Smart Distribution Management System for Smart-Transformer

Smart Distribution Transformer for Voltage Regulation and Frequency Support

Research Engineer -Mr. Indoopa Manamperi &
Ms. Lasanthika Dissawa
Team Leader -Prof. J.B. Ekanayake
Co-Investigators -Dr. P.J. Binduhewa
-Dr. S.G. Abeyratne
-Dr. Lilantha Samaranayake

Contents

Optimum power flow solving algorithm.

Representative equations for the optimization problem

Kron reduction technique was used to reduce the 4 wire distribution system to a 3 wire system[27]. It was assumed that neutral was grounded at multiple points. The radial power flow equations was represented by DistFlow equations[28].

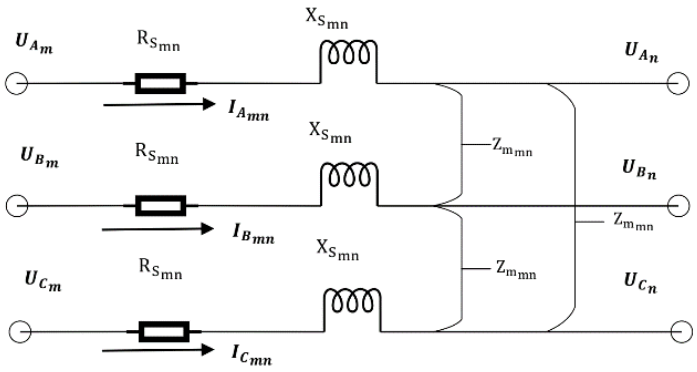

Figure 1 - Diagram of m-n line segment

For the m-n line segment shown in Figure 1, equations for phase A active and reactive power flows were written as in (1) and (2) respectively.

$$p_{A_{mn}} = \sum_{k:(j,k)\in E} p_{A_{nk}} + \mathrm{P}_{\mathrm{A_n}}^{\mathrm{C}} - \mathrm{P}_{\mathrm{A_n}}^{\mathrm{G}} + {P_A^{Cur}}_n + \mathrm{Re}\{\boldsymbol{I_{A_{mn}}}^* \times (\mathrm{R}_{\mathrm{s_{mn}}}\boldsymbol{I_{A_{mn}}} + \mathrm{R}_{\mathrm{m_{mn}}}\boldsymbol{I_{B_{mn}}} + \mathrm{R}_{\mathrm{m_{mn}}}\boldsymbol{I_{C_{mn}}})\} \quad (1)$$

$$q_{A_{mn}} = \sum_{k:(j,k)\in E} q_{A_{nk}} + \mathrm{Q}_{\mathrm{A_n}}^{\mathrm{C}} \pm Q_{A_n}^{G} + \mathrm{Im}\{\boldsymbol{I_{A_{mn}}}^* \times (\mathrm{R}_{\mathrm{s_{mn}}}\boldsymbol{I_{A_{mn}}} + \mathrm{R}_{\mathrm{m_{mn}}}\boldsymbol{I_{B_{mn}}} + \mathrm{R}_{\mathrm{m_{mn}}}\boldsymbol{I_{C_{mn}}})\} \quad (2)$$

where,

$\boldsymbol{I_{A_{mn}}}$, $p_{A_{mn}}$, $q_{A_{mn}}$ are the complex current, active and reactive powers flowing from node m to n in phase A

$\mathrm{P}_{\mathrm{A_n}}^{\mathrm{C}}$ and $\mathrm{Q}_{\mathrm{A_n}}^{\mathrm{C}}$ are the real and reactive power consumption of phase A of node n

$\mathrm{P}_{\mathrm{A_n}}^{\mathrm{G}}$, $Q_{A_n}^{G}$ and ${P_A^{Cur}}_n$ are the real and reactive power generation and active power curtailment from the PVs and inverters in phase A of node n.

Two new variables, $W_{A_m} = |\boldsymbol{U_{A_m}}|^2$ and $L_{A_{mn}} = |\boldsymbol{I_{A_{mn}}}|^2$ for every bus and every line were introduced. Then variables in non-convex terms ($U_A^R{}_n, I_A^R{}_{mn}, U_A^{Im}{}_n, I_A^{Im}{}_{mn}$) of equation (1) and (2) were replaced by values obtained from load flow. The exact equations used are given in Appendix A as equations A1 and A2.

Equation for phase A, the apparent power flow of m-n line segment is written as:

$$\left|\boldsymbol{I_{A_{mn}}}\right|^2\left|\boldsymbol{U_{A_m}}\right|^2 = p_{A_{mn}}^2 + q_{A_{mn}}^2 \quad (3)$$

Then with substitutions equation (3) was rewritten as:

$$W_{\boldsymbol{A_{mn}}} L_{\boldsymbol{A_m}} = p_{A_{mn}}^2 + q_{A_{mn}}^2 \quad (4)$$

Using Ohm's law, equation for relationship between phase A voltage magnitudes is expressed as in (5).

$$\boldsymbol{U_{A_m}} = \left(\boldsymbol{U_{A_n}} - \boldsymbol{I_{A_{mn}}}\mathrm{Z_S} - \boldsymbol{I_{B_{mn}}}\mathrm{Z_m} - \boldsymbol{I_{C_{mn}}}\mathrm{Z_m}\right) \quad (5)$$

where $\boldsymbol{U_{A_m}}$ and $\boldsymbol{U_{A_n}}$ are the complex voltage at node m and n of phase A and are given by $\boldsymbol{U_{A_m}} = U_A^R{}_m + jU_A^{Im}{}_m$ and $\boldsymbol{U_{A_n}} = U_A^R{}_n + jU_A^{Im}{}_n$

Similarly, $\boldsymbol{I_{A_m}} = I_A^R{}_m + jI_A^{Im}{}_m$; $\boldsymbol{I_{B_m}} = I_B^R{}_m + jI_B^{Im}{}_m$ and $\boldsymbol{I_{C_m}} = I_C^R{}_m + jI_C^{Im}{}_m$

To keep the variables in optimization program independent of angle and to make equations linear, the square of equation (5) was used for optimization after some substitutions. Variables in non-convex terms ($U_A^R{}_n, I_A^R{}_{mn}, U_A^{Im}{}_n, I_A^{Im}{}_{mn}$) of square equation (5) were replaced by values obtained from load flow. The exact equation used is given in Appendix A as equation (A3).

Constraints

Voltage constraints

The L-N voltage was maintained within ±6% from the nominal value[29]. The following constraint was used to restrict the squared magnitude of voltages within the allowable range. Voltage variation capability of smart transformer was included by keeping the following inequality for transformer bus.

$$216^2 \leq \left|\boldsymbol{U_{A_m}}\right|^2 \leq 244^2 \quad (6)$$

Inverter active power curtailment constraints

Capability to curtail full active power generation is provided to inverters. However, the curtailment of active power was minimized using the objective function.

$$0 \leq P_{A_n}^{Cur} \leq \mathrm{P}_{\mathrm{A_n}}^{\mathrm{G}} \tag{7}$$

Cost functions

To minimize the overall cost incurred for maintaining the voltage within permissible range, the minimization of the line losses and the inverter active power curtailment were considered.

Cost of line losses

Line losses include the cost of Ohmic losses incurred in 3 phase cables and neutral. Here, with the available variables, only the losses in 3 phase lines were considered for the objective as:

$$\sum_{\forall i \in E}(\mathrm{R_{s_i}}) \times \left|\boldsymbol{I_{A_i}}\right|^2 + \sum_{\forall i \in E}(\mathrm{R_{s_i}}) \times \left|\boldsymbol{I_{B_i}}\right|^2 + \sum_{\forall i \in E}(\mathrm{R_{s_i}}) \times \left|\boldsymbol{I_{C_i}}\right|^2$$

With $L_{A,B,C_i} = \left|\boldsymbol{I_{A,B,C_i}}\right|^2$

$$\sum_{\forall i \in E}\left(\mathrm{R_{s_i}}\right) \times L_{A_i} + \sum_{\forall i \in E}\left(\mathrm{R_{s_i}}\right) \times L_{B_i} + \sum_{\forall i \in E}\left(\mathrm{R_{s_i}}\right) \times L_{C_i} \tag{8}$$

Cost of power curtailment

This term includes the power curtailment of every single-phase inverter, and it is expressed as in (9).

$$\sum_{\forall i \in N_A} P_A^{Cur}{}_i + \sum_{\forall i \in N_B} P_B^{Cur}{}_i + \sum_{\forall i \in N_C} P_C^{Cur}{}_i \tag{9}$$

Convex optimization problem

Nonlinear equality constraint in (4) is non-convex and it is relaxed as in equation (10).

$$L_{A_{mn}} \geq \frac{p_{A_{mn}}^2}{W_{A_m}} + \frac{q_{A_{mn}}^2}{W_{A_m}} \tag{10}$$

Relaxed constraint (10) represents a second order cone. Then additional linear inequality constraint as defined by equation (11) was used to add linear cuts to SOC relaxation from the

second iteration onwards. Solutions are converged to a more optimum value in the first iteration and the constants of equations in second iteration are calculated from solutions generated in the first iteration. This upper bound is reduced in each iteration, gradually increasing the exactness of solutions. Squared current magnitude generated from a load flow using previous iteration's data is used as a part of the upper bound.

$$L_{A_{mn}} \leq \left| \boldsymbol{I_{A_{mn\ (Load\ flow)}}} \right|^2 + \frac{10000}{10^{iteration}} \tag{11}$$

Here, $\left| \boldsymbol{I_{A_{mn\ (Load\ flow)}}} \right|^2$ is the squared magnitude of current value generated from load flow from previous iteration using OpenDSS. The term $\frac{10000}{10^{iteration}}$ in (11) is a constant for a particular iteration. It reduces to 100 in third iteration. This constant was chosen empirically to allow larger search space in initial iterations. Additional search space from $\boldsymbol{I_{A_{mn\ (Load\ flow)}}}$ was reduced 10 times for subsequent iteration using $\frac{10000}{10^{iteration}}$ term. This acts as an increasingly tightening cutting planes to SOC relaxations in each phase. Because of this upper bound, sufficiently exact solution can be obtained after the third iteration of optimization program.

Equations (10) and (11) were repeated for other 2 phases.

Then the optimization problem was represented by cost functions (8) and (9) as below subjected to the constraints given by (6), (7), (10), (11), A1, A2 and A3 for all 3 phases.

$$\text{minimize } \frac{0.11}{2} \times \left\{ \sum_{\forall i \in E} \left(\mathrm{R_{s_i}} \right) \times L_{A_i} + \sum_{\forall i \in E} \left(\mathrm{R_{s_i}} \right) \times L_{B_i} + \sum_{\forall i \in E} \left(\mathrm{R_{s_i}} \right) \times L_{C_i} \right\} + \frac{0.11}{2} \times \left\{ \sum_{\forall i \in N_A} {P_A^{Cur}}_i + \sum_{\forall i \in N_B} {P_B^{Cur}}_i + \sum_{\forall i \in N_C} {P_C^{Cur}}_i \right\} \tag{12}$$

Solution method

Three or fewer runs of the optimization program (equation (12)) was executed while reducing the upper bound of squared current (equation (11)). Optimization is performed for much larger convex terms with variables ($W_{A_m}, L_{A_{mn}}, p_{A_{mn}}, q_{A_{mn}}, Q_{A_n}^G, {P_A^{Cur}}_n$) while variables in non-convex terms ($U_{A\ n}^R, I_{A\ mn}^R, U_{A}^{Im}{}_n, I_{A}^{Im}{}_{mn}$) acting as constants for the considered iteration of optimization

program. When number of iterations increase, updating terms become closer to the solution generated by the optimization program. The flow chart of the algorithm is provided in Figure 2. First, a load flow is performed using OpenDSS with zero curtailment of active power generation of inverters. Then optimization problem is solved. Constants in equations A1, A2, and A3 are replaced from the respective values of load flow. Then new load flow is performed after updating curtailment values generated from optimization problem. Then after updating equation (11), the optimization problem is solved after updating constants in equations A1, A2, and A3. The repetition of 3 iterations of this process was sufficient to obtain sufficiently exact solutions. This solution approach closely relates to the sequential convex programing[30].

This algorithm was implemented in Matlab with CVX optimization toolbox[31] and SDPT3 as the solver. OpenDSS was used to run the load flow and update variables in non-convex terms.

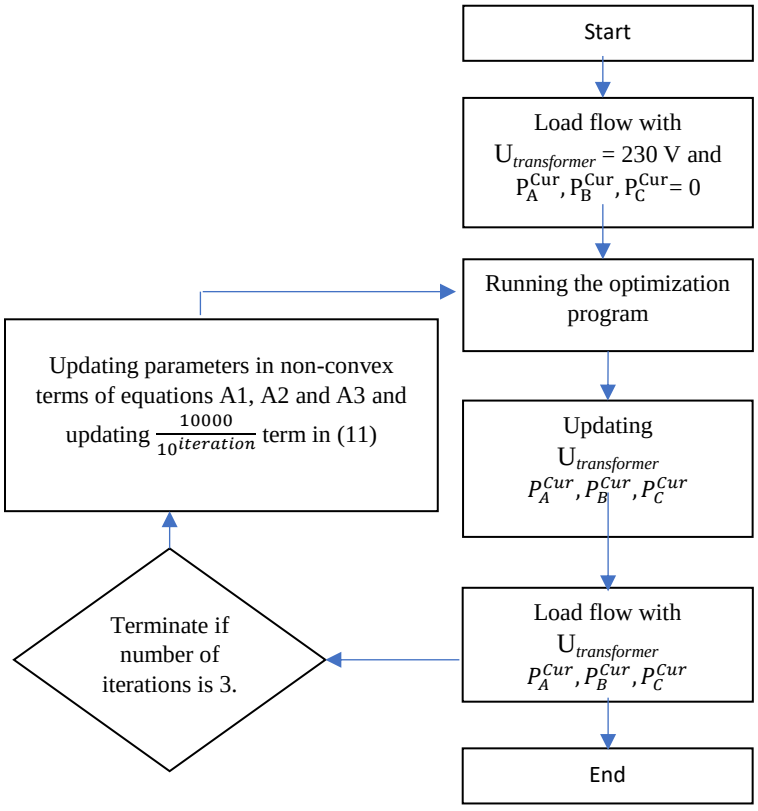

Figure 2 - flow chart of the OPF solving algorithm

Matlab function for newly developed optimum power flow solving method

```
function [ V_A,Q_A, Q_B, Q_C, bat_A, bat_B, bat_C]= Europian_test_case_SOCP()

clc;

clear;

n=906;

% data acquiring

loaddata_A=zeros(906,3);

loaddata_B=zeros(906,3);

loaddata_C=zeros(906,3);

gendata_A=zeros(906,1);

gendata_B=zeros(906,1);

gendata_C=zeros(906,1);

Loads = evalin('base','Loads');

 Loadprofile1=
ReadLoadprofile(strcat('C:\Users\Indoopa\Downloads\European_LV_Test_Feeder_v2\European_LV_CS
V\Load Profiles\Load_profile_',num2str(1),'.csv'));

 mult=double(Loadprofile1.('mult'));

 L=double(Loads.('kW'));

 %[~,m_i]=max(mult);

 m_i=721;

 keySet = {'A','B','C',};

 valueSet = [1,2,3];

 phase = containers.Map(keySet,valueSet);

for i=1 : length(L)
```

```
Loadprofile=
ReadLoadprofile(strcat('C:\Users\Indoopa\Downloads\European_LV_Test_Feeder_v2\European_LV_CSV\Load Profiles\Load_profile_',num2str(i),'.csv'));

 if char(table2array(Loads(i,4)))=='A'

    loaddata_A(double(table2array(Loads(i,3))),1)=double(table2array(Loads(i,3)));


loaddata_A(double(table2array(Loads(i,3))),2)=1000*double(table2array(Loads(i,8)))*double(table2array(Loadprofile(m_i,2)));

    pf=double(table2array(Loads(i,9)))

     gendata_A(double(table2array(Loads(i,3))))=1000*double(table2array(Loads(i,11)));


loaddata_A(double(table2array(Loads(i,3))),3)=loaddata_A(double(table2array(Loads(i,3))),2)*tan(acos(pf))

    loaddata_A(double(table2array(Loads(i,3))),2)=loaddata_A(double(table2array(Loads(i,3))),2);

 elseif char(table2array(Loads(i,4)))=='B'

   loaddata_B(double(table2array(Loads(i,3))),1)=double(table2array(Loads(i,3)));


loaddata_B(double(table2array(Loads(i,3))),2)=1000*double(table2array(Loads(i,8)))*double(table2array(Loadprofile(m_i,2)));

    pf=double(table2array(Loads(i,9)));

     gendata_B(double(table2array(Loads(i,3))))=1000*double(table2array(Loads(i,11)));


loaddata_B(double(table2array(Loads(i,3))),3)=loaddata_B(double(table2array(Loads(i,3))),2)*tan(acos(pf));

    loaddata_B(double(table2array(Loads(i,3))),2)=loaddata_B(double(table2array(Loads(i,3))),2);

 else

    loaddata_C(double(table2array(Loads(i,3))),1)=double(table2array(Loads(i,3)));


loaddata_C(double(table2array(Loads(i,3))),2)=1000*double(table2array(Loads(i,8)))*double(table2array(Loadprofile(m_i,2)));
```

```
    pf=double(table2array(Loads(i,9)));
     gendata_C(double(table2array(Loads(i,3))))=1000*double(table2array(Loads(i,11)));

loaddata_C(double(table2array(Loads(i,3))),3)=loaddata_C(double(table2array(Loads(i,3))),2)*tan(acos(
pf));
    loaddata_C(double(table2array(Loads(i,3))),2)=loaddata_C(double(table2array(Loads(i,3))),2);
  end
end


   Q_load_A=zeros(length(loaddata_A(:,1)),1);
   Q_load_B=zeros(length(loaddata_B(:,1)),1);
   Q_load_C=zeros(length(loaddata_C(:,1)),1);
   P_load_A=zeros(length(loaddata_A(:,1)),1);
   P_load_B=zeros(length(loaddata_B(:,1)),1);
   P_load_C=zeros(length(loaddata_C(:,1)),1);
 for (i=1:n)
       Q_load_A(i)=loaddata_A(i,3);
      Q_load_B(i)=loaddata_B(i,3);
      Q_load_C(i)=loaddata_C(i,3);


      P_load_A(i)=loaddata_A(i,2)-gendata_A(i);
      P_load_B(i)=loaddata_B(i,2)-gendata_B(i);
      P_load_C(i)=loaddata_C(i,2)-gendata_C(i);
   %end
end
linedata=zeros(n-1,7);
```

```
LineCodes = evalin('base','LineCodes') ;

Lines = evalin('base','Lines');

l_code=string(Lines.('LineCode'));


for i=1 : (n-1)

   code=find(ismember(LineCodes{:,1},l_code(i)));

   l=double(table2array(Lines(i,5)))/1000;

          %strcat('New Line.',num2str(double(table2array(Lines(i,1)))), ' Bus1=',num2str(), '
Bus2=',num2str(double((table2array(Lines(i,3)))),' R1=',num2str(), '
X1=',num2str(l*double(table2array(LineCodes(code,4)))), '
R0=',num2str(l*double(table2array(LineCodes(code,5)))),'
X0=',num2str(l*double(table2array(LineCodes(code,6)))), ' C1=0 C0=0 Length=1' );

   linedata(i,1)=i;

   linedata(i,2)=double(table2array(Lines(i,2)));

   linedata(i,3)=double(table2array(Lines(i,3)));

   linedata(i,4)=l*double(table2array(LineCodes(code,3)));%r1

   linedata(i,5)=l*double(table2array(LineCodes(code,4)));%x1

   linedata(i,6)=l*double(table2array(LineCodes(code,5)));%xo

   linedata(i,7)=l*double(table2array(LineCodes(code,6)));%xo

end

l=zeros(n-1,1);

nl=linedata(:,2);% left node

nr=linedata(:,3);% right node

r1=linedata(:,4);

ro=linedata(:,6);

x1=linedata(:,5);

xo=linedata(:,7);
```

```
rs=(2*r1+ro)/3;  % Obtaining impedance from sequence positive and zero sequence impedance values
xs=(2*x1+xo)/3;
rm=(ro-r1)/3; %obtaining mutual impedance
xm=(xo-x1)/3;
itr=0;
%starting three iterations of optimization program
while (itr<1)
   tic
   fprintf(1,'Computing the optimal solution ...');
   cvx_begin %starting optimization using CVX
      variables p_A(n-1)  q_A(n-1) Q_A(n) v_A(n) V_start p_B(n-1)  q_B(n-1) Q_B(n) v_B(n) p_C(n-1) q_C(n-1) Q_C(n) v_C(n) bat_A(n) bat_B(n) bat_C(n) l_A(n-1) l_B(n-1) l_C(n-1)
       whos
       minimize norm(l_A)+norm(l_B)+norm(l_C) %objective function
       subject to
     216*216<=V_start<=244*244  % smart transformer voltage variation
    for i=1:n
        216*216<=v_A(i)<=244*244
        216*216<=v_B(i)<=244*244
        216*216<=v_C(i)<=244*244
     end
     for i=1:n
         if(gendata_A(i)>0)
          -1000<=Q_A(i)<=1000
          0<=bat_A(i)<=000
```

```
    else
        Q_A(i)==0
       bat_A(i)==0
    end
    if(gendata_B(i)>0)
     -1000<=Q_B(i)<=1000
     0<=bat_B(i)<=000
    else
       Q_B(i)==0
       bat_B(i)==0
    end
    if(gendata_C(i)>0)
     -1000<=Q_C(i)<=1000
     0<=bat_C(i)<=000
    else
       Q_C(i)==0
       bat_C(i)==0
   end
end
j=find(nl==1);
for k=1:length(j)
     v_A(nl(j(k)))==V_start
     v_B(nl(j(k)))==V_start
     v_C(nl(j(k)))==V_start
```

```
        quad_over_lin(p_A(j(k)),v_A(nl(j(k))))+quad_over_lin(q_A(j(k)),v_A(nl(j(k))))<=l_A(j(k))

        quad_over_lin(p_B(j(k)),v_B(nl(j(k))))+quad_over_lin(q_B(j(k)),v_B(nl(j(k))))<=l_B(j(k))

        quad_over_lin(p_C(j(k)),v_C(nl(j(k))))+quad_over_lin(q_C(j(k)),v_C(nl(j(k))))<=l_C(j(k))

 %equations for voltage magnitude

        v_A(nr(j(k)))==V_start-2*(rs(j(k))*p_A(j(k))+xs(j(k))*q_A(j(k)))-
2*(rm(j(k))*p_B(j(k))+xm(j(k))*q_B(j(k)))-
2*(rm(j(k))*p_C(j(k))+xm(j(k))*q_C(j(k)))+((rs(j(k)))^2+xs(j(k))^2)*l_A(j(k))+((rm(j(k)))^2+xm(j(k))^2)*l_B(j
(k))+((rm(j(k)))^2+xm(j(k))^2)*l_C(j(k))

        v_B(nr(j(k)))==V_start -2*(rs(j(k))*p_B(j(k))+xs(j(k))*q_B(j(k)))-
2*(rm(j(k))*p_A(j(k))+xm(j(k))*q_A(j(k)))-
2*(rm(j(k))*p_C(j(k))+xm(j(k))*q_C(j(k)))+((rs(j(k)))^2+xs(j(k))^2)*l_B(j(k))+((rm(j(k)))^2+xm(j(k))^2)*l_A(j
(k))+((rm(j(k)))^2+xm(j(k))^2)*l_C(j(k))

        v_C(nr(j(k)))==V_start-2*(rs(j(k))*p_C(j(k))+xs(j(k))*q_C(j(k)))-
2*(rm(j(k))*p_B(j(k))+xm(j(k))*q_B(j(k)))-
2*(rm(j(k))*p_A(j(k))+xm(j(k))*q_A(j(k)))+((rs(j(k)))^2+xs(j(k))^2)*l_C(j(k))+((rm(j(k)))^2+xm(j(k))^2)*l_B(j
(k))+((rm(j(k)))^2+xm(j(k))^2)*l_A(j(k))



        e=find(nl==nr(j(k)));

        if(length(e)==0) % end node

           [j(k),nr(j(k))];

 %equations for active and reactive power flows
p_A((j(k)))==P_load_A(nr(j(k)))+bat_A(nr(j(k)))+rs(j(k))*l_A(j(k))+rm(j(k))*l_B(j(k))+rm(j(k))*l_C(j(k))
q_A((j(k)))==Q_A(nr(j(k)))+Q_load_A(nr(j(k)))+xs(j(k))*l_A(j(k))+xm(j(k))*l_B(j(k))+xm(j(k))*l_C(j(k))


p_B((j(k)))==P_load_B(nr(j(k)))+bat_B(nr(j(k)))+rs(j(k))*l_B(j(k))+rm(j(k))*l_A(j(k))+rm(j(k))*l_C(j(k))
q_B((j(k)))==Q_B(nr(j(k)))+Q_load_B(nr(j(k)))+xs(j(k))*l_B(j(k))+xm(j(k))*l_A(j(k))+xm(j(k))*l_C(j(k))


p_C((j(k)))==P_load_C(nr(j(k)))+bat_C(nr(j(k)))+rs(j(k))*l_C(j(k))+rm(j(k))*l_B(j(k))+rm(j(k))*l_A(j(k))
q_C((j(k)))==Q_C(nr(j(k)))+Q_load_C(nr(j(k)))+xs(j(k))*l_C(j(k))+xm(j(k))*l_B(j(k))+xm(j(k))*l_A(j(k))

           continue;

        end
```

```
p_A(j(k))==P_load_A(nr(j(k)))+sum(p_A(e))+bat_A(nr(j(k)))+rs(j(k))*l_A(j(k))+rm(j(k))*l_B(j(k))+rm(j(k))*l_C(j(k))
q_A(j(k))==Q_A(nr(j(k)))+Q_load_A(nr(j(k)))+sum(q_A(e))+xs(j(k))*l_A(j(k))+xm(j(k))*l_B(j(k))+xm(j(k))*l_C(j(k))


p_B(j(k))==P_load_B(nr(j(k)))+sum(p_B(e))+bat_B(nr(j(k)))+rs(j(k))*l_B(j(k))+rm(j(k))*l_C(j(k))+rm(j(k))*l_A(j(k))
q_B(j(k))==Q_B(nr(j(k)))+Q_load_B(nr(j(k)))+sum(q_B(e))+xs(j(k))*l_B(j(k))+xm(j(k))*l_C(j(k))+xm(j(k))*l_A(j(k))


p_C(j(k))==P_load_C(nr(j(k)))+sum(p_C(e))+bat_C(nr(j(k)))+rs(j(k))*l_C(j(k))+rm(j(k))*l_B(j(k))+rm(j(k))*l_A(j(k))
q_C(j(k))==Q_C(nr(j(k)))+Q_load_C(nr(j(k)))+sum(q_C(e))+xs(j(k))*l_C(j(k))+xm(j(k))*l_B(j(k))+xm(j(k))*l_A(j(k))

    end

    for i=2:n

     j=find(nl==i);

     if(length(j)==0)

            continue;

     end

       for k=1:length(j)

       nl(j(k));

       quad_over_lin(p_A(j(k)),v_A(nl(j(k))))+quad_over_lin(q_A(j(k)),v_A(nl(j(k))))<=l_A(j(k))

       quad_over_lin(p_B(j(k)),v_B(nl(j(k))))+quad_over_lin(q_B(j(k)),v_B(nl(j(k))))<=l_B(j(k))

       quad_over_lin(p_C(j(k)),v_C(nl(j(k))))+quad_over_lin(q_C(j(k)),v_C(nl(j(k))))<=l_C(j(k))


       v_A(nr(j(k)))==v_A(nl(j(k)))-2*(rs(j(k))*p_A(j(k))+xs(j(k))*q_A(j(k)))-
2*(rm(j(k))*p_B(j(k))+xm(j(k))*q_B(j(k)))-
2*(rm(j(k))*p_C(j(k))+xm(j(k))*q_C(j(k)))+((rs(j(k)))^2+xs(j(k))^2)*l_A(j(k))+((rm(j(k)))^2+xm(j(k))^2)*l_B(j(k))+((rm(j(k)))^2+xm(j(k))^2)*l_C(j(k))
```

```
        v_B(nr(j(k)))==v_B(nl(j(k))) -2*(rs(j(k))*p_B(j(k))+xs(j(k))*q_B(j(k)))-
2*(rm(j(k))*p_A(j(k))+xm(j(k))*q_A(j(k)))-
2*(rm(j(k))*p_C(j(k))+xm(j(k))*q_C(j(k)))+((rs(j(k)))^2+xs(j(k))^2)*l_B(j(k))+((rm(j(k)))^2+xm(j(k))^2)*l_A(j
(k))+((rm(j(k)))^2+xm(j(k))^2)*l_C(j(k))

        v_C(nr(j(k)))==v_C(nl(j(k)))-2*(rs(j(k))*p_C(j(k))+xs(j(k))*q_C(j(k)))-
2*(rm(j(k))*p_B(j(k))+xm(j(k))*q_B(j(k)))-
2*(rm(j(k))*p_A(j(k))+xm(j(k))*q_A(j(k)))+((rs(j(k)))^2+xs(j(k))^2)*l_C(j(k))+((rm(j(k)))^2+xm(j(k))^2)*l_B(j
(k))+((rm(j(k)))^2+xm(j(k))^2)*l_A(j(k))

        e=find(nl==nr(j(k)));

        if(length(e)==0) % end node

           [j(k),nr(j(k))];

p_A((j(k)))==P_load_A(nr(j(k)))+bat_A(nr(j(k)))+rs(j(k))*l_A(j(k))+rm(j(k))*l_B(j(k))+rm(j(k))*l_C(j(k))
q_A((j(k)))==Q_A(nr(j(k)))+Q_load_A(nr(j(k)))+xs(j(k))*l_A(j(k))+xm(j(k))*l_B(j(k))+xm(j(k))*l_C(j(k))


p_B((j(k)))==P_load_B(nr(j(k)))+bat_B(nr(j(k)))+rs(j(k))*l_B(j(k))+rm(j(k))*l_A(j(k))+rm(j(k))*l_C(j(k))
q_B((j(k)))==Q_B(nr(j(k)))+Q_load_B(nr(j(k)))+xs(j(k))*l_B(j(k))+xm(j(k))*l_A(j(k))+xm(j(k))*l_C(j(k))


p_C((j(k)))==P_load_C(nr(j(k)))+bat_C(nr(j(k)))+rs(j(k))*l_C(j(k))+rm(j(k))*l_B(j(k))+rm(j(k))*l_A(j(k))
q_C((j(k)))==Q_C(nr(j(k)))+Q_load_C(nr(j(k)))+xs(j(k))*l_C(j(k))+xm(j(k))*l_B(j(k))+xm(j(k))*l_A(j(k))

           continue;

        end


p_A(j(k))==P_load_A(nr(j(k)))+sum(p_A(e))+bat_A(nr(j(k)))+rs(j(k))*l_A(j(k))+rm(j(k))*l_B(j(k))+rm(j(k))*l_
C(j(k))
q_A(j(k))==Q_A(nr(j(k)))+Q_load_A(nr(j(k)))+sum(q_A(e))+xs(j(k))*l_A(j(k))+xm(j(k))*l_B(j(k))+xm(j(k))*l_
C(j(k))

p_B(j(k))==P_load_B(nr(j(k)))+sum(p_B(e))+bat_B(nr(j(k)))+rs(j(k))*l_B(j(k))+rm(j(k))*l_C(j(k))+rm(j(k))*l_
A(j(k))
q_B(j(k))==Q_B(nr(j(k)))+Q_load_B(nr(j(k)))+sum(q_B(e))+xs(j(k))*l_B(j(k))+xm(j(k))*l_C(j(k))+xm(j(k))*l_
A(j(k))

p_C(j(k))==P_load_C(nr(j(k)))+sum(p_C(e))+bat_C(nr(j(k)))+rs(j(k))*l_C(j(k))+rm(j(k))*l_B(j(k))+rm(j(k))*l_
A(j(k))
q_C(j(k))==Q_C(nr(j(k)))+Q_load_C(nr(j(k)))+sum(q_C(e))+xs(j(k))*l_C(j(k))+xm(j(k))*l_B(j(k))+xm(j(k))*l_A
(j(k))
```

```
        end
       end
      cvx_end
   fprintf(1,'Done! \n');
   % Display results
   disp('------------------------------------------------------------------------');
   disp('line currents');
    disp('line voltages');
   disp(sqrt(v_A));
   disp('p');
   disp(p_A);
   disp('q');
   disp(q_A);
   disp('Q');
   disp(Q_A);
     x=[1:n];
       plot(x,sqrt(v_A),'^',x,sqrt(v_B),'^',x,sqrt(v_C),'^');
     xlabel('Node')
    ylabel('Voltage(V)')
    legend({'V_A','V_B','V_C'});
     itr=itr+1;
   if(itr>0)
   break;
   end
end
```

```
      toc
    V_A=sqrt(v_A);
    correctness1=0;correctness2=0;correctness3=0;
   for i=1:n-1
      correctness1=correctness1+ abs( p_A(i)^2/v_A(i)+q_A(i)^2/v_A(i)-l_A(i));
      correctness2=correctness2+ abs( p_B(i)^2/v_B(i)+q_B(i)^2/v_B(i)-l_B(i));
      correctness3=correctness3+ abs( p_C(i)^2/v_C(i)+q_C(i)^2/v_C(i)-l_C(i));
    end
end
```

Matlab function for load flow using OpenDSS

```
function [S,I,VMSi_A,VMSi_B,VMSi_C]= Europian_test_case_OpenDss_version2(V_A,Q_A, Q_B, Q_C,
bat_A, bat_B, bat_C)

Lines = evalin('base','Lines');

n=length(table2array(Lines(:,2)))+1;

load_column=11;

V_A=230*ones(n);Q_A=zeros(n); Q_B=zeros(n); Q_C=zeros(n); bat_A=zeros(n); bat_B=zeros(n);
bat_C=zeros(n);

loaddata_A=zeros(n,3);

loaddata_B=zeros(n,3);

loaddata_C=zeros(n,3);

gendata_A=zeros(n,1);

gendata_B=zeros(n,1);

gendata_C=zeros(n,1);


Loads = evalin('base','Loads');

Loadprofile1=
ReadLoadprofile(strcat('C:\Users\Indoopa\Downloads\European_LV_Test_Feeder_v2\European_LV_CS
V\Load Profiles\Load_profile_',num2str(1),'.csv'));

 mlt=double(Loadprofile1.('mult'));

 L=double(Loads.('kW'));

 m_i=7;

for i=1 : length(L)

 Loadprofile=
ReadLoadprofile(strcat('C:\Users\Indoopa\Downloads\European_LV_Test_Feeder_v2\European_LV_CS
V\Load Profiles\Load_profile_',num2str(i),'.csv'));

  if char(table2array(Loads(i,4)))=='A'

    loaddata_A(double(table2array(Loads(i,3))),1)=double(table2array(Loads(i,3)));
```

```
loaddata_A(double(table2array(Loads(i,3))),2)=1000*double(table2array(Loads(i,8)))*double(table2array(Loadprofile(m_i,2)));

    pf=double(table2array(Loads(i,9)));

    gendata_A(double(table2array(Loads(i,3))))=0.9*1000*double(table2array(Loads(i,load_column)));


loaddata_A(double(table2array(Loads(i,3))),3)=loaddata_A(double(table2array(Loads(i,3))),2)*tan(acos(pf));


 elseif char(table2array(Loads(i,4)))=='B'

   loaddata_B(double(table2array(Loads(i,3))),1)=double(table2array(Loads(i,3)));


loaddata_B(double(table2array(Loads(i,3))),2)=1000*double(table2array(Loads(i,8)))*double(table2array(Loadprofile(m_i,2)));

    gendata_B(double(table2array(Loads(i,3))))=0.9*1000*double(table2array(Loads(i,load_column)));

    pf=double(table2array(Loads(i,9)));

loaddata_B(double(table2array(Loads(i,3))),3)=loaddata_B(double(table2array(Loads(i,3))),2)*tan(acos(pf));

    %
gendata_B(double(table2array(Loads(i,3))))=gen*loaddata_B(double(table2array(Loads(i,3))),2)/3500;;

  elseif char(table2array(Loads(i,4)))=='C'

    loaddata_C(double(table2array(Loads(i,3))),1)=double(table2array(Loads(i,3)));


loaddata_C(double(table2array(Loads(i,3))),2)=1000*double(table2array(Loads(i,8)))*double(table2array(Loadprofile(m_i,2)));

    gendata_C(double(table2array(Loads(i,3))))=0.9*1000*double(table2array(Loads(i,load_column)));

    pf=double(table2array(Loads(i,9)));

loaddata_C(double(table2array(Loads(i,3))),3)=loaddata_C(double(table2array(Loads(i,3))),2)*tan(acos(pf));
```

```
    %
endata_C(double(table2array(Loads(i,3))))=gen*loaddata_C(double(table2array(Loads(i,3))),2)/3500;;
  else
     pause;
  end
end
linedata=zeros(n-1,7);
LineCodes = evalin('base','LineCodes') ;
l_code=string(Lines.('LineCode'));
l_length=zeros(length(l_code),1)
for i=1 : (n-1)
    code=find(ismember(LineCodes{:,1},l_code(i)));
    l=double(table2array(Lines(i,5)))/1000;
    linedata(i,1)=i;
    linedata(i,2)=double(table2array(Lines(i,2)));
    linedata(i,3)=double(table2array(Lines(i,3)));
    linedata(i,4)=l*double(table2array(LineCodes(code,3)));%r1
    linedata(i,5)=l*double(table2array(LineCodes(code,4)));%x1
    linedata(i,6)=l*double(table2array(LineCodes(code,5)));%ro
    linedata(i,7)=l*double(table2array(LineCodes(code,6)));%xo
end
    linedata(1,5)=linedata(1,5)+0.0150; % adding transformer reactance to line 1
    linedata(1,4)=linedata(1,4)+0.00150; % adding transformer resistance to line 1
  DSSObj = actxserver('OpenDSSEngine.DSS');
Start=DSSObj.Start(0);
DSSText = DSSObj.Text;
```

```
DSSText.Command = 'clear';

DSSText.Command ='New Circuit.Simple';

DSSText.Command = 'set defaultbasefreq=50 ';

DSSText.Command =strcat('Edit Vsource.Source BasekV=',num2str(sqrt(3)*V_A(1)/1000),' pu=1.00
ISC3=3000000000000 ISC1=250000000000') ;

        %adding lines to OpenDSS

          DSSText.Command =strcat('New Line.1 Bus1=SourceBus.1.2.3.0 ', ' Bus2=',num2str(linedata(1,3)
),'.1.2.3.0  R1=',num2str(linedata(1,4)), ' X1=',num2str(linedata(1,5)), ' R0=',num2str(linedata(1,6)),'
X0=',num2str(linedata(1,7)), ' C1=0 C0=0 Length=1' );

for i=2 : length(linedata(:,1))

            DSSText.Command =strcat('New Line.',num2str(i), ' Bus1=',num2str(linedata(i,2)), '.1.2.3.0
Bus2=',num2str(linedata(i,3) ),'.1.2.3.0  R1=',num2str(linedata(i,4)), ' X1=',num2str(linedata(i,5)), '
R0=',num2str(linedata(i,6)),' X0=',num2str(linedata(i,7)), ' C1=0 C0=0 Length=1' );

        end

                              DSSText.Command ='New EnergyMeter.M1 Element=Line.1';

          %adding loads to OpenDSS

        for i=1 : length(Loads.('kW'))


           if char(table2array(Loads(i,4)))=='A'


               DSSText.Command =strcat('New Load.', num2str(i), ' Bus1=',
num2str(loaddata_A(double(table2array(Loads(i,3))),1)),'.1.0 Model=1 kV=0.23 phase=1', ' kW=',
num2str(loaddata_A(double(table2array(Loads(i,3))),2)/1000-
gendata_A(double(table2array(Loads(i,3))))/1000+bat_A(double(table2array(Loads(i,3))))/1000), ' kVar='
,num2str(loaddata_A(double(table2array(Loads(i,3))),3)/1000+Q_A(double(table2array(Loads(i,3))))/100
0));%Q_A(i)/1000));%

                DSSText.Command =strcat( ' AddBusMarker
Bus=',num2str(loaddata_A(double(table2array(Loads(i,3))),1)),' code=3 color=Red size=20');

           elseif char(table2array(Loads(i,4)))=='B'
```

```
            DSSText.Command =strcat('New Load.', num2str(i), ' Bus1=',
num2str(loaddata_B(double(table2array(Loads(i,3))),1)),'.2.0 Model=1 kV=0.23 phase=1', ' kW=',
num2str(loaddata_B(double(table2array(Loads(i,3))),2)/1000-
gendata_B(double(table2array(Loads(i,3))))/1000+bat_B(double(table2array(Loads(i,3))))/1000), ' kVar='
,num2str(loaddata_B(double(table2array(Loads(i,3))),3)/1000+Q_B(double(table2array(Loads(i,3))))/100
0));%Q_B(i)/1000));%

            DSSText.Command =strcat( ' AddBusMarker
Bus=',num2str(loaddata_B(double(table2array(Loads(i,3))),1)),' code=3 color=Green size=30');

          elseif char(table2array(Loads(i,4)))=='C'

            DSSText.Command =strcat('New Load.', num2str(i), ' Bus1=',
num2str(loaddata_C(double(table2array(Loads(i,3))),1)),'.3.0 Model=1 kV=0.23 phase=1', ' kW=',
num2str(loaddata_C(double(table2array(Loads(i,3))),2)/1000-
gendata_C(double(table2array(Loads(i,3))))/1000+bat_C(double(table2array(Loads(i,3))))/1000), ' kVar='
,num2str(loaddata_C(double(table2array(Loads(i,3))),3)/1000+Q_C(double(table2array(Loads(i,3))))/100
0));%Q_C(i)/1000));%

            DSSText.Command =strcat( ' AddBusMarker
Bus=',num2str(loaddata_C(double(table2array(Loads(i,3))),1)),' code=3 color=Blue size=40 dots=yes
labels=y subs=y ');

          else

            pause;

          end

        end

          DSSCircuit = DSSObj.ActiveCircuit ;

          DSSSolution = DSSCircuit.Solution;

        DSSLines = DSSCircuit.Lines;

        DSSText.Command='Set mode=snapshot Voltagebases=[11 0.415]'

        DSSText.Command='calcv'

          DSSSolution.Solve;

        I=zeros(length(l_code),6);

        i_Line = DSSLines.First;

        V_DSS_Lines=zeros(length(l_code),12);
```

```
while i_Line > 0

   i_Line;

   Currents_DSS_Lines(i_Line,:) = DSSCircuit.ActiveCktElement.Currents;

   I(i_Line,:)=Currents_DSS_Lines(i_Line,1:6);

   V_DSS_Lines(i_Line,:) = DSSCircuit.ActiveCktElement.Voltages;

   S(i_Line,:)=V_DSS_Lines(i_Line,1:6);

   i_Line = DSSLines.Next;

end

  DSSText.Command='Buscoords C:\Users\Indoopa\Documents\OpenDSS\Buscoords2.csv   ! load
in bus coordinates';

VMSi_A=DSSCircuit.AllNodeVmagByPhase(1);

VMSi_B=DSSCircuit.AllNodeVmagByPhase(2);

VMSi_C=DSSCircuit.AllNodeVmagByPhase(3);

 DSSText.Command = 'plot type=circuit quantity=Losses Max=200   dots=yes labels=yes subs=yes
C1=$00FF0000'

DSSText.Command ="Show Isolated";

Linelosses=DSSCircuit.ActiveCktElement.Losses

Linelosses=zeros(1,2);

for i=1 : length(linedata(:,1))

    %DSSText.Command =strcat('New Line.',num2str(i), ' Bus1=',num2str(linedata(i,2)), '.1.2.3.0
Bus2=',num2str(linedata(i,3) ),'.1.2.3.0  R1=',num2str(linedata(i,4)), ' X1=',num2str(linedata(i,5)), '
R0=',num2str(linedata(i,6)),' X0=',num2str(linedata(i,7)), ' C1=0 C0=0 Length=1' );

    DSSCircuit.SetActiveElement([strcat('Line.',num2str(i))]);

     Linelosses=Linelosses+DSSCircuit.ActiveCktElement.Losses;

     DSSCircuit.ActiveCktElement.Losses;

end

x=[1:n];
```

```
        Linelosses
    %displaying results
figure(1);
up=230*1.06*ones(n);
down=230*(1-0.06)*ones(n);
plot(x,VMSi_A,'^ r',x,VMSi_B,'* g',x,VMSi_C,'v b',x,up,'-',x,down,'-');
ylabel('Voltage(V)');
xlabel('Node');
legend({'U_A','U_B','U_C'});

 figure(2);
   x=[1:n];
   plot(x,Q_A/1000,'^ r',x,Q_B/1000,'* g',x,Q_C/1000,'v b');
   xlim([1 n]);
   ylabel('Reactive power(kVar)');
   xlabel('Node');
   legend({'L_A','L_B','L_C'});

   figure(3);
   x=1:n;
   plot(x,gendata_A/1000,'^ r',x,gendata_B/1000,'* g',x,gendata_C/1000,'v b');
   xlim([1 n]);
   ylabel('Power generation (kW)');
   xlabel('Node');
   legend({'Gen_A','Gen_B','Gen_C'});
```

```
figure(6);
x=[1:n];
plot(x,bat_A/1000,'^ r',x,bat_B/1000,'* g',x,bat_C/1000,'v b');
xlim([1 n]);
ylabel('Battery power(kW)');
xlabel('Node');
legend({'Bat_A','Bat_B','Bat_C'});
```

Droop control based algorithm

This methodology was implemented practically in the lab. Control algorithm was implemented in a Rasberry pi. Distribution grid was simulated using OpenDSS in a PC. Communication between PC and Rasberry pi was performed using wifi.

Python code for control algorithm (Implemented in Raspberry pi)

```
import numpy as np

import socket

import sys

import math



HOST = '169.254.231.6' #Server ip

PORT = 4000

client=('169.254.58.28',4000)

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

s.bind((HOST, PORT))

#s.listen(1)

#conn, addr = s.accept()

#print ('Connected by', addr)

##############################################################################################

controlsource=4

v=0.4

busnumber=10

QMSimax= 400*np.ones(busnumber, dtype=int)

print(QMSimax[2])

kMSi=np.ones(busnumber, dtype=int)

deltavmax=14

Vo=230
```

```
VMSi=230*np.ones(busnumber, dtype=int)
loaddata=np.matrix([ [1,0,0] ,
                     [2,1000,250],
                     [3,1000,250],
                     [4,1000,250],
                     [5,-5000,250],
                     [6,1000,250],
                     [7,1000,250],
                     [8,1000,250],
                     [9,1000,250],
                     [10,1000,250]])
print(loaddata[2,2])
Q_load=loaddata[:,2]
#print(Q_load)
encoding = 'utf-8'
int_list=[int(i) for i in loaddata[:,2] ]
int_list.insert(0,int(round(1000*v)))
#print (int_list)
string_list=[str(i) for i in int_list ]
s.sendto(bytes(' '.join(string_list),encoding),client)
while 1:
   data, addr = s.recvfrom(50000)
   print (data)
   print (data.decode(encoding))
   string_list=(data.decode(encoding)).split()
```

```
VMSi=[int(i) for i in string_list]
if loaddata[controlsource,2]>30 : #%smart transformer voltage changing
   v=v-0.01
elif loaddata[controlsource,2]<-30 :
   v=v+0.01
if v<0.374:
   v= 0.374
if v>0.422:
   v= 0.422
itr=0
while itr<1:

   if np.mean(VMSi)<244 and np.mean(VMSi)>216 and np.std(VMSi) <10:
      break
         for i in range(1,busnumber):
          kMSi[i]= QMSimax[i]/deltavmax
               if (VMSi[i] <220)  or ( VMSi[i] >240 ) : # current voltage is higher and more capacitive
injected
         loaddata[i,2]=Q_load[i]+kMSi[i]*(VMSi[i]-Vo) #droop control
         #loaddata[i,2]=loaddata[i,2]+kkMSi[i]*(VMSi[i]-Vo)

      if loaddata[i,2]<-1000:
         loaddata[i,2]=-1000

      if loaddata[i,2]>3000:
         loaddata[i,2]=2000
```

```
        itr=itr+1


    int_list=[int(i) for i in loaddata[:,2] ]
    int_list.insert(0,int(round(1000*v)))
    print (int_list)


    string_list=[str(i) for i in int_list ]
    #if not data: break
    print ( string_list)
    print(bytes(' '.join(string_list),encoding))
    s.sendto(bytes(' '.join(string_list),encoding),client)
    #break
s.close()
```

Python code for simulating distribution grid

```
from flask import Flask

import requests as req

import math

import win32com.client

import numpy as np

import time

import pandas as pd

import math

import matplotlib.pyplot as plt

from numpy.random import seed

from numpy.random import randint


data = pd.read_csv
(r'C:\Users\Indoopa\Downloads\European_LV_Test_Feeder_v2\European_LV_CSV\LineCodes.csv')

df = pd.DataFrame(data)

line_code = df.values.tolist()


data = pd.read_csv
(r'C:\Users\Indoopa\Downloads\European_LV_Test_Feeder_v2\European_LV_CSV\Lines.csv')

df = pd.DataFrame(data)

lines = df.values.tolist()


data = pd.read_csv
(r'C:\Users\Indoopa\Downloads\European_LV_Test_Feeder_v2\European_LV_CSV\Loads.csv')

df = pd.DataFrame(data)
```

```
loads = df.values.tolist()

data = pd.read_csv
(r'C:\Users\Indoopa\Downloads\European_LV_Test_Feeder_v2\European_LV_CSV\LoadShapes.csv')

df = pd.DataFrame(data)

loadShapes = df.values.tolist()

def grid(line_code,lines,loads,loadShapes,v,q_smart,PV_gen):

    DSSObj = win32com.client.Dispatch('OpenDSSEngine.DSS')

  Start=DSSObj.Start(0)

  DSSText = DSSObj.Text

  DSSText.Command = "clear"

  DSSText.Command = "set defaultbasefreq=50 "

  DSSText.Command ="New Circuit.Simple"

  DSSText.Command = "Set Voltagebases = 0.4"

  DSSText.Command ="Edit Vsource.Source BasekV="+str(v/1000)+" pu=1.00 ISC3=300000
ISC1=250000"

  for i in range(1,len(line_code)):

    DSSText.Command ='New Linecode.'+ line_code[i][0]+' R1='+line_code[i][2]+' X1='+line_code[i][3] +'
R0='+line_code[i][4] +' X0='+line_code[i][5]+' C1='+line_code[i][6] +' C0='+line_code[i][7]+'
Units='+line_code[i][8]


  DSSText.Command ='New Line.'+str(1)+' Bus1=SourceBus.1.2.3.0  Bus2='+str(lines[1][2])+'.1.2.3.0
Linecode='+ str(lines[1][6]) +' Length='+ str(float(lines[1][4])/1000)


  for i in range(2,len(lines)):

    DSSText.Command ='New Line.'+str(i)+' Bus1='+str(lines[i][1])+ '.1.2.3.0
Bus2='+str(lines[i][2])+'.1.2.3.0 Linecode='+ str(lines[i][6]) +' Length='+ str(float(lines[i][4])/1000)

  for i in range(2,len(loads)):

    var=int(loads[i][7])*math.tan(math.acos(float(loads[i][8])))
```

```
        var=var+q_smart[i-2]/1000

        data = pd.read_csv
(r'C:\Users\Indoopa\Downloads\European_LV_Test_Feeder_v2\European_LV_CSV\Load
Profiles\Load_profile_'+str(i-1)+'.csv')

        df = pd.DataFrame(data)

        loadprofile = df.values.tolist()

        mult=float(loadprofile[720][1])

        if loads[i][3]=='A':

            DSSText.Command ='New Load.'+ str(i-1)+ ' Bus1='+loads[i][2]+'.1.0 Model=1 kV=0.23 phase=1'+ '
kW='+str(int(loads[i][7])*mult-int(PV_gen[i-2]/1000))+ ' kVar='+str(var)

        elif loads[i][3]=='B':

            DSSText.Command ='New Load.'+ str(i-1)+ ' Bus1='+loads[i][2]+'.2.0 Model=1 kV=0.23 phase=1'+ '
kW='+str(int(loads[i][7])*mult-int(PV_gen[i-2]/1000))+ ' kVar='+str(var)

        elif loads[i][3]=='C':

            DSSText.Command ='New Load.'+ str(i-1)+ ' Bus1='+loads[i][2]+'.3.0 Model=1 kV=0.23 phase=1'+ '
kW='+str(int(loads[i][7])*mult-int(PV_gen[i-2]/1000))+ ' kVar='+str(var)

        else:

            print('error')

    DSSCircuit = DSSObj.ActiveCircuit

    DSSSolution = DSSCircuit.Solution

    DSSSolution.Solve()

    #DSSText.Command ='Show eventlog'

    #DSSText.Command ='new monitor.m1 PVSystem.PV mode=1 ppolar=no'

    #DSSText.Command ='show mon m1'


    DSSLines = DSSCircuit.Lines

    VMSi_A=DSSCircuit.AllNodeVmagPUByPhase(1)
```

```
    VMSi_B=DSSCircuit.AllNodeVmagPUByPhase(2)
    VMSi_C=DSSCircuit.AllNodeVmagPUByPhase(3)
    DSSObj.ClearAll()
    return [VMSi_A,VMSi_B,VMSi_C]
def pv(loads,i):
    seed(i)
    PV_gen = randint(0,7000, len(loads)-2)
    return PV_gen
def smart_inverter_response(loads,nodevoltage):
    v=230
    deltav=30
    load_data = np.zeros(len(loads)-2)
    for i in range(0,len(loads)-2):
        if (nodevoltage[i]<207):
            load_data[i]=0.25*7000
        elif(nodevoltage[i]>207 and nodevoltage[i]<253):
            load_data[i]= (0.25*7000/23)*(v-nodevoltage[i])
        else:
            load_data[i]=-0.25*7000
    time.sleep(0.1)
    return load_data
def loadvoltage(loads,vset):
    nodevoltage=np.zeros(len(loads)-2)
    for i in range(2,len(loads)):
        if loads[i][3]=='A':
```

```
            nodevoltage[i-2]=vset[0][int(loads[i][2])-1]
            #print(nodevoltage[i-2])
        elif loads[i][3]=='B':
            nodevoltage[i-2]=vset[1][int(loads[i][2])-1]
            #print(nodevoltage[i-2])
        elif loads[i][3]=='C':
            nodevoltage[i-2]=vset[2][int(loads[i][2])-1]
            #print(nodevoltage[i-2])
        else:
            print('error')
    return nodevoltage
def smarttransformer(v,Q):
    if Q>100 :
        v=v+1
    elif Q<-100 :
        v=v-1
    if v<374:
        v= 374
    if v>422:
        v= 422
    return v
#################################################################################################
i=0
f, axs = plt.subplots(2,4)
PV_gen=np.zeros(len(loads)-2)
```

```
v=400
while(True):
    i=i+1
    PV_gen=pv(loads,i)
    v=400
    q_smart=np.zeros(len(loads)-2)
    m=0
    while(m<15):
        m=m+1
        vset=grid(line_code,lines,loads,loadShapes,v,q_smart,PV_gen)
        nodevoltage=loadvoltage(loads,vset)
        nodevoltage_tr=nodevoltage
        q_smart_tr=q_smart
        v_tr=vset
        j=0
        while(j<5):
            plt.clf()
            j=j+1
            q_smart=smart_inverter_response(loads,nodevoltage)
            vset=grid(line_code,lines,loads,loadShapes,v,q_smart,PV_gen)
            axs[0,0].plot(vset[0], 'r^',vset[1], 'g^',vset[2], 'b^')
            axs[0,0].set_title('Phase voltage (V) '+str(j))
            plt.subplot(axs[0,0])
            axs[0,1].plot(PV_gen, 'r-^')
            axs[0,1].set_title('PV generation (W)')
```

```
    plt.subplot(axs[0,1])
    axs[0,2].plot(q_smart, 'g-^')
    axs[0,2].set_title('Q generation (Var) '+str(round(sum(abs(q_smart)))))
    plt.subplot(axs[0,2])
    nodevoltage=loadvoltage(loads,vset)
    axs[0,3].plot(nodevoltage, 'g-^')
    axs[0,3].set_title('Load voltage (V)')
    plt.subplot(axs[0,3])
    ############################################################
    axs[1,0].plot(v_tr[0], 'r^',v_tr[1], 'g^',v_tr[2], 'b^')
    axs[1,0].set_title('Phase voltage (V) '+str(m) +' '+str(round(v_tr[0][0])))
    plt.subplot(axs[1,0])
    axs[1,1].plot(PV_gen, 'r-^')
    axs[1,1].set_title('PV generation (W)')
    plt.subplot(axs[1,1])
    axs[1,2].plot(q_smart_tr, 'g-^')
    axs[1,2].set_title('Q generation (Var) '+str(round(sum(abs(q_smart_tr)))))
    plt.subplot(axs[1,2])
    nodevoltage=loadvoltage(loads,vset)
    axs[1,3].plot(nodevoltage_tr, 'g-^')
    axs[1,3].set_title('Load voltage (V)')
    plt.subplot(axs[1,3])
    plt.pause(0.0001)
print(str(round(v)))
v=smarttransformer(v,q_smart[50])
```

```
        file1 = open(r"C:\Users\Indoopa\Desktop\V.txt","w")
        file1.write(str(round(v)))
        ''' print("http://10.30.13.42:5000/"+str(round(v))+" "+str(round(q_smart[50])))
        smart_transformer_resp = req.get("http://10.30.13.42:5000/"+str(round(v))+"
"+str(int(round(q_smart[50]))))
        print('voltage of smart transformer '+smart_transformer_resp .text)
        v=int(smart_transformer_resp .text)   '''
       # plt.pause(0.0001)
     plt.figure(1)
    plt.plot(vset[0], 'r^',vset[1], 'g^',vset[2], 'b^')
    plt.ylabel('Phase voltage (V)')
    plt.draw()
    plt.pause(0.0001)
    plt.clf()
    plt.figure(2)
    plt.plot(PV_gen, 'r^')
    plt.ylabel('PV generation (W)')
    plt.draw()
    plt.pause(0.0001)
    #plt.clf()

    plt.figure(3)
    plt.plot(q_smart, 'r^')
    plt.ylabel('Q generation (Var)')
    plt.draw()
    plt.pause(0.0001)
```

```
    plt.clf()

DSSText.Command ='New XYCurve.Eff npts=4 xarray=[.1 .2 .4 1.0] yarray=[1 1 1 1]'

DSSText.Command ='New XYCurve.FatorPvsT npts=4 xarray=[0 25 75 100] yarray=[1.2 1.0 .8 .6] '

DSSText.Command ='New PVSystem.PV phases=3 bus1=200 Pmpp=100 kV=0.23 kVA=100 conn=wye
%Cutin=10 %Cutout=10 EffCurve=Eff P-TCurve=FatorPvsT Temperature=25 irradiance=0.1 debug=yes'

DSSText.Command ='New XYCurve.vv_curve npts=4 Yarray=(1.0,1.0,-1.0,-1.0) XArray=(0.5,0.95,1.05,1.5)'

DSSText.Command ='New InvControl.InvPVCtrl mode=VOLTVAR voltage_curvex_ref=rated
vvc_curve1=vv_curve EventLog=yes'
```

Appendix A

The following equations were obtained by expanding equations (1) and (2). Variables in non-convex terms ($U_{A}^{R}{}_{n}, I_{A}^{R}{}_{mn}, U_{A}^{Im}{}_{n}, I_{A}^{Im}{}_{mn}$) were replaced from respective values obtained from the load flow. Then, $W_{A_m} = |\boldsymbol{U_{A_m}}|^2$ and $L_{A_{mn}} = |\boldsymbol{I_{A_{mn}}}|^2$ substitutions were made to make the equations linear.

$$p_{A_{mn}} = \sum_{k:(j,k)\in E} p_{A_{nk}} + \mathrm{P}_{\mathrm{A_n}}^{\mathrm{C}} - \mathrm{P}_{\mathrm{A_n}}^{\mathrm{G}} + \mathrm{R}_{\mathrm{s_{mn}}} L_{A_{mn}} + \mathrm{R}_{\mathrm{m_{mn}}}\left(I_{A}^{R}{}_{mn}\, I_{B}^{R}{}_{mn} + I_{A}^{Im}{}_{mn}\, I_{B}^{Im}{}_{mn}\right) - \mathrm{X}_{\mathrm{m_{mn}}}\left(I_{A}^{R}{}_{mn}\, I_{B}^{Im}{}_{mn} - I_{A}^{Im}{}_{m}\, I_{B}^{R}{}_{m}\right) + \mathrm{R}_{\mathrm{m_{mn}}}\left(I_{A}^{R}{}_{mn}\, I_{C}^{R}{}_{mn} + I_{A}^{Im}{}_{mn}\, I_{C}^{Im}{}_{mn}\right) - \mathrm{X}_{\mathrm{m_{mn}}}\left(I_{A}^{R}{}_{mn}\, I_{C}^{Im}{}_{mn} - I_{A}^{Im}{}_{m}\, I_{C}^{R}{}_{m}\right) \quad \text{(A1)}$$

$$q_{A_{mn}} = \sum_{k:(j,k)\in E} q_{A_{nk}} + \mathrm{Q}_{\mathrm{A_n}}^{\mathrm{C}} + Q_{A_n}^{G} + \mathrm{X}_{\mathrm{s_{mn}}} L_{A_{mn}} + \mathrm{X}_{\mathrm{m_{mn}}}\left(I_{A}^{R}{}_{mn}\, I_{B}^{R}{}_{mn} + I_{A}^{Im}{}_{mn}\, I_{B}^{Im}{}_{mn}\right) + \mathrm{R}_{\mathrm{m_{mn}}}\left(I_{A}^{R}{}_{mn}\, I_{B}^{Im}{}_{mn} - I_{A}^{Im}{}_{m}\, I_{B}^{R}{}_{m}\right) + \mathrm{X}_{\mathrm{m_{mn}}}\left(I_{A}^{R}{}_{mn}\, I_{C}^{R}{}_{mn} + I_{A}^{Im}{}_{mn}\, I_{C}^{Im}{}_{mn}\right) + \mathrm{R}_{\mathrm{m_{mn}}}\left(I_{A}^{R}{}_{mn}\, I_{C}^{Im}{}_{mn} - I_{A}^{Im}{}_{m}\, I_{C}^{R}{}_{m}\right) \quad \text{(A2)}$$

$$W_{A_m} = W_{A_n} + \mathrm{R}_{\mathrm{S_{mn}}}{}^{2} L_{A_{mn}} + \mathrm{X}_{\mathrm{S_{mn}}}{}^{2} L_{A_{mn}} + \mathrm{R}_{\mathrm{m_{mn}}}{}^{2} L_{B_{mn}} + \mathrm{X}_{\mathrm{m_{mn}}}{}^{2} L_{B_{mn}} + \mathrm{R}_{\mathrm{m_{mn}}}{}^{2} L_{C_{mn}} + \mathrm{X}_{\mathrm{m_{mn}}}{}^{2} L_{C_{mn}} - 2\, p_{mn_A} - 2\, q_{mn_A} - 2V_{A}^{R}{}_{n}\mathrm{R}_{\mathrm{m_{mn}}} I_{B}^{R}{}_{mn} + 2V_{A}^{R}{}_{n}\mathrm{X}_{\mathrm{m_{mn}}} I_{B}^{Im}{}_{mn} - 2V_{A}^{R}{}_{n}\mathrm{R}_{\mathrm{m_{mn}}} I_{C}^{R}{}_{mn} + 2V_{A}^{R}{}_{n}\mathrm{X}_{\mathrm{m_{mn}}} I_{C}^{Im}{}_{mn} - 2V_{A}^{Im}{}_{n}\mathrm{X}_{\mathrm{m_{mn}}} I_{B}^{R}{}_{mn} - 2V_{A}^{Im}{}_{n}\mathrm{R}_{\mathrm{m_{mn}}} I_{B}^{Im}{}_{mn} - 2V_{A}^{Im}{}_{n}\mathrm{X}_{\mathrm{m_{mn}}} I_{C}^{R}{}_{mn} - 2V_{A}^{Im}{}_{n}\mathrm{R}_{\mathrm{m_{mn}}} I_{C}^{Im}{}_{mn} + 2\mathrm{R}_{\mathrm{S_{mn}}} I_{A}^{R}{}_{mn}\mathrm{R}_{\mathrm{m_{mn}}} I_{B}^{R}{}_{mn} - 2\mathrm{R}_{\mathrm{S_{mn}}} I_{A}^{R}{}_{mn}\mathrm{X}_{\mathrm{m_{mn}}} I_{B}^{Im}{}_{mn} + 2\mathrm{R}_{\mathrm{S_{mn}}} I_{A}^{R}{}_{mn}\mathrm{R}_{\mathrm{m_{mn}}} I_{C}^{R}{}_{mn} - 2\mathrm{R}_{\mathrm{S_{mn}}} I_{A}^{R}{}_{mn}\mathrm{X}_{\mathrm{m_{mn}}} I_{C}^{Im}{}_{mn} - 2\mathrm{X}_{\mathrm{S_{mn}}} I_{A}^{Im}{}_{mn}\mathrm{R}_{\mathrm{m_{mn}}} I_{B}^{R}{}_{mn} + 2\mathrm{X}_{\mathrm{S_{mn}}} I_{A}^{Im}{}_{mn}\mathrm{X}_{\mathrm{m_{mn}}} I_{B}^{Im}{}_{mn} - 2\mathrm{X}_{\mathrm{S_{mn}}} I_{A}^{Im}{}_{mn}\mathrm{R}_{\mathrm{m_{mn}}} I_{C}^{R}{}_{mn} - 2\mathrm{X}_{\mathrm{S_{mn}}} I_{A}^{Im}{}_{mn}\mathrm{X}_{\mathrm{m_{mn}}} I_{C}^{Im}{}_{mn} + 2\mathrm{R}_{\mathrm{m_{mn}}} I_{B}^{R}{}_{mn}\mathrm{R}_{\mathrm{m_{mn}}} I_{C}^{R}{}_{mn} - 2\mathrm{R}_{\mathrm{m_{mn}}} I_{B}^{R}{}_{mn}\mathrm{X}_{\mathrm{m_{mn}}} I_{C}^{Im}{}_{mn} + 2\mathrm{X}_{\mathrm{m_{mn}}} I_{B}^{Im}{}_{mn}\mathrm{X}_{\mathrm{m_{mn}}} I_{C}^{Im}{}_{mn} + 2\mathrm{X}_{\mathrm{S_{mn}}} I_{A}^{R}{}_{mn}\mathrm{X}_{\mathrm{m_{mn}}} I_{B}^{R}{}_{mn} + 2\mathrm{X}_{\mathrm{S_{mn}}} I_{A}^{R}{}_{mn}\mathrm{X}_{\mathrm{m_{mn}}} I_{C}^{R}{}_{mn} + 2\mathrm{X}_{\mathrm{S_{mn}}} I_{A}^{R}{}_{mn}\mathrm{R}_{\mathrm{m_{mn}}} I_{C}^{Im}{}_{mn} + 2\mathrm{R}_{\mathrm{S_{mn}}} I_{A}^{Im}{}_{mn}\mathrm{X}_{\mathrm{m_{mn}}} I_{B}^{R}{}_{mn} + 2\mathrm{R}_{\mathrm{S_{mn}}} I_{A}^{Im}{}_{mn}\mathrm{R}_{\mathrm{m_{mn}}} I_{B}^{Im}{}_{mn} + 2I_{A}^{Im}{}_{mn}\mathrm{X}_{\mathrm{m_{mn}}} I_{C}^{R}{}_{mn} + 2\mathrm{R}_{\mathrm{S_{mn}}} I_{A}^{Im}{}_{mn}\mathrm{R}_{\mathrm{m_{mn}}} I_{C}^{Im}{}_{mn} + 2\mathrm{X}_{\mathrm{m_{mn}}} I_{B}^{R}{}_{mn}\mathrm{X}_{\mathrm{m_{mn}}} I_{C}^{R}{}_{mn} + 2\mathrm{X}_{\mathrm{m_{mn}}} I_{B}^{R}{}_{mn}\mathrm{R}_{\mathrm{m_{mn}}} I_{C}^{Im}{}_{mn} + 2\mathrm{R}_{\mathrm{m_{mn}}} I_{B}^{Im}{}_{mn}\mathrm{R}_{\mathrm{m_{mn}}} I_{C}^{Im}{}_{mn} \quad \text{(A3)}$$