```
clc;clear;
%% Standardizing generator sub-transient reactances
s_new = 100; % MVA
s_old = [250; 100; 80; 50; 50;]; % MVA

x1 = 15/100;
x2 = 12/100;
x3 = 10/100;
x4 = 9/100;
x5 = 8/100;
x = [1 x1; 2 x2; 3 x3; 4 x4; 5 x5];

% converting xolds to xnews
x(:,2) = (x(:,2)./s_old)*s_new;
[r_x,~] = size(x);
%% Y bus calculation

% import data as table
linedata = readmatrix("line_data.xlsx");
% for k=1:r_x
%     linedata(end+1,1) = x(k,1);
%     linedata(end,2) = x(k,1);
%     linedata(end,4) = x(k,2);
% end
[r_data,~] = size(linedata);

% Preprocessing
number_of_buses = max(max(linedata(:,1:2))); % picking the highest numbered bus
y_bus = zeros(number_of_buses);
[length_y_bus,~] = size(y_bus);

% formatting the impedances
R = linedata(:,3);
X = linedata(:,4);
B = linedata(:,5);
impedances = [R X.*1i B*1i./2];

% calculating the diagonal elements
for m=1:length_y_bus % m is the index of the y bus (row)
    z = zeros(1,3); % initialize 3 columns (R,X,B)
    for n=1:r_data % n is the index of the linedata (row)
        if linedata(n,1)==m||linedata(n,2)==m
            z = [z; linedata(n,3:5)]; % collecting all the impedances
        end
    end
    [r_z,~] = size(z);
    for impedance=1:r_z
        R = z(impedance,1)
        X = z(impedance,2)
        B = z(impedance,3)
        if R~=0 || X~=0
            y_bus(m,m) = y_bus(m,m) + 1./(R+X.*1i) + B.*1i/2;
        end
    end
end

% calculating off diagonal elements
nodes = linedata(:,1:2);
y_off = zeros(14,14);
impedances = impedances(:,1)+impedances(:,2);
[r,~] = size(linedata);


for m=1:r
    nodes(m,1)
    nodes(m,2)
    y_off(nodes(m,1),nodes(m,2)) = -1/impedances(m);
    y_off(nodes(m,2),nodes(m,1)) = y_off(nodes(m,1),nodes(m,2));
end

y_bus = y_bus + y_off;
%% z bus calculation
z_bus = y_bus^-1;

% adding the generator sub-transient reactances to the z-bus matrix
for k=1:length(x)
    z_bus(k,k) = x(k,2)+z_bus(k,k);
end

%% prefault voltages

v = readmatrix('voltages.xlsx'); % prefault voltages
v_magnitudes = v(:,1);
v_angles = deg2rad(v(:,2));

% converting to rectangular
vx = v_magnitudes.*cos(v_angles);
vy = v_magnitudes.*sin(v_angles);
v = vx+vy*1i;

%% q1: sags

%Calculate the voltage sag at bus 4 and bus 13 when a three-phase-fault occurs at each bus
%in the system
sag_4_q1 = zeros(14,1);
sag_13_q1 = zeros(14,1);
for bus=1:14
```

```
    %if bus~=4|bus~=13
        sag_4_q1(bus) = (1-z_bus(4,bus)/z_bus(bus,bus))*v(bus);
        sag_13_q1(bus) = (1-z_bus(13,bus)/z_bus(bus,bus))*v(bus);
    %end
end

% sag magnitudes
sag_4_q1 = abs(sag_4_q1);
sag_13_q1 = abs(sag_13_q1);
%% q2: sags (lines 4-5 and 6-13 are open)

% forming the y,z buses
y_bus_q2 = find_y_bus('line_data_q2.xlsx');
z_bus_q2 = y_bus_q2^-1;
% adding the generator sub-transient reactances to the z-bus matrix
for k=1:length(x)
    z_bus_q2(k,k) = x(k,2)+z_bus_q2(k,k);
end

%faults
sag_4_q2 = zeros(14,1);
sag_13_q2 = zeros(14,1);
for bus=1:14
    %if bus~=4|bus~=13
        sag_4_q2(bus) = (1-z_bus_q2(4,bus)/z_bus_q2(bus,bus))*v(bus);
        sag_13_q2(bus) = (1-z_bus_q2(13,bus)/z_bus_q2(bus,bus))*v(bus);
    %end
end
% sag magnitudes
sag_4_q2 = abs(sag_4_q2);
sag_13_q2 = abs(sag_13_q2);
%% q3: sags (no x3 and x5)

s_old = [250 100 50]; % MVA
x_q3 = [x1 x2 x4]; % removing the sub transient reactances
x_q3 = x_q3*1i.*s_new./s_old; % converting to new MVA base

z_bus_q3 = y_bus^-1;
% adding the generator sub-transient reactances to the z-bus matrix
for k=1:length(x_q3)
    z_bus_q3(k,k) = x_q3(k)+z_bus_q3(k,k);
end

% faults
sag_4_q3 = zeros(14,1);
sag_13_q3 = zeros(14,1);
for bus=1:14
    %if bus~=4|bus~=13
        sag_4_q3(bus) = (1-z_bus_q3(4,bus)/z_bus_q3(bus,bus))*v(bus);
        sag_13_q3(bus) = (1-z_bus_q3(13,bus)/z_bus_q3(bus,bus))*v(bus);
    %end
end
% sag magnitudes
sag_4_q3 = abs(sag_4_q3);
sag_13_q3 = abs(sag_13_q3);
%% q4: no of sags

d = linedata(:,6); %line distance
frequency_100 = linedata(:,7); % faults/100km/yr

frequency_d = frequency_100.*d./100;
[r,c] = size(nodes);
% calculating average sags
average_sags_4 = zeros(r,3); % will be of the same size as the nodes matrix
average_sags_4(:,1:2) = nodes;
average_sags_13 = zeros(r,3); % will be of the same size as the nodes matrix
average_sags_13(:,1:2) = nodes;

% sags at lines for bus 4
for k=1:r
    sag1 = sag_4_q1(nodes(k,1));
    sag2 = sag_4_q1(nodes(k,2));
    average_sags_4(k,3) = 0.5*(sag1+sag2);
end

% sags at lines for bus 13
for k=1:r
    sag1 = sag_13_q1(nodes(k,1));
    sag2 = sag_13_q1(nodes(k,2));
    average_sags_13(k,3) = 0.5*(sag1+sag2);
end

frequency_table = [nodes average_sags_4(:,3) average_sags_13(:,3) frequency_d];
f_sag_4 = 0;
f_sag_13 = 0;
% frequency of sag under 40%=0.4 pu
[r,~] = size(frequency_table);
for k=1:r
    if frequency_table(k,3)<0.4
        f_sag_4 = f_sag_4 + frequency_table(k,5);
    end
    if frequency_table(k,4)<0.4
        f_sag_13 = f_sag_13 + frequency_table(k,5);
    end
end

%% q4: Bar chart

```

```
% creating chart
chart_intervals = 0:0.1:1; % each number represents an upper bound for an interval
chart_intervals = chart_intervals';
chart = [chart_intervals zeros(length(chart_intervals),2)];
[r,~] = size(chart);

% filling the chart
% sags at bus 4
for k=1:length(sag_4_q3)
    for index=2:r
        if chart(index-1,1)<sag_4_q3(k)&&sag_4_q3(k)<chart(index,1)
            chart(index,2) = chart(index,2)+1;
        end
    end
end
% sags at bus 13
for k=1:length(sag_13_q3)
    for index=2:r
        if chart(index-1,1)<sag_13_q3(k)&&sag_13_q3(k)<chart(index,1)
            chart(index,3) = chart(index,3)+1;
        end
    end
end

%%
```