METU EE

EE472
Power System Analysis II

Project 1
Task 2
Spring-2022

Can Aydın
2304053

1. Introduction

Load Flow Analysis is a significantly important method to understand the operation of power system analysis. In order to compute this analysis, engineers need topology information. Then, they can construct bus admittance matrix to be able to continue analyzing process. Bus admittance matrix reveals the necessary information to solve nonlinear differential equations to observe the operation of the system. There are different methods to solve nonlinear differential equations. The most common ways in power systems are Newton-Raphson and Gauss-Seidel method. In this project, we will implement fast decoupled load flow method in MATLAB. In task 1, the data is read from ieee file and bus admittance matrix is constructed. In this second task, a method of load flow analysis, Fast Decoupled Load Flow, is implemented in MATLAB.

2. Results

In Fast Decoupled Load Flow analysis, B' and B'' matrices are implemented directly from bus admittance matrix. B matrix is equal to imag(Y_bus). B' matrix includes the busses apart from slack bus. Similarly, B'' matrix includes only PQ busses. Then, the following formula is used to calculate ΔP and ΔQ.

$$\text{Real power} = P_i = R_e \left\{ \mathbf{V}_i^* \sum_{k=1}^{n} \mathbf{Y}_{ik} \mathbf{V}_k \right\} \quad \ldots(6.58a)$$

$$\text{Reactive power} = Q_i = -I_m \left\{ \mathbf{V}_i^* \sum_{k=1}^{n} \mathbf{Y}_{ik} \mathbf{V}_k \right\} \quad \ldots(6.58b)$$

Figure.1: Calculation of P_i and Q_i.

Then, the following formula is used to calculate Δθ and ΔV.

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} J_1 & 0 \\ 0 & J_2 \end{bmatrix} \begin{bmatrix} \Delta \delta \\ \Delta |V| \end{bmatrix}$$

Figure.2: Calculation of Δθ and ΔV.

In this project, flat start is used and convergence criteria is chosen as 0.001 pu. However, for 57 and 118 bus system, it is chosen as 0.01 pu, since, it is not possible to reach a solution with 0.001 pu.

The advantage of FDLF method is, Jacobian matrix is not calculated for each iteration. Coupling between P-θ and Q-V helps us to make assumptions to eliminate elements of Jacobian matrix. Hence, we can choose a constant B' and B'' matrices directly from B matrix. It is easy to implement and computation effort is reduced. Convergence is reached in 16 iterations for 14 and 30 bus systems. For 57 bus system, 19 iteration is needed.

2.1 Simulation Results

	Voltage	Angle	Pg	Qg
1	1.06000000000000	0	2.32393473125510	-0.165495683429565
2	1.04500000000000	-4.9825941873	0.399998032268455	0.435560189762617
3	1.01000000000000	-12.725109249	-2.83429328018503e-06	0.250745653743306
4	1.01767236152071	-10.312920171	0	0
5	1.01951435182569	-8.7738621286	0	0
6	1.07000000000000	-14.220899913	-1.14694394613979e-05	0.127354775458709
7	1.06152336110455	-13.359685155	0	0
8	1.09000000000000	-13.359685155	0	0.176210822571912
9	1.05593904084886	-14.938598364	0	0
10	1.05099093796877	-15.097344904	0	0
11	1.05691009749680	-14.790625435	0	0
12	1.05502299425925	-15.073500921	0	0
13	1.05045922151299	-15.157081143	0	0
14	1.03556528027198	-16.034015674	0	0

Table.1: Simulation Results for 14 bus system

	Voltage	Angle	Pg	Qg
1	1.06000000000000	0	2.60952219975626	-0.165263207106504
2	1.04300000000000	-5.35000247108351	0.399999671014077	0.495647519182947
3	1.02070978561725	-7.53230171468447	0	0
4	1.01172547318693	-9.28452868655398	0	0
5	1.01000000000000	-14.1671238387933	1.41134300835066e-06	0.369356906118834
6	1.01023133930881	-11.0654088032232	0	0
7	1.00236165915965	-12.8661543684273	0	0
8	1.01000000000000	-11.8145683051828	7.32809658227085e-06	0.372166458757528
9	1.05089957953640	-14.1098521015222	0	0
10	1.04511509131159	-15.7006274781250	0	0
11	1.08200000000000	-14.1098521015222	0	0.161781994911623
12	1.05710008742167	-14.9441287918834	0	0
13	1.07100000000000	-14.9441287918834	0	0.106334331224190
14	1.04213020740444	-15.8356782225282	0	0
15	1.03771901503619	-15.9284773732667	0	0
16	1.04437502378248	-15.5272380493795	0	0
17	1.03988907313670	-15.8623305681733	0	0
18	1.02817075631237	-16.5427664783735	0	0
19	1.02565908077155	-16.7164865430245	0	0
20	1.02973941085308	-16.5199065033196	0	0
21	1.03271545086231	-16.1433651706891	0	0
22	1.03324721783157	-16.1291319434155	0	0
23	1.02720593957818	-16.3191133422918	0	0
24	1.02159177912566	-16.4956414569345	0	0
25	1.01733114961983	-16.0677634693270	0	0
26	0.999678469843737	-16.4869473662275	0	0
27	1.02323413588246	-15.5434104454882	0	0
28	1.00679801649092	-11.6893083232378	0	0
29	1.00339502093239	-16.7733780786094	0	0
30	0.991919937656071	-17.6562513545761	0	0

Table.2: Simulation Results for 30 bus system

	Voltage	Angle	Pg	Qg
1	1.04000000000000	0	4.23675607081550	1.11862912053703
2	1.01000000000000	-1.18820947574635	1.37854312917685e-06	-0.00754852161751607
3	0.985000000000000	-5.98833018351342	0.400066304874717	-0.00880297086613610
4	0.980778642339093	-7.33761078026351	0	0
5	0.976498345782576	-8.54672025801133	0	0
6	0.980000000000000	-8.67446057816491	3.12420021353343e-05	0.00887663165820694
7	0.984185486741359	-7.60178550273643	0	0
8	1.00500000000000	-4.47829619149919	4.50002585255255	0.621218461662385
9	0.980000000000000	-9.58505133391047	6.66678698844070e-05	0.0232253954107355
10	0.986236442342832	-11.4499844842769	0	0

11	0.973949949062285	-10.1936419204973	0	0
12	1.01500000000000	-10.4714753492923	3.10008161409706	1.28656952942492
13	0.978874095747303	-9.80384349228531	0	0
14	0.970158033500423	-9.35072387835571	0	0
15	0.988018953925801	-7.19040037757925	0	0
16	1.01336829524550	-8.85913522108285	0	0
17	1.01745388079010	-5.39600334832547	0	0
18	1.00065095832133	-11.7298930101844	0	0
19	0.970111321279350	-13.2262755353195	0	0
20	0.963727171544790	-13.4439984163474	0	0
21	1.00841142172155	-12.9286417999444	0	0
22	1.00965283580325	-12.8739207190180	0	0
23	1.00823558096045	-12.9391627814902	0	0
24	0.999086063749132	-13.2917308782508	0	0
25	0.982227993495139	-18.1742918812854	0	0
26	0.958683709109713	-12.9809593389927	0	0
27	0.981455419010375	-11.5129785560772	0	0
28	0.996608187794790	-10.4808004014873	0	0
29	1.01018159872742	-9.77223155036621	0	0
30	0.962292850761931	-18.7209448125116	0	0
31	0.935381875817491	-19.3854567051880	0	0
32	0.949009694223837	-18.5126135954178	0	0
33	0.943054748936516	-18.2816374124256	0	0
34	0.958813280684590	-14.1456118539873	0	0
35	0.965882444135208	-13.9021607081580	0	0
36	0.975584622052337	-13.6324423199818	0	0
37	0.984716283298055	-13.4455498050990	0	0
38	1.01272653722021	-12.7342510514155	0	0
39	0.982652523079624	-13.4905422120005	0	0
40	0.972571061143344	-13.6559111092592	0	0
41	0.996167869718094	-14.0772935865405	0	0
42	0.966415991878069	-15.5320483278435	0	0
43	1.00954105966131	-11.3548280123999	0	0
44	1.01672854105339	-11.8562453951210	0	0
45	1.03597101510744	-9.27027694548154	0	0
46	1.05976287067234	-11.1166623557375	0	0
47	1.03320210975043	-12.5125038629530	0	0
48	1.02728560262392	-12.6097995417946	0	0
49	1.03620143141257	-12.9362973806475	0	0
50	1.02330426677558	-13.4127794552418	0	0
51	1.05225164556808	-12.5337019755274	0	0
52	0.979998486568196	-11.4839448809470	0	0
53	0.970578787892442	-12.2370103125991	0	0
54	0.996132035563867	-11.6995027971375	0	0
55	1.03077441744547	-10.8016022287896	0	0
56	0.968201469989628	-16.0628341398058	0	0
57	0.964679301986151	-16.5827993699817	0	0

Table.3: Simulation Results for 57 bus system

2.2 Power Losses

14 Bus	13.67425 MW
30 Bus	17.55131 MW
57 Bus	-27.09709 MW

Table.4: Power losses for 14,30,57 bus systems.

For 14 and 30 bus systems, convergence criteria is 0.001 pu. For 57 bus system, the results is reached by changing convergence criteria to 0.01 pu. It can be seen that there is something wrong as the power loss is negative.

2.3 Convergence Performance

As previously mentioned, the convergence criteria is chosen as 0.001 pu for this project. However, since it is not possible to reach a solution for 57 and 118 bus systems it is changed to 0.01 pu. 14 and 30 bus systems reached convergence after 16 iterations. We can say that bus and iteration number is not directly proportional. On the other hand, the higher number of busses in a system, the longer one iteration lasts. Overall time spent increases with increasing number of busses for this project (at least with my code).

As can be seen from the simulation results, the results are similar with the data in the IEEE data file. Therefore, we can conclude that the code works properly. Moreover, when we change the criteria to 0.1 pu, 57 bus system reaches convergence in 4 iterations. However, for 118 bus system, it is not possible to reach convergence. Using Newton-Raphson method might solve the problem.

To compare the total time spent, convergence criteria is chosen as 0.01 to have more precise results.

	Total Iteration	**Time spent(s)**
14 Bus	5	0.010965
30 Bus	5	0.021819
57 Bus	19	0.027920

Table.5: Time spent for different systems with same convergence criteria.

	Time spent(s)
Convergence Criteria = 0.001 pu	0.012056
Convergence Criteria = 0.01 pu	0.010965

Table.6: Time spent for different convergence criteria in 14 bus system.

	Time spent(s)
Given Load	0.0308
Load -40%	0.0294
Load +40%	0.0395

Table.7: Time spent for different loads in 30 bus system.

Overall, iteration number is not related with bus number. However, total time spent is directly proportional with bus number. In addition, the smaller convergence criteria, the longer simulation lasts. Finally, the amount of load can change the total time needed for simulation. As load increases, complexity increases. Hence, total time spent is higher. When the system is larger, it is more complex. Hence, simulation time is larger to obtain load flow analysis.

3. Code

```
function [V_bus,Angle_bus,Pg, Qg] = e230405_Aydin(cdf_path)
    dataFile = cdf_path;
    format long;
    epsilon = 0.001;
    [busStart,busEnd,branchStart,branchEnd] = getLineNum(dataFile);
    busData = writeBusData(dataFile,busStart,busEnd);
    sBase = readTitle(dataFile);
    sBase = str2num(sBase);
    busNum = busEnd-busStart+1;
    branchData = writeBranchData(dataFile,branchStart,branchEnd);
    Y_bus = writeOnBus(busData,branchData,busNum);
    slackBus = findSlack(busData);

    slackBusNumber = findYBusEntries(slackBus,busNum,busData);
    pvBuses = findPVBus(busData);
    pqBuses = findPQBus(busData);

    Y_bus = writeOnBus(busData,branchData,busNum);
    bMatrix = -imag(Y_bus);
    bPrime = createBprime(Y_bus,slackBusNumber);
    bDoublePrime =
createBDoublePrime(Y_bus,pqBuses,pvBuses,slackBus,bPrime,busNum,busData);
    busVoltage = findBusVoltages(busData,slackBusNumber,busNum);
    for t=1:length(pqBuses)
        busVoltage(pqBuses(t)) = 1;
    end
    [pSpec,qSpec] = findSpecValues(busData,slackBusNumber,sBase);
    initA = zeros(busNum,1);
%     tic;
    for i = 1:20
        [deltaP,P,deltaQ,Q] =
findDeltaPQ(Y_bus,pSpec,qSpec,busVoltage,initA);
        firstCorMat = deltaP./busVoltage;
        firstCorMat(slackBusNumber,:) = [];
        deltaA = inv(bPrime)*(firstCorMat);
        secondCorMat = deltaQ./busVoltage;
        thirdCorMat = zeros(length(pqBuses),1);
        for k=1:length(pqBuses)
            thirdCorMat(k) = secondCorMat(pqBuses(k),:);
        end
        deltaV = inv(bDoublePrime)*(thirdCorMat);


        if abs(deltaP(2:end))<=epsilon
            fprintf("deltaP Converges\n")
            if sum(abs(deltaQ(2:end))<=epsilon)
                fprintf("deltaQ also converges\n")
                V_bus = busVoltage;
                Angle_bus = initA * (360/(2*pi));
```

```
                Pg = P;
                Qg = Q;

                for t=1:length(pqBuses)
                    Pg(pqBuses(t)) = 0;
                    Qg(pqBuses(t)) = 0;
                end
             break
             end
        end
        fprintf("iteration:%d\n",i)
        for p=1:(busNum-1)
            initA(p+1) = initA(p+1)+deltaA(p);
        end

        for t=1:length(pqBuses)
            busVoltage(pqBuses(t)) = busVoltage(pqBuses(t)) + deltaV(t);
        end

    end
%     toc;
    Pgen = Pg(1);
    Qgen = Qg(1);
    Pg = zeros(length(deltaP),1);
    Qg = zeros(length(deltaQ),1);
    Pg(1) = P(1);
    Qg(1) = Q(1);
    for i = 1:length(pvBuses)
        if busData(pvBuses(i),4)>0
            Pgen = Pgen+ P(pvBuses(i)) + busData(pvBuses(i),4)/sBase;
            Pg(pvBuses(i)) = P(pvBuses(i)) +busData(pvBuses(i),4)/sBase;
        else
            Pgen = Pgen+ P(pvBuses(i));
            Pg(pvBuses(i)) = P(pvBuses(i));
        end

        if busData(pvBuses(i),5)>0
          Qgen = Qgen+ Q(pvBuses(i)) +busData(pvBuses(i),5)/sBase;
          Qg(pvBuses(i)) = Q(pvBuses(i)) +busData(pvBuses(i),5)/sBase;
        else
          Qgen = Qgen+ Q(pvBuses(i));
          Qg(pvBuses(i)) = Q(pvBuses(i));
        end
    end
  Pload = sum(busData(1:end,4))/sBase;
  Qload = sum(busData(1:end,5))/sBase;
  PowerLoss = Pgen-Pload;
  fprintf("Total Loss is %d MW",PowerLoss*sBase)


end

function [deltaP,P,deltaQ,Q] = findDeltaPQ(Y_bus, initP,initQ,initV,initA)
    P = zeros(length(Y_bus),1);
    Q = zeros(length(Y_bus),1);
    G = real(Y_bus);
    B = imag(Y_bus);
    for i = 1:length(Y_bus)
        for k = 1:length(Y_bus)
```

```
            P(i) = P(i) + initV(k)*initV(i)*((G(i,k)*cos(initA(i)-
initA(k)))+(B(i,k)*sin(initA(i)-initA(k))));
        end
    end
    deltaP = initP-P;

    for i = 1:length(Y_bus)
        for k = 1:length(Y_bus)
            Q(i) = Q(i) + initV(k)*initV(i)*((G(i,k)*sin(initA(i)-
initA(k)))-(B(i,k)*cos(initA(i)-initA(k))));
        end
    end

    deltaQ = initQ-Q;

end

function [pSpec,qSpec]= findSpecValues(busData,slackBusNumber,sBase)

    pSpec = (busData(1:end,6)-busData(1:end,4))/sBase;
    qSpec = (busData(1:end,7)-busData(1:end,5))/sBase;
%     pSpec(slackBusNumber,;) = [];
%     qSpec(slac
end
function [V] = findBusVoltages(busData,slackBusNumber,busNum)

    V = ones(busNum,1);
    for i=1:length(busData)
        V(i,1) = busData(i,3);
    end

%      V(slackBusNumber,:) = [];

end
function pvBus = findPVBus(busData)
    pvBus = [];
    for i = 1:length(busData)
        if busData(i,2) == 2
            pvBus = [pvBus,busData(i,1)];
        end
    end
end

function pqBus = findPQBus(busData)
    pqBus = [];
    for i = 1:length(busData)
        if busData(i,2) == 0
            pqBus = [pqBus,busData(i,1)];
        end
    end
end

function [sBase] = readTitle(dataFile)
    rawData = readlines(dataFile);
    splitRawData = split(rawData(1));
    sBase = splitRawData(5);
end

function slackBusNum = findSlack(busData)
```

```
    for i = 1:length(busData)
        if busData(i,2) == 3
            slackBusNum = busData(i,1);
            break;
        end
    end

end
function [busStart,busEnd,branchStart,branchEnd] = getLineNum(dataFile)
    i = 1;
    busStart = 0;
    branchStart = 0;
    busEnd = 0;
    branchEnd = 0;
    rawData = readlines(dataFile);
    while i
        x = split(rawData(i));
        if x(1) == "BUS" && x(2) == "DATA"
            busStart = i+1;
        elseif x(1) == "-999" && branchStart == 0
            busEnd = i-1;
        elseif x(1) == "BRANCH" && x(2) == "DATA"
            branchStart = i+1;
        elseif x(1) == "-999" && branchStart ~= 0
            branchEnd = i-1;
            break
        end
        i = i+1;
    end
end

function [busData] = writeBusData(dataFile,busStart,busEnd)
    rawData = readlines(dataFile);
    busData = {};
    for i = busStart:busEnd
        rowData = split(rawData(i));
        if busEnd-busStart+1 == 300
            if rowData(1) == ""
                busData = [busData;
rowData(2),rowData(6),rowData(7),rowData(9),rowData(10),rowData(11),rowData
(12),rowData(17),rowData(18)];
                busData = double(busData);
            else
                busData = [busData;
rowData(1),rowData(5),rowData(6),rowData(8),rowData(9),rowData(10),rowData(
11),rowData(16),rowData(17)];
                busData = double(busData);
            end

        elseif rowData(1) == "" && busEnd-busStart+1 ~= 300
            busData = [busData; rowData(2),rowData(end-13),rowData(end-
12),rowData(end-10),rowData(end-9),rowData(end-8),rowData(end-
7),rowData(end-2),rowData(end-1)];
            busData = double(busData);

        elseif rowData(1)~=""
            busData = [busData; rowData(1),rowData(end-13),rowData(end-
12),rowData(end-10),rowData(end-9),rowData(end-8),rowData(end-
7),rowData(end-2),rowData(end-1)];
            busData = double(busData);
        end
```

```
    end
end

function [branchData] = writeBranchData(dataFile,branchStart,branchEnd)
    rawData = readlines(dataFile);
    branchData = {};
    for i = branchStart:branchEnd
        rowData = split(rawData(i));
        if rowData(1) == ""
            branchData = [branchData;
rowData(2),rowData(3),rowData(8),rowData(9),rowData(10),rowData(16),rowData
(17)];
            branchData = double(branchData);
        else
            branchData = [branchData;
rowData(1),rowData(2),rowData(7),rowData(8),rowData(9),rowData(15),rowData(
16)];
            branchData = double(branchData);
        end

    end
end
function [numOfBus] = findYBusEntries(data,busNum,busData)
    for i = 1:busNum
        if data == busData(i,1)
            numOfBus = i;
            return
        end
    end
end

function Bprime = createBprime(Y_bus,slackBusNumber)
    B = -imag(Y_bus);
    Bcopy = B;

    Bcopy(slackBusNumber,:) = [];
    Bcopy(:,slackBusNumber) = [];

    Bprime = Bcopy;

end



function BDoubleprime = createBDoublePrime(Y_bus,
pqBuses,pvBuses,slackBus,bPrime,busNum,busData)
      bPrimeCopy = bPrime;
      pqBusNumber = length(pqBuses);
      pvBusNumber = length(pvBuses);
      slackBusNumber = findYBusEntries(slackBus,busNum,busData);
      deleteCount = 0;
      for i=1:pvBusNumber
          numOfBus = findYBusEntries(pvBuses(i),busNum,busData);

          if numOfBus < slackBusNumber
              bPrimeCopy(numOfBus-deleteCount,:) = [];
              bPrimeCopy(:,numOfBus-deleteCount) = [];
          else
              bPrimeCopy(numOfBus-1-deleteCount,:) = [];
              bPrimeCopy(:,numOfBus-1-deleteCount) = [];
          end
```

```
            deleteCount = deleteCount + 1;
        end

        BDoubleprime = bPrimeCopy;
end


function [yBus] = writeOnBus(busData,branchData,busNum)
    yBus = zeros(busNum,busNum);
    % Find Yij entries of Ybus matrix.
    for i=1:length(branchData)
        zBus = 0;
        zBus = branchData(i,3) + j * branchData(i,4);
        busEntity = -(1/zBus);
        if branchData(i,6) ~= 0
            busEntity = busEntity/branchData(i,6);
        end
        yBusX = findYBusEntries(branchData(i,1),busNum,busData);
        yBusY = findYBusEntries(branchData(i,2),busNum,busData);
        yBus(yBusX,yBusY) = yBus(yBusX,yBusY) + busEntity;
        yBus(yBusY,yBusX) = yBus(yBusY,yBusX) + busEntity;
    end

    % Fill the empty entries of lower triangle matrix.
%       yBus = yBus+transpose(yBus);



    % Sum all the columns to find Yii entry.
    for i=1:busNum
        temp = 0;
        for k= 1: busNum
            if i ~= k
                temp = temp + yBus(i,k);
            end
        end
        yBus(i,i) = yBus(i,i) - temp;
    end
    % Add B line to the Yii entries.
    for i=1:length(branchData)
        if branchData(i,5) ~= 0 && branchData(i,6) == 0
            zBus = 0;
            zBus = (j * branchData(i,5))/2;
            yBusX = findYBusEntries(branchData(i,1),busNum,busData);
            yBusY = findYBusEntries(branchData(i,2),busNum,busData);
            yBus(yBusX,yBusX) = yBus(yBusX,yBusX)+zBus;
            yBus(yBusY,yBusY) = yBus(yBusY,yBusY)+zBus;
        end
    end

    % Consider the tap ratios.
    for i=1:length(branchData)
        if branchData(i,6) ~= 0
            tapRatio = branchData(i,6);
            endCoeff = (tapRatio-1)/(tapRatio);
            fromCoeff = (1-tapRatio)/(tapRatio*tapRatio);
            zBus = 0;
            zBus = branchData(i,3) + j * branchData(i,4);
            busEntity = (1/zBus);
            addToFrom = busEntity.*fromCoeff;
```

```
            addToEnd = busEntity.*endCoeff;

            yBusX = findYBusEntries(branchData(i,1),busNum,busData);
            yBusY = findYBusEntries(branchData(i,2),busNum,busData);

            yBus(yBusX,yBusX) = yBus(yBusX,yBusX)+addToFrom;
            yBus(yBusY,yBusY) = yBus(yBusY,yBusY)+addToEnd;
        end
    end


    % Add B shunt to the Yii entries
    for i=1:busNum
        if busData(i,8) ~= 0 || busData(i,9) ~= 0
            zBus = 0;
            zBus = busData(i,8) + j * busData(i,9);

            yBusX = findYBusEntries(busData(i,1),busNum,busData);
%             yBusY = findYBusEntries(busData(i,1),busNum,busData);
            yBus(yBusX,yBusX) = yBus(yBusX,yBusX)+zBus;

        end
    end

end
```