
    %======================================================================
    ansr(t,:)=[bbst gbst];
    outt(t,:)=[t gmin gbst];
    eval(t)=gbst;
    %======================================================================
end
ansr;
%==========================================================================
fprintf('=================================================================')
sss1=sort(outt(:,3));
sss2=zeros(Max_iter,1);
sss3=[Max_iter:-1:1];
sss2(sss3,1)=sss1;
Fitness=sss2;
%==========================================================================
pp1=ansr(Max_iter,2*dim+2);
pp2=find(ansr(:,2*dim+2)==pp1);
pp3=ansr(pp2(1,1),1:2*dim)';
%==========================================================================
```

```
fprintf('=================================================================')
Qcc_Locations=pp3(1:dim,1);
Qcc_Size=pp3(dim+1:2*dim,1);
%==========================================================================
Qcc_Locations_Size_Kvar=[Qcc_Locations Qcc_Size]
% BF_C(Qcc_Locations,Qcc_Size)

Qcc_Total_Kvar=sum(Qcc_Locations_Size_Kvar(:,2))
%==========================================================================
bvar1=zeros(Nb,1);
%==========================================================================
ss1=Qcc_Locations;
ss2=Qcc_Size;
bvar1(ss1,1)=ss2;
Qcc1_Size_Kvar=bvar1;
%==========================================================================
backward_forward_sweep_final                   % Backward Forward Load Flow
%==========================================================================
fprintf('=================================================================')
V_bus_with_C=V_bus;                    %Voltage magnitude at each bus (P.u.)
fprintf('=================================================================')
V_bus_tot_with_C=sum(V_bus_with_C)
%==========================================================================
%                        Voltage magnitude without Capacitors
%==========================================================================
fprintf('=================================================================')
V_bus_without_C=[ 1
      0.99414
      0.98902
      0.98205
      0.97606
      0.97041
      0.96659
      0.96448
      0.96201
      0.96083
      0.96037
      0.96023
      0.98869
      0.98838
       0.9883
      0.98829
      0.96595
      0.96224
      0.95815
```

```
      0.95485
      0.95199
      0.94872
      0.94603
      0.94351
       0.9423
      0.94183
      0.94169
      0.96625
      0.96603
      0.96591
      0.96049
      0.96015
      0.95998
      0.95992];
fprintf('=================================================================')
V_bus_tot_without_C=sum(V_bus_without_C)
%==========================================================================
figure(1)
plot(outt(:,1),Fitness);
xlabel('Iteration');
ylabel('Objective Function');
%==========================================================================
figure(2)
plot(1:Nb,V_bus_without_C,'r',1:Nb,V_bus_with_C,'b')
xlabel('Buses');
ylabel('Bus Voltages (p.u.)');
%==========================================================================
%                           Power Loss in each branch
%==========================================================================
fprintf('===============Active power loss in each branch (kW)=============')
Plosskw = (Pl) * 100000;
fprintf('=============Reactive power loss in each branch (kVAR)===========')
Qlosskw = (Ql) * 100000;
%==========================================================================
%                              Total Power Loss
%==========================================================================
fprintf('=================Total active power loss (kW)===================')
Total_PLoss = (PLL) * 100000
fprintf('===============Total reactive power loss (kVAR)=================')
Total_QLoss = (QLL) * 100000
%==========================================================================
fprintf('=================minimum voltage===================')
minimum_voltage = min(V_bus)
```

BF

```
%==========================================================================
%                               LINE DATA [Ohm]
%==========================================================================
%branch no sending  reciving  R(Ohm)     X(Ohm)
%==========================================================================
LD=[1 1 2 0.1170 0.0480
    2 2 3 0.1072 0.0440
    3 3 4 0.1645 0.0457
    4 4 5 0.1495 0.0415
    5 5 6 0.1495 0.0415
    6 6 7 0.3144 0.0540
    7 7 8 0.2096 0.0360
    8 8 9 0.3144 0.0540
    9 9 10 0.2096 0.0360
    10 10 11 0.1310 0.0225
    11 11 12 0.1048 0.0180
    12 3 13 0.1572 0.0270
    13 13 14 0.2096 0.0360
    14 14 15 0.1048 0.0180
    15 15 16 0.0524 0.0090
    16 6 17 0.1794 0.0498
    17 17 18 0.1645 0.0457
    18 18 19 0.2079 0.0473
    19 19 20 0.1890 0.0430
    20 20 21 0.1890 0.0430
    21 21 22 0.2620 0.0450
    22 22 23 0.2620 0.0450
    23 23 24 0.3144 0.0540
    24 24 25 0.2096 0.0360
    25 25 26 0.1310 0.0225
    26 26 27 0.1048 0.0180
    27 7 28 0.1572 0.0270
    28 28 29 0.1572 0.0270
    29 29 30 0.1572 0.0270
    30 10 31 0.1572 0.0270
    31 31 32 0.2096 0.0360
    32 32 33 0.1572 0.0270
    33 33 34 0.1048 0.0180];
%==========================================================================
%                          BUS DATA [kW and kVar]
%==========================================================================
  % bus no     activepower   reactivepower
BD1=[1  0   0
```

```
    2   230 142.5
    3   0   0
    4   230 142.5
    5   230 142.5
    6   0 0
    7   0   0
    8   230 142.5
    9   230 142.5
    10  0   0
    11  230 142.5
    12  137 84
    13  72  45
    14  72  45
    15  72  45
    16  13.5    7.5
    17  230 142.5
    18  230 142.5
    19  230 142.5
    20  230 142.5
    21  230 142.5
    22  230 142.5
    23  230 142.5
    24  230 142.5
    25  230 142.5
    26  230 142.5
    27  137 85
    28  75  48
    29  75  48
    30  75  48
    31  57  34.5
    32  57  34.5
    33  57  34.5
    34  57  34.5];
%===========================================================================
BD=[BD1(:,1) BD1(:,2) BD1(:,3)-Qcc];
%===========================================================================
br=length(LD);
no=length(BD);
%===========================================================================
MVAb=100;
KVb=11;
Zb=(KVb^2)/MVAb;
%===========================================================================
%                              Per Unit Values
%===========================================================================
```

```
R=(LD(:,4))/Zb;
XL=(LD(:,5))/Zb;
P=(BD(:,2))./(1000*MVAb);
Q=(BD(:,3))./(1000*MVAb);
%===========================================================================
%              Code for  bus-injection to branch-current matrix
%===========================================================================
bibc=zeros(size(LD,1),size(LD,1));
for i=1:size(LD,1)
    if LD(i,2)==1
      bibc(LD(i,3)-1,LD(i,3)-1)=1;
    else
       bibc(:,LD(i,3)-1)=bibc(:,LD(i,2)-1);
       bibc(LD(i,3)-1,LD(i,3)-1)=1;
    end
end
S=complex(P,Q);                                 % complex power
Vo=ones(size(LD,1),1);% initial bus votage% 10 change to specific data value
S(1)=[];
VB=Vo;
iteration=100;
% iteration=input('number of iteration : ');
%===========================================================================
for ip=1:iteration
    %=======================================================================
    %                          Backward Sweep
    %======================================================================
    I=conj(S./VB);                                      % injected current
    Z=complex(R,XL);                           %branch impedance
    ZD=diag(Z);                                        %makeing it diagonal
    IB=bibc*I;                                              %branch current
    %=======================================================================
    %                          Forward Sweep
    %=======================================================================
    TRX=bibc'*ZD*bibc;
    VB=Vo-TRX*I;
end
%===========================================================================
Vbus=[1;VB];
% display(Vbus);
% display(IB);
V_bus=abs(Vbus);
I_Line=abs(IB);
%===========================================================================
%                               Power Loss
```

```
%==========================================================================
Ibrp=[abs(IB) angle(IB)*180/pi];
PLL(1,1)=0;
QLL(1,1)=0;
% losses
for f2=1:size(LD,1)
    Pl(f2,1)=(Ibrp(f2,1)^2)*R(f2,1);
    Ql(f2,1)=X(f2,1)*(Ibrp(f2,1)^2);
    PLL(1,1)=PLL(1,1)+Pl(f2,1);
    QLL(1,1)=QLL(1,1)+Ql(f2,1);
end
%==========================================================================
Plosskw=(Pl)*100000;
Qlosskw=(Ql)*100000;
PLoss=(PLL)*100000;
QLoss=(QLL)*100000;
%==========================================================================
```

Bff

```
% clc
% clear all
%===========================================================================
%                               LINE DATA [Ohm]
%===========================================================================
%branch no sending  reciving  R(Ohm)     X(Ohm)
%===========================================================================
LD = [1 1 2 0.1170 0.0480
    2 2 3 0.1072 0.0440
    3 3 4 0.1645 0.0457
    4 4 5 0.1495 0.0415
    5 5 6 0.1495 0.0415
    6 6 7 0.3144 0.0540
    7 7 8 0.2096 0.0360
    8 8 9 0.3144 0.0540
    9 9 10 0.2096 0.0360
    10 10 11 0.1310 0.0225
    11 11 12 0.1048 0.0180
    12 3 13 0.1572 0.0270
    13 13 14 0.2096 0.0360
    14 14 15 0.1048 0.0180
    15 15 16 0.0524 0.0090
    16 6 17 0.1794 0.0498
    17 17 18 0.1645 0.0457
    18 18 19 0.2079 0.0473
    19 19 20 0.1890 0.0430
    20 20 21 0.1890 0.0430
    21 21 22 0.2620 0.0450
    22 22 23 0.2620 0.0450
    23 23 24 0.3144 0.0540
    24 24 25 0.2096 0.0360
    25 25 26 0.1310 0.0225
    26 26 27 0.1048 0.0180
    27 7 28 0.1572 0.0270
    28 28 29 0.1572 0.0270
    29 29 30 0.1572 0.0270
    30 10 31 0.1572 0.0270
    31 31 32 0.2096 0.0360
    32 32 33 0.1572 0.0270
    33 33 34 0.1048 0.0180];

%===========================================================================
%                          BUS DATA [kW and kVar]
```

```
%==========================================================================
  % bus no     activepower   reactivepower
BD1=[1  0 0
    2 230 142.5
    3 0 0
    4 230 142.5
    5 230 142.5
    6 0 0
    7 0 0
    8 230 142.5
    9 230 142.5
    10  0 0
    11  230 142.5
    12  137 84
    13  72  45
    14  72  45
    15  72  45
    16  13.5  7.5
    17  230 142.5
    18  230 142.5
    19  230 142.5
    20  230 142.5
    21  230 142.5
    22  230 142.5
    23  230 142.5
    24  230 142.5
    25  230 142.5
    26  230 142.5
    27  137 85
    28  75  48
    29  75  48
    30  75  48
    31  57  34.5
    32  57  34.5
    33  57  34.5
    34  57  34.5];
%==========================================================================
BD=[BD1(:,1) BD1(:,2) BD1(:,3)-Qcc1_Size_Kvar];
%==========================================================================
br=length(LD);
no=length(BD);
%==========================================================================
MVAb=100;
KVb=11;
Zb=(KVb^2)/MVAb;
```

```matlab
%===========================================================================
%                              Per Unit Values
%===========================================================================
R=(LD(:,4))/Zb;
X=(LD(:,5))/Zb;
P=(BD(:,2))./(1000*MVAb);
Q=(BD(:,3))./(1000*MVAb);
%===========================================================================
F=LD(:,2:3);
M=max(LD(:,2:3));
N=max(M);
f1=[1:N]';
for i=1:N
    g1=find(F(:,:)==i);
    h1(i)=length(g1);
end
k1(:,1)=f1;
k1(:,2)=h1';
cent=1;                                      % cent=input('central bus  ');
  %=========================================================================
  % this section of the code is to adjust line data to the standard
  %=========================================================================
  NLD=zeros(N,size(LD,2));
  c=find(LD(:,2:3)==cent);
  NLD=LD(c,:);
  LD(c,:)=[];
  t=find(k1(:,1)==cent);
  k1(t,2)=k1(t,2)-size(c,1);
  j=size(c,1);
  i=1;
  while sum(k1(:,2))>0
        c=[];
        b=[];
        t=[];
        [c e]=find(LD(:,2:3)==NLD(i,3));
        if size(c,2)~=0
           b=LD(c,:);
           LD(c,:)=[];
           t=find(k1(:,1)==NLD(i,3));
           k1(t,2)=k1(t,2)-(size(c,1)+1);
           d=find(b(:,3)==NLD(i,3));
           b(d,2:3)=[b(d,3),b(d,2)];
           NLD(j+1:j+size(c,1),:)=b;
           j=j+size(c,1);
        end
```

```
        i=i+1;
  end
  LD=sortrows(NLD,3);
  %==========================================================================
  % end the data is represented in standard format
%==========================================================================
%             Code for  bus-injection to branch-current matrix
%==========================================================================
bibc=zeros(size(LD,1),size(LD,1));
for i=1:size(LD,1)
    if LD(i,2)==1
      bibc(LD(i,3)-1,LD(i,3)-1)=1;
    else
       bibc(:,LD(i,3)-1)=bibc(:,LD(i,2)-1);
       bibc(LD(i,3)-1,LD(i,3)-1)=1;
    end
end
S=complex(P,Q);                                  % complex power
Vo=ones(size(LD,1),1);% initial bus votage% 10 change to specific data value
S(1)=[];
VB=Vo;
iteration=100;
% iteration=input('number of iteration : ');
%==========================================================================
for ip=1:iteration
    %======================================================================
    %                          Backward Sweep
    %=====================================================================
    I=conj(S./VB);                                          % injected current
    Z=complex(R,X);                            %branch impedance
    ZD=diag(Z);                                         %makeing it diagonal
    IB=bibc*I;                                               %branch current
    %======================================================================
    %                          Forward Sweep
    %======================================================================
    TRX=bibc'*ZD*bibc;
    VB=Vo-TRX*I;
end
%==========================================================================
Vbus=[1;VB];
% display(Vbus);
% display(IB);
V_bus=abs(Vbus);
I_Line=abs(IB);
%==========================================================================
```

```
%                                Power Loss
%==========================================================================
Ibrp=[abs(IB) angle(IB)*180/pi];
PLL(1,1)=0;
QLL(1,1)=0;
% losses
for f2=1:size(LD,1)
    Pl(f2,1)=(Ibrp(f2,1)^2)*R(f2,1);
    Ql(f2,1)=X(f2,1)*(Ibrp(f2,1)^2);
    PLL(1,1)=PLL(1,1)+Pl(f2,1);
    QLL(1,1)=QLL(1,1)+Ql(f2,1);
end
%==========================================================================
Plosskw=(Pl)*100000;
Qlosskw=(Ql)*100000;
PLoss=(PLL)*100000;
QLoss=(QLL)*100000;
%==========================================================================
```