EE-472 Power System Analysis II

Project I: Submitting The Bus Admittance Matrix

Akif ARSLAN
1949080

March 25, 2019

Introduction

In this assignment, working with IEEE common data format and obtaining bus admittance matrix is studied. MATLAB is used as the computational tool. Sparsity pattern of $\mathbf{Y_{BUS}}$ is observed and plotted. When creating computational algorithm, speed was one of the concerns as well as accuracy. Accuracy of the results are compared with the ones obtained by using the PET software and confirmed.

a: Sparsity pattern plot of $\mathbf{Y_{BUS}}$

- By using MATLAB and provided **IEEE-300 BUS SYSTEM** file, the $\mathbf{Y_{BUS}}$ is constructed as a sparse pattern rather than first constructing a 300x300 zero matrix then converting to a sparse matrix by considering computational time and memory usage efficiency.
- By using MATLAB's **sparse** function, a 300x300 sparse $\mathbf{Y_{BUS}}$ matrix is obtained from the created $\mathbf{Y_{BUS}}$ matrix on the basis of given **cdf** file. $\mathbf{Y_{BUS}}$ matrix's sparsity pattern can be seen at Figure 1 as 3D, 2D and histogram respectively. Plots are obtained by taking absolute value of complex $\mathbf{Y_{BUS}}$ matrix.

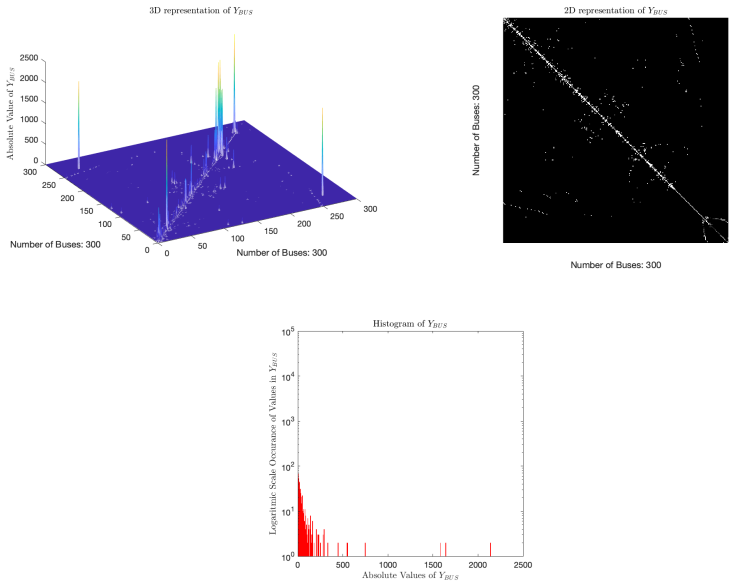

Figure 1: Sparsity Pattern of 300x300 $\mathbf{Y_{BUS}}$

- The MATLAB algorithm created for 300-BUS system is also tested for the cases 118-BUS and 14-BUS system. Sparsity pattern of the both system are shown below at Figure 2 and Figure 3 respectively. It's noted that; as system size increases, sparsity pattern of the $\mathbf{Y_{BUS}}$ also increases.

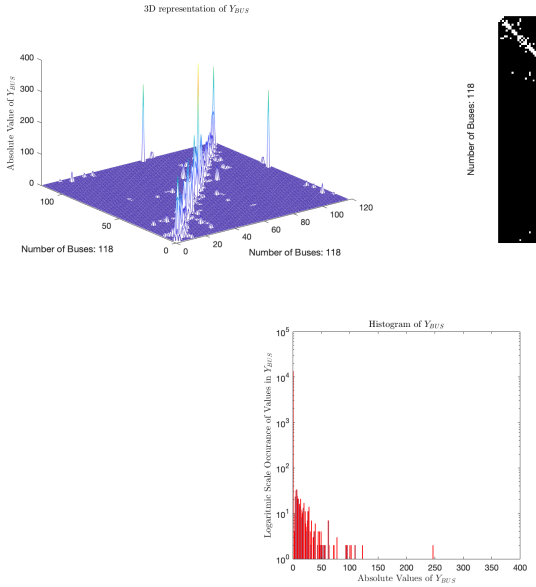

Figure 2: 118-BUS SYSTEM Sparsity Pattern

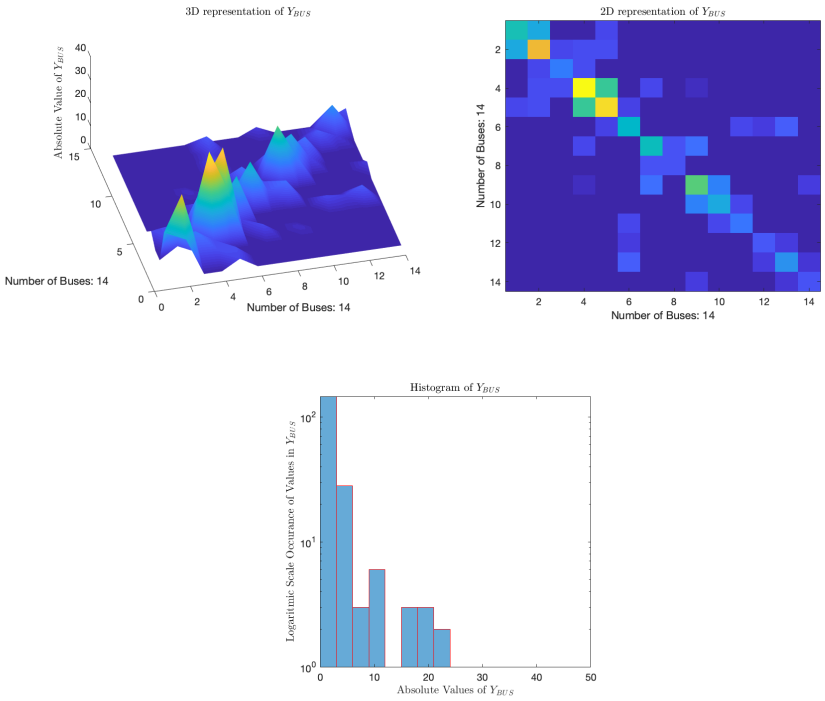

Figure 3: 14-BUS SYSTEM Sparsity Pattern

b: Solution duration of I =$\mathbf{Y_{BUS}V}$

By using MATLAB's **tic-toc** function, solution time is found for a 300-BUS SYSTEM as;
for $V = Y_{BUS}^{-1} \cdot I$: 0.053889 seconds.
for $V = Y_{BUS} \setminus I$: 0.005499 seconds

c: Number of elements used to store fisparse structurefi.

- Constructed $\mathbf{Y_{BUS}}$ matrix from **IEEE-300 BUS SYSTEM** by MATLAB code:
  $1122 \times 4 = 4488$ matrix and 35904 Bytes
- 300x300 $\mathbf{Y_{BUS}}$ matrix (created by using MATLAB's **full** function):
  $300 \times 300 = 90.000$ elements and 1440000 Bytes
- Sparse $\mathbf{Y_{BUS}}$ matrix:
  $300 \times 300 = 90.000$ elements, 29336 Bytes

d: Comments

comments on Part a.

In a huge power system network there is no direct connection between most of the busses. Hence bus admittance matrix includes mostly sparsity. That's why keeping only non-zero elements and their indices in the memory save a lot of space in a computer's memory.

comments on Part b.

According to MATLAB's website[1], **inv** operation calculates a complete inverse of a given matrix, whereas, '\' operant uses ***Gaussian elimination***, without explicitly forming the inverse matrix to solve system of linear equations. Hence, this decreases the computation time for large matrices significantly.

comments on Part c.

As it's stated at section **a**, bus admittance matrices are mostly composed of zeros except diagonal entries. Hence keeping the non-zero elements and their addresses in the memory is an efficient way of saving information. When we convert a matrix to a sparse matrix by using MATLAB's **sparse** function, and then inspect it by using **whos** function, MATLAB says it still includes 300x300 elements, but size of it seems decreased from 1440000 Bytes to 29336 Bytes, mean $\approx$ 50 times reduction.

References

[1] `https://www.mathworks.com/help/matlab/ref/inv.html?searchHighlight=inv`

MATLAB CODE

```
function [Y_bus, t_solution, number_of_non_zero_elements, V_bus] = ...
    e194908_arslan_PF(argin1)
% EE 472 Power System Analysis II Fall 2019
% Term Project Part I: Submitting Bus Admittance Matrix.
% Instructor         : Assist. Prof. Murat GOL
% Course Assistant.  : Mustafa Erdem Sezgin
%=========================================================================
% Akif ARSLAN 1949080
%=========================================================================
% FORM Y_BUS
tic
ieee300 = fopen(argin1,'r');
% Find where bus data starts.
while true
    line_data = fgetl(ieee300);
    if length(line_data) ≥ 3
        if strcmpi(line_data(1:3),'BUS')
            break
        end
    end
end
% allocate indexing matrix for faster iteration
indexing_matrix = uint16(zeros(1000,1));
shunt_G_and_B = zeros(1000,1);
i = uint16(1);
while true
    line_data = fgetl(ieee300);
    % break the loop when BUS DATA finishes
    if length(line_data) < 50
        % cut unused parts
        indexing_matrix = indexing_matrix(1:i-1);
        shunt_G_and_B = shunt_G_and_B(1:i-1);
        number_of_buses = i-1;
        break
    end
    indexing_matrix(i) = str2double(line_data(1:4));
    shunt_G_and_B(i) = str2double(line_data(107:114)) + ...
        str2double(line_data(115:122))*1i;
    i = i + 1;
end
% Go where BRANCH DATA STARTS
while true
    line_data = fgetl(ieee300);
    if length(line_data) ≥ 6
        if strcmpi(line_data(1:6),'BRANCH')
            break
        end
    end
end
% allocate memory for Y_BUS matrix for faster iteration
Y_BUS = zeros(number_of_buses*4,4);
%Create a 2x2 temporary pair admittance matrix
Y_BUS_temp = zeros(2,2);
k=uint16(1);
while true
```

```
    line_data = fgetl(ieee300);
    % break the loop when BRANCH DATA finishes
    if length(line_data) < 50
        if sum(Y_BUS(:,1) == 0,1)
            % Cut unused parts of the Y_BUS
            zero_cut = find(Y_BUS(:,1)==0, 1, 'first');
            Y_BUS = Y_BUS(1:zero_cut-1,:);
        end
        break
    end
    %Use indexing as given in BUS data order.
    Yi = find(indexing_matrix == str2double(line_data(1:4))); % "from" bus
    Yj = find(indexing_matrix == str2double(line_data(6:9))); % "to" bus
    %Get Resistance and Reactance data and turn into line admittance.
    %R = str2double(line_data(20:29));
    %X = str2double(line_data(30:40));
    line_admittance = 1/(str2double(line_data(20:29)) + ...
        str2double(line_data(30:40))*1i);
    %Get line charing B data and divide by 2
    line_charging = (str2double(line_data(41:50))/2)*1i;
    % If there is any tap or phase shifter include their effects.
    switch str2double(line_data(19))
        case 0 % 0 ==> A line.
            Y_BUS_temp(1,1) = line_admittance + line_charging + ...
                shunt_G_and_B(Yi);% Yii
            Y_BUS_temp(1,2) = -line_admittance; % Yij
            Y_BUS_temp(2,1) = -line_admittance; %Yji
            Y_BUS_temp(2,2) = line_admittance + line_charging + ...
                shunt_G_and_B(Yj); % Yjj
        case {1,2,3} % 1,2,3 ==> There is a tap changer.
            tap = str2double(line_data(77:82));
            Y_BUS_temp(1,1) = (line_admittance/(tap^2)) + ...
                shunt_G_and_B(Yi);% Yii
            Y_BUS_temp(1,2) = -line_admittance/tap; % Yij
            Y_BUS_temp(2,1) = -line_admittance/tap; % Yji
            Y_BUS_temp(2,2) = line_admittance + shunt_G_and_B(Yj); % Yjj
        case 4 % 4 ==> There is a phase shifter.
            tap = str2double(line_data(77:82));
            p_shift = str2double(line_data(84:90));
            p_shift = p_shift*pi/180;
            Y_BUS_temp(1,1) = line_admittance/tap^2 + ...
                shunt_G_and_B(Yi);% Yii
            Y_BUS_temp(1,2) = -line_admittance/(cos(p_shift) - ...
                sin(p_shift)*1i); % Conjugate
            Y_BUS_temp(2,1) = -line_admittance/(cos(p_shift) + ...
                sin(p_shift)*1i);
            Y_BUS_temp(2,2) = line_admittance + shunt_G_and_B(Yj);
        otherwise
            disp('line information has not found')
    end
    % Once shunt values are used remove them to avoid adding again at next
    % iterations.
    shunt_G_and_B(Yi) = 0;
    shunt_G_and_B(Yj) = 0;

    is_Yi_used = logical(sum(any(Y_BUS(:,1:2) == Yi)));
    is_Yj_used = logical(sum(any(Y_BUS(:,1:2) == Yj)));
    % If both Yi and Yj busses are used at previous iteration, don't create
    % new Yii and Yjj. Only create Yij and Yji.
```

```
106     if is_Yi_used && is_Yj_used
107         index_temp = find(sum(Y_BUS(:,[1,2])==Yi,2)==2);
108         Y_BUS(index_temp,3) = Y_BUS(index_temp,3) + real(Y_BUS_temp(1,1));
109         Y_BUS(index_temp,4) = Y_BUS(index_temp,4) + imag(Y_BUS_temp(1,1));
110 
111         index_temp = find(sum(Y_BUS(:,[1,2])==Yj,2)==2);
112         Y_BUS(index_temp,3) = Y_BUS(index_temp,3) + real(Y_BUS_temp(2,2));
113         Y_BUS(index_temp,4) = Y_BUS(index_temp,4) + imag(Y_BUS_temp(2,2));
114     else
115         % If one of the buses are used to calculate Y_BUS in previous
116         % steps, don't creat new entry for it. Add its new admittance value
117         % to previous one.
118         if is_Yi_used || is_Yj_used
119             % Check, which one of Yi and Yj are used before
120             if is_Yi_used
121                 index_temp = find(sum(Y_BUS(:,[1,2])==Yi,2)==2);
122                 Y_BUS(index_temp,3) = Y_BUS(index_temp,3) + ...
                        real(Y_BUS_temp(1,1));
123                 Y_BUS(index_temp,4) = Y_BUS(index_temp,4) + ...
                        imag(Y_BUS_temp(1,1));
124 
125                 Y_BUS(k,1) = Yj;
126                 Y_BUS(k,2) = Yj;
127                 Y_BUS(k,3) = real(Y_BUS_temp(2,2));
128                 Y_BUS(k,4) = imag(Y_BUS_temp(2,2));
129                 k = k + 1;
130             else % if is_Yj_used
131                 Y_BUS(k,1) = Yi;
132                 Y_BUS(k,2) = Yi;
133                 Y_BUS(k,3) = real(Y_BUS_temp(1,1));
134                 Y_BUS(k,4) = imag(Y_BUS_temp(1,1));
135                 k = k + 1;
136 
137                 index_temp = find(sum(Y_BUS(:,[1,2])==Yj,2)==2);
138                 Y_BUS(index_temp,3) = Y_BUS(index_temp,3) + ...
                        real(Y_BUS_temp(2,2));
139                 Y_BUS(index_temp,4) = Y_BUS(index_temp,4) + ...
                        imag(Y_BUS_temp(2,2));
140             end
141             % If neighter Yi or Yj bus number is used previously,
142             %construct new Yii and Yjj.
143         else
144             Y_BUS(k,1) = Yi;
145             Y_BUS(k,2) = Yi;
146             Y_BUS(k,3) = real(Y_BUS_temp(1,1));
147             Y_BUS(k,4) = imag(Y_BUS_temp(1,1));
148             k = k + 1;
149 
150             Y_BUS(k,1) = Yj;
151             Y_BUS(k,2) = Yj;
152             Y_BUS(k,3) = real(Y_BUS_temp(2,2));
153             Y_BUS(k,4) = imag(Y_BUS_temp(2,2));
154             k = k + 1;
155         end
156     end
157     % Cunstruc Yij = Yji element for any of the cases.
158     Y_BUS(k,1) = Yi;
159     Y_BUS(k,2) = Yj;
160     Y_BUS(k,3) = real(Y_BUS_temp(1,2));
```

```
161     Y_BUS(k,4) = imag(Y_BUS_temp(1,2));
162     k = k + 1;
163 
164     Y_BUS(k,1) = Yj;
165     Y_BUS(k,2) = Yi;
166     Y_BUS(k,3) = real(Y_BUS_temp(2,1));
167     Y_BUS(k,4) = imag(Y_BUS_temp(2,1));
168     k = k + 1;
169 end
170 % Close the text file after iteration.
171 fclose(ieee300);
172 % Use "sparse" fuction to obtain Y_BUS as a sparse matrix. sparse(i,j,v)
173 Y_bus = sparse(Y_BUS(:,1),Y_BUS(:,2), Y_BUS(:,3) + Y_BUS(:,4)*1i);
174 number_of_non_zero_elements = nnz(Y_bus);
175 % Create a random I vector.
176 I = rand(number_of_buses,1);
177 % Calculate V_bus
178 V_bus = Y_bus\I;
179 t_solution = toc;
```