



Middle East Technical University
Electrical & Electronics Engineering
Department

EE471 – Power System Analysis I

Term Project Report

Yasin Enes Çalışkan

2304319

Introduction

In this project, it is expected from us to create Ybus and Zbus matrices of a system with bus and line information. The functions used to create the desired matrices within the scope of the project were written using MATLAB. System information is given in an excel document. In this report, the functions created to meet the project requirements and their outputs are detailed.

Implementation

In the implementation of the project code, I first created Ybus, then I performed the LU factorization. Then I created Zbus by applying forward and backward substitution to the matrices I created with LU factorization.

a. Creation of Ybus

First of all, I started to create the project code by reading the data from the excel file containing the system data. I used the "readmatrix" function to convert the data I read from the Excel file into matrix. In this way, I obtained "data_line" and "data_bus" matrices from two different sheets. While creating Ybus, I divided the data in each column in excel into a separate matrix for convenience. Since the given values are only numbers, I converted some of the numbers into complex numbers by multiplying "1i" according to the given data type.

After obtaining all the data, I first created a matrix consisting of zeros in the length and width of the total bus number to set up Ybus. Then I created a for loop to fill this all-zero matrix. In this for loop, the relevant entry in Ybus is filled by checking whether the tap ratio value in each line given in the line sheet is 1 or not. To complete Ybus, I added the shunt admittance value I obtained from the bus sheet to all diagonal entries of Ybus. The code I created as a result of all these processes is as follows;

```
clc
close all;
filepath= input('Please type the source folder!\n','s');
path=convertCharsToStrings(filepath);
data_line= readmatrix(path,'sheet','line');
data_bus= readmatrix(path,'sheet','bus');

FromBus = data_line(:,1);
ToBus= data_line(:,2);
Sres = data_line(:,3);           %series resistance
Srec = 1i.*data_line(:,4);       %series reactance
y=1./(Sres+Srec);                %series admittance
B = 1i.*(data_line(:,5)./2);     %half of the total charging admittance
tap = data_line(:,6);            %tap ratio

Shuntcon = data_bus(:,2);         %shant conductance
Shuntsus = 1i.*data_bus(:,3);    %shunt susceptance
```

```

ShuntAdm = Shuntcon + Shuntsus;           %shunt admittance

bus=length(data_bus(:,1));
line=length(FromBus);
Ybus=zeros(bus,bus);

for k=1:line
    p=FromBus(k);
    q=ToBus(k);
    if tap(k)==1
        Ybus(p,p)=Ybus(p,p)+y(k)+B(k);
        Ybus(q,q)=Ybus(q,q)+y(k)+B(k);
        Ybus(p,q)=Ybus(p,q)-y(k);
        Ybus(q,p)=Ybus(p,q);
    else
        %considering tap changing
        Ybus(p,p)=Ybus(p,p)+y(k)/(tap(k)^2)+B(k);
        Ybus(q,q)=Ybus(q,q)+y(k)+B(k);
        Ybus(p,q)=Ybus(p,q)-y(k)/tap(k);
        Ybus(q,p)=Ybus(p,q);
    end
end

for l=1:bus
    Ybus(l,l)=Ybus(l,l)+ShuntAdm(l);       %addition of shunt admittance to
diagonals
end

```

b. LU Factorization

I created the function that will perform LU factorization in a separate m file from the main code. In the main code, I performed my operation by calling the function. In order to convert the Ybus I obtained in the previous code blocks into lower and upper triangular form, I first created a "U" matrix consisting of all zeros and then an identity matrix with one diagonal. I wrote a function consisting of nested for loops to fill the matrices I created in the form I want. Two matrices formed at the end of the for loop form the function output. The "LUfactorization" function I created and code block in main code are respectively as follows;

```

function [L,U]= LUfactorization(A)
m=height(A);
U=zeros(m);
L=eye(m);

for j=1:m
    U(1,j)=A(1,j);
end
for i=2:m
    for j=1:m
        for k=1:i-1
            s1=0;
            if k==1
                s1=0;
            else
                for p=1:k-1

```

```

        s1=s1+L(i,p)*U(p,k);
    end
    end
    L(i,k)=(A(i,k)-s1)/U(k,k);
end
for k=i:m
    s2=0;
    for p=1:i-1
        s2=s2+L(i,p)*U(p,k);
    end
    U(i,k)=A(i,k)-s2;
end
end
end

```

```
[L,U]=LUfactorization(Ybus); %LU factorization function called
```

c. Forward and Backward Substitutions

I created these parts of the code in two separate m files as two functions separate from the main code. Forward and backward substitutions are simply used to solve systems of equations created with lower and upper triangular matrices. Therefore, when applying forward substitution to the L matrix to create a Zbus, the row containing the number of the bus whose impedance is to be found in the result matrix is selected so that the other rows are zero. To find the impedance of all the buses and create the Zbus, the forward substitution function is called sequentially for each bus, using the for loop. Likewise, the matrix, which has the same length and width as the number of buses obtained after forward substitution, is used as the input matrix for backward substitution. Zbus is obtained as a result of the operations. The functions I created for forward and backward substitutions and the code blocks I created to call functions in the main code are as follows, respectively;

```

function x= ForwardSub(a,b)
    n=length(b);
    x=zeros(n,1);
    x(1)=b(1)/a(1,1);
    for i=2:n
        x(i)=(b(i)-a(i,1:i-1)*x(1:i-1))./a(i,i);
    end

```

```

function y = BackwardSub(a,b)
    n = length(b);
    y=zeros(n,1);
    y(n) = b(n)/a(n,n);
    for i=n-1:-1:1
        y(i)=(b(i)-a(i,i+1:n)*y(i+1:n))/a(i,i);
    end

```

```

b=eye(bus); %creating an identity matrix
x=zeros(bus,bus);

for n=1:bus
    x(:,n)= ForwardSub(L,b(:,n)); %forward substitution function called
end

Zbus=zeros(bus,bus);
for s=1:bus
    Zbus(:,s)=BackwardSub(U,x(:,s)); %backward substitution function called
end

```

Outputs

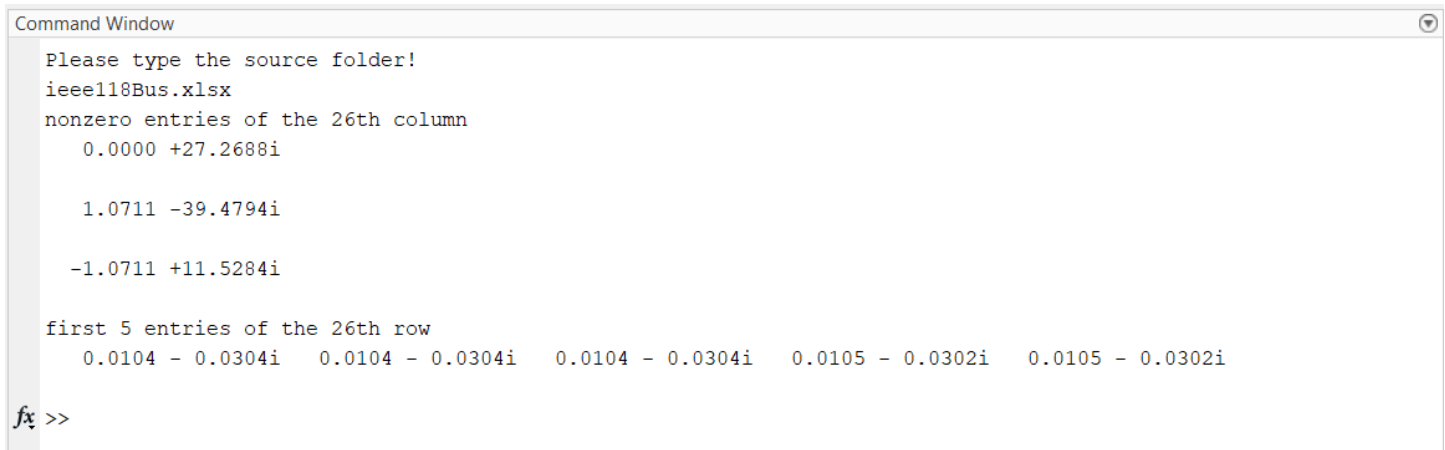
The code block created to observe the desired outputs on the terminal and plot the sparsity pattern is as follows;

```

%outputs
disp('nonzero entries of the 26th column')
for m=1:bus
    if Ybus(m,26)~=0
        disp(Ybus(m,26)) %displaying nonzero entries of the 26th column
    end
end
disp('first 5 entries of the 26th row')
disp(Zbus(26,1:5)) %%displaying first 5 entries of the 26th row
spy(Ybus)

```

Displayed outputs are as follows;



```

Command Window
Please type the source folder!
ieee118Bus.xlsx
nonzero entries of the 26th column
0.0000 +27.2688i

1.0711 -39.4794i

-1.0711 +11.5284i

first 5 entries of the 26th row
0.0104 - 0.0304i    0.0104 - 0.0304i    0.0104 - 0.0304i    0.0105 - 0.0302i    0.0105 - 0.0302i
fx >>

```

Figure 1. Screen image for outputs of the MATLAB code.

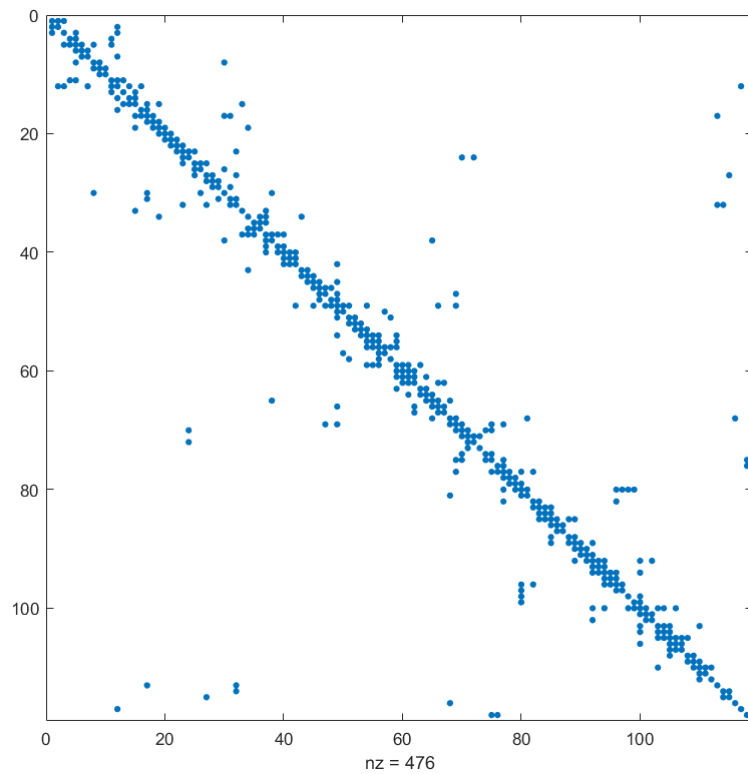


Figure 2. Sparsity pattern plot for Y_{bus} .

As can be seen from “nz” value which is under the sparsity pattern plot, number of elements used to store the same Y_{BUS} matrix as a ‘sparse structure’ is 476.

Appendix (All Code Files)

e230431_Ybus.m

```

clc
close all;
filepath= input('Please type the source folder!\n','s');
path=convertCharsToStrings(filepath);
data_line= readmatrix(path,'sheet','line');
data_bus= readmatrix(path,'sheet','bus');

FromBus = data_line(:,1);
ToBus= data_line(:,2);
Sres = data_line(:,3);           %series resistance
Srec = 1i.*data_line(:,4);       %series reactance
y=1./(Sres+Srec);                %series admittance
B = 1i.*(data_line(:,5)./2);     %half of the total charging admittance
tap = data_line(:,6);            %tap ratio

Shuntcon = data_bus(:,2);        %shant conductance
Shuntsus = 1i.*data_bus(:,3);    %shunt susceptance
ShuntAdm = Shuntcon + Shuntsus;  %shunt admittance

```

```

bus=length(data_bus(:,1));
line=length(FromBus);
Ybus=zeros(bus,bus);

for k=1:line
    p=FromBus(k);
    q=ToBus(k);
    if tap(k)==1
        Ybus(p,p)=Ybus(p,p)+y(k)+B(k);
        Ybus(q,q)=Ybus(q,q)+y(k)+B(k);
        Ybus(p,q)=Ybus(p,q)-y(k);
        Ybus(q,p)=Ybus(p,q);
    else
        %considering tap changing
        Ybus(p,p)=Ybus(p,p)+y(k)/(tap(k)^2)+B(k);
        Ybus(q,q)=Ybus(q,q)+y(k)+B(k);
        Ybus(p,q)=Ybus(p,q)-y(k)/tap(k);
        Ybus(q,p)=Ybus(p,q);
    end
end

for l=1:bus
    Ybus(l,l)=Ybus(l,l)+ShuntAdm(l);    %addition of shunt admittance to
diagonals
end

[L,U]=LUfactorization(Ybus);    %LU factorization function called

b=eye(bus);    %creating an identity matrix
x=zeros(bus,bus);

for n=1:bus
    x(:,n)= ForwardSub(L,b(:,n));    %forward substitution function called
end

Zbus=zeros(bus,bus);
for s=1:bus
    Zbus(:,s)=BackwardSub(U,x(:,s));    %backward substitution function called
end

%outputs
disp('nonzero entries of the 26th column')
for m=1:bus
    if Ybus(m,26)~=0
        disp(Ybus(m,26))    %displaying nonzero entries of the 26th column
    end
end
disp('first 5 entries of the 26th row')
disp(Zbus(26,1:5))    %%displaying first 5 entries of the 26th row
spy(Ybus)

```

LUfactorization.m

```

function [L,U]= LUfactorization(A)
m=height(A);

```

```

U=zeros(m);
L=eye(m);

for j=1:m
    U(1,j)=A(1,j);
end
for i=2:m
    for j=1:m
        for k=1:i-1
            s1=0;
            if k==1
                s1=0;
            else
                for p=1:k-1
                    s1=s1+L(i,p)*U(p,k);
                end
            end
            L(i,k)=(A(i,k)-s1)/U(k,k);
        end
        for k=i:m
            s2=0;
            for p=1:i-1
                s2=s2+L(i,p)*U(p,k);
            end
            U(i,k)=A(i,k)-s2;
        end
    end
end
end

```

ForwardSub.m

```

function x= ForwardSub(a,b)
n=length(b);
x=zeros(n,1);
x(1)=b(1)/a(1,1);
for i=2:n
    x(i)=(b(i)-a(i,1:i-1)*x(1:i-1))./a(i,i);
end

```

BackwardSub.m

```

function y = BackwardSub(a,b)
n = length(b);
y=zeros(n,1);
y(n) = b(n)/a(n,n);
for i=n-1:-1:1
    y(i)=(b(i)-a(i,i+1:n)*y(i+1:n))/a(i,i);
end

```
