METU EE

EE472
Power System Analysis II

Project 1
Task 1
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 will be read from ieee file and bus admittance matrix will be constructed.

2. Sparsity Patterns

We will investigate the sparsity patterns of bus admittance matrices for 30,57 and 118 bus systems.

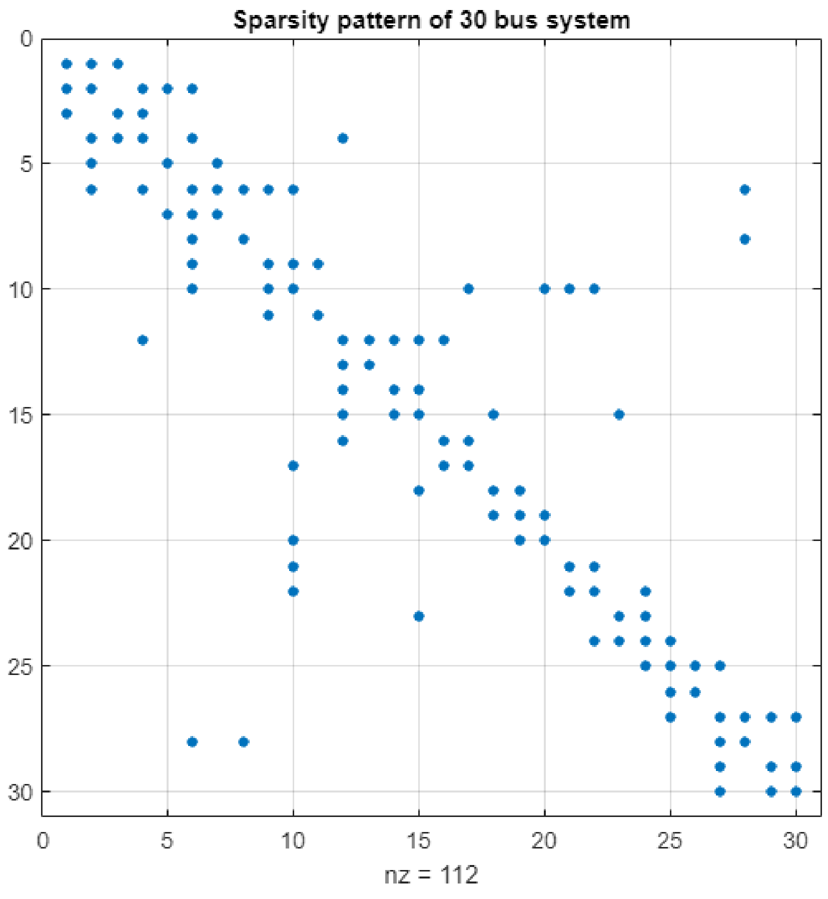

Figure.1: Sparsity pattern of 30 bus system.

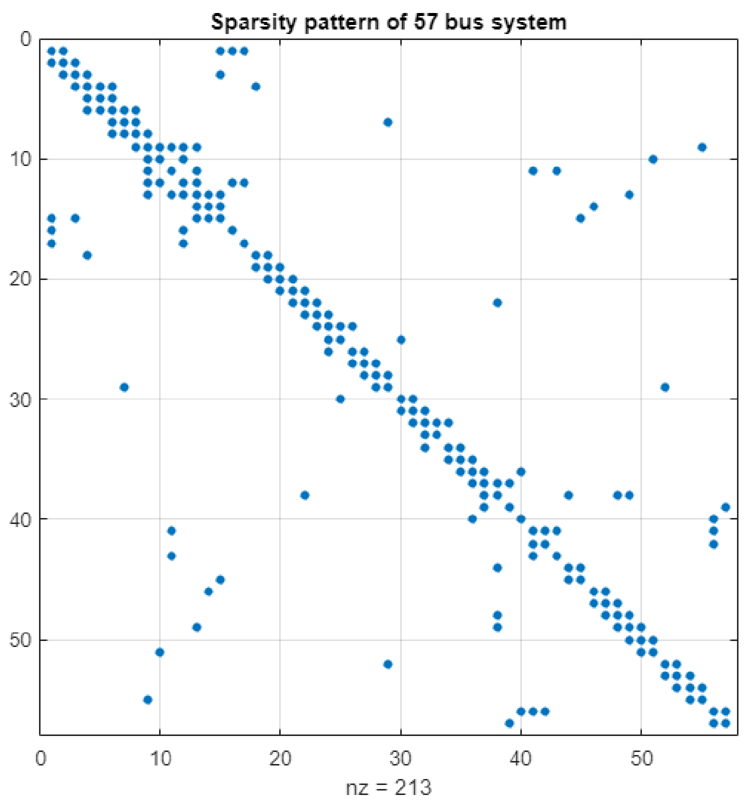

Figure.2: Sparsity pattern of 57 bus system

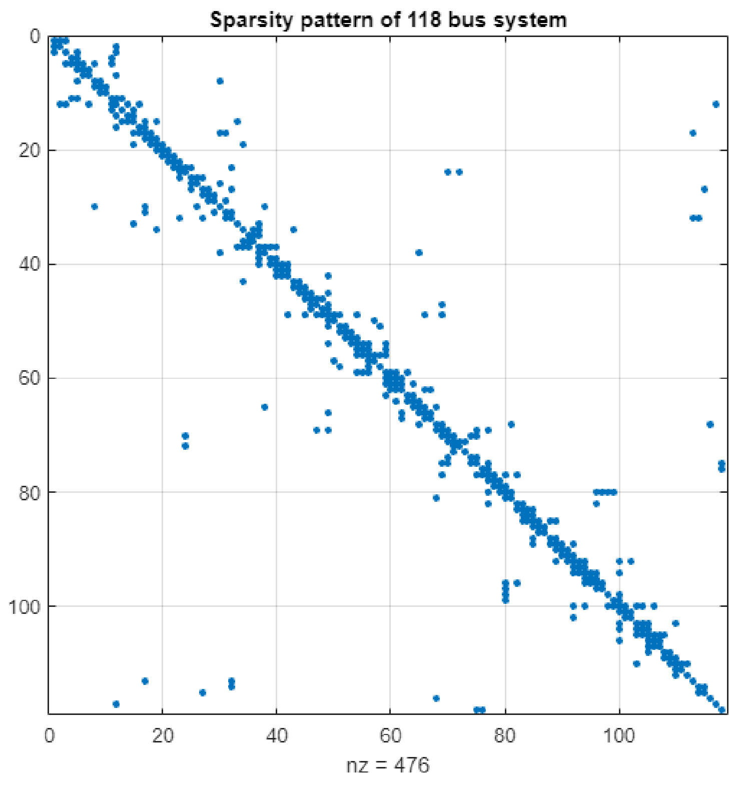

Figure.3: Sparsity pattern of 118 bus system

As can be seen from the above figures, most of the non-diagonal entities of Y Bus Matrix are zero. The reason behind that is, in a topology, a bus is generally connected to very few number of busses. Therefore, there are about 5 non-diagonal entities are not zero in Y Bus matrix for a bus. On the other hand, we observe that all of the diagonal entities are non-zero. This occurs due to the connection to the bus to system. Connection to system creates an admittance for a bus and we observe non-zero diagonal entity.

	30 bus system	**57 bus system**	**118 bus system**
Non-zero entities	112	213	476

Table.1: Non-zero entities of Y Bus Matrix.

3. Performance Comparison

	30 bus system	**57 bus system**	**118 bus system**
From Z Bus (sec)	0.000430	0.000547	0.001479
Direct Division (sec)	0.000415	0.000509	0.000809

Table.2: Time taken to compute bus voltage vector.

In the table.2 above, time taken to compute bus voltage vector is compared with two different method. In the first method (calculation from Z Bus), Y Bus is inverted and multiplied with current vector. As there are lots of calculation in this method. It is expected to take longer time. In the second method, Y Bus matrix is directly divided by current vector. Hence, it requires less time. In general, power systems have lots of busses and analysis is important. Therefore, time taken for computation should be minimized. Considering systems might be larger than 118 busses, we should also consider time taken for these systems.

4. Sparse Structure

In bus admittance matrix, terms that are non-zero indicates the connection of related bus. Hence, total number of non-zero entities is equal to:

$$Non - zero\ entities = Bus\ Number + 2\ x\ Branch\ Number - Shunt\ Connections$$

For example, for the data with 118 busses:
118 + 2 x 186 – 14 = 476. This result can also be verified from figure.3.

The number of elements to store can be seen for figure.1, figure.2, figure.3.

	30 bus system	**57 bus system**	**118 bus system**
Non-zero entities	112	213	476

Table.3: The number of elements to store in respective bus systems.

5. Code for Y Bus Construction

```
function [Y_bus] = e230405_Aydin(cdf_path)
    dataFile = cdf_path;
    format long;
    [sBase] = readTitle(dataFile);
    [busStart,busEnd,branchStart,branchEnd] = getLineNum(dataFile);
    busNum = busEnd-busStart+1;
    busData = writeBusData(dataFile,busStart,busEnd);
    branchData = writeBranchData(dataFile,branchStart,branchEnd);
    Y_bus = writeOnBus(busData,branchData,busNum);
end

function [sBase] = readTitle(dataFile)
    rawData = readlines(dataFile);
    splitRawData = split(rawData(1));
    sBase = splitRawData(5);
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 rowData(1) == ""
            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);
        end
        if 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));
        branchData = [branchData;
rowData(2),rowData(3),rowData(8),rowData(9),rowData(10),rowData(16),rowData(17)];
        branchData = double(branchData);
    end
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
        yBus(branchData(i,1),branchData(i,2)) = 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;
            yBus(branchData(i,1),branchData(i,1)) =
yBus(branchData(i,1),branchData(i,1))+zBus;
            yBus(branchData(i,2),branchData(i,2)) =
yBus(branchData(i,2),branchData(i,2))+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;

            yBus(branchData(i,1),branchData(i,1)) =
yBus(branchData(i,1),branchData(i,1))+addToFrom;
            yBus(branchData(i,2),branchData(i,2)) =
yBus(branchData(i,2),branchData(i,2))+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);
            yBus(busData(i,1),busData(i,1)) =
yBus(busData(i,1),busData(i,1))+zBus;
            % yBus(branchData(i,2),branchData(i,2)) =
yBus(branchData(i,2),branchData(i,2))+zBus;
        end
    end

end
```