یا لطیف

دانشکده مهندسی برق

تمرین متلب پخش توان به روش نیوتون رافسون

تهیه کننده و نویسنده:

رضا آدینه پور

استاد مربوطه:

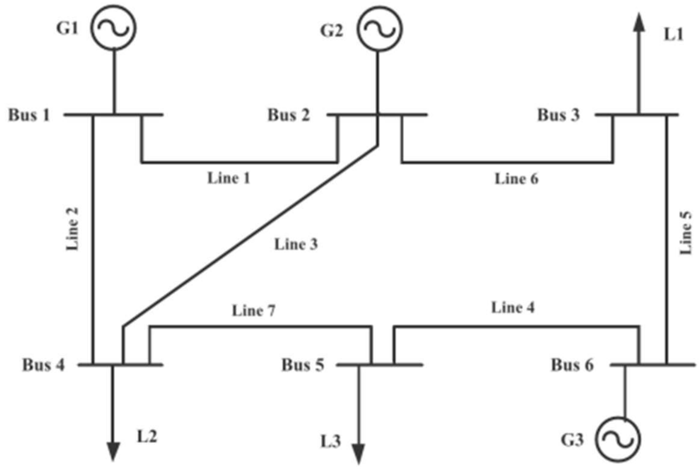
جناب اقای دکتر رحیمیان

تاریخ تهیه و ارائه:

دی ماه ۱۴۰۰

تمرین پخش بار (روش گوس-سایدل به کمک نرم‌افزار)

اطلاعات واحدهای تولیدی، شبکه انتقال و میزان مصرف در یک سیستم قدرت ۶ باس به ترتیب در جدول‌های (۱)، (۲) و (۳) داده شده است. ۳ واحد تولیدی در شبکه وجود دارد. ژنراتورهای ۱، ۲ و ۳ به ترتیب در باس‌های ۱، ۲ و ۶ قرار دارند. باس ۱ را از نوع مبنا فرض کنید. ولتاژ باس ۱ را 1 pu با زاویه صفر درجه در نظر بگیرید. اندازه ولتاژ باس ۲ را 1.05 pu فرض کنید. مقدار اولیه برای اندازه و زاویه ولتاژهای مجهول را به ترتیب 1 pu و 0 درجه در نظر بگیرید. مقدار خطای قابل قبول را برای محاسبه ولتاژها 0.0001 فرض کنید.



شکل (۱): شبکه ۶ باس

گزارش ارائه شده بایستی شامل موارد زیر باشد:

الف) نوع باس‌ها و متغیرهای حالت مجهول را تعیین و مسأله پخش بار را بر اساس روش گوس-سایدل فرمول‌بندی کنید.

ب) برنامه پخش‌بار مربوطه را در محیط نرم‌افزار MATLAB بنویسید و ضمیمه کنید. (کد نمونه آماده شده است که لازم است در بخش مرتبط با باس‌های کنترل ولتاژ توسط دانشجویان تکمیل شود.)

ج) اندازه و زاویه ولتاژ باس‌ها را از اولین تکرار تا آخرین تکرار در شکل‌های جداگانه ترسیم کنید. (دو نمودار برای ولتاژ هر باس)

د) ولتاژ باس‌ها (اندازه و زاویه)، میزان تولید اکتیو و راکتیو باس مبنا، میزان تولید راکتیو باس‌های کنترل ولتاژ P-|V| شبکه و تلفات توان اکتیو را تعیین کنید.

جدول (۱): اطلاعات واحدهای تولیدی

Generator	Bus	P^G(pu)	Q^G(pu)	$Q^{G,min}$(pu)	$Q^{G,max}$(pu)
1	1	نامعلوم	نامعلوم	-2	2
2	2	1.4	نامعلوم	-1	1.4
3	6	0.2	0.1	-0.1	0.2

ستون اول: شماره ژنراتور، ستون دوم: شماره باسی که ژنراتور در آن قرار دارد، ستون سوم: تولید اکتیو واحدها، ستون چهارم: تولید راکتیو واحدها، ستون پنجم و ششم: حداقل و حداکثر توان راکتیو قابل تولید توسط واحدها.

جدول (۲): اطلاعات شبکه انتقال

Line	From	To	Z (pu)	Capacity (pu)	Y/2 (pu)
1	1	2	0.1+j0.17	2	j0
2	1	4	0.15+j0.258	1	j0
3	2	4	0.12+j0.197	2	j0
4	5	6	0.08+j0.14	1	j0
5	3	6	0.01+j0.018	1	j0
6	2	3	0.02+j0.037	2	j0
7	4	5	0.02+j0.037	2	j0

جدول (۳): اطلاعات مصرف

Bus	P^D(pu)	Q^D(pu)
1	0.6	0.25
2	0.5	0.2
3	0.4	0.15
4	0.4	0.15
5	0.5	0.2
6	0.5	0.2

الف) ابتدا نوع باس ها و متغیر های مجهول را مشخص کنید؟

باس۱: مبنا (طبق صورت مسئله) **معلومات**: ولتاژ(اندازه و زاویه) **مجهولات**: توان اکتیو و راکتیو تزریقی

باس۲: PV، **معلومات**: اندازه ولتاژ و توان اکتیو تزریقی **مجهولات**: زاویه ولتاژ و توان راکتیو تزریقی

باس۳ تا ۶: PQ، **معلومات**: توان اکتیو و راکتیو تزریقی **مجهولات**: ولتاژ (اندازه و زاویه)

کد نوشته شده بر اساس کد همان توضیحی توسط اقای دکتر تکمیل شده است و فایل های آن ضمیمه فایل ارسالی شده است.

اطلاعات باس ها و خطوط به صورت زیر در فایل data.m تکمیل شده است.

Type:

1: slack bus

2: voltage controlled bus

3: consumption bus

```
% input data (line and bus)
% bus types: type-1=slack, type-2=PV, type-3=PQ

function [linedata,busdata,tolerence,npv,nlb,lpv] = data()
    %          |  From |  To    |   R      |    X     |   Y/2      |  Capacity    |
    %          |  Bus  | Bus    |          |          |            |              |
    linedata = [ 1       2        0.10       0.170      0.0          2.0;
                 1       4        0.15       0.258      0.0          1.0;
                 2       4        0.12       0.197      0.0          2.0;
                 5       6        0.08       0.140      0.0          1.0;
                 3       6        0.01       0.018      0.0          1.0;
                 2       3        0.02       0.037      0.0          2.0;
                 4       5        0.02       0.37       0.0          2.0;];


    %          |Bus | Type | Vsp | theta |  PGi    |  QGi    |  PLi    |   QLi   | Qmin | Qmax |
B_shunt  |
    busdata = [ 1      1     1.00      0       0.0       0.0       0.6       0.25     -2.0     2.0
0.0;
                2      2     1.05      0       0.0       0.0       0.5       0.20     -1.0     1.4
0.0;
                3      3     1.00      0       0.0       0.0       0.4       0.15      0.0     0.0
0.0;
                4      3     1.00      0       0.0       0.0       0.4       0.15      0.0     0.0
0.0;
                5      3     1.00      0       0.0       0.0       0.5       0.20      0.0     0.0
0.0;
                6      3     1.00      0       0.1       0.1       0.5       0.20     -0.1     0.2
0.0;];

    nlb=4; % number of LOAD-buses
    npv=1; % number of PV-buses
    lpv=2; % lowest value of PV-bus number

    tolerence=0.0001; % Defining the value of accepted tolerence
end
```

کد نوشته شده برای ۱۰ تکرار به صورت زیر است:

```
clear;clc;close all;

MVA_base=100.0; %Defining the Base-MVA
[linedata,busdata,tolerence,nPV,nLB,lPV] = data(); %Calling the function containing the input data

fb=linedata(:,1);  % Storing the from-bus numbers
tb=linedata(:,2);  % Storing the to-bus numbers
r=linedata(:,3);   % Storing the resistance(s)
x=linedata(:,4);   % Storing the reactance(s)
b=linedata(:,5);   % Storing the half-line charging(shunt-admittance)
a=linedata(:,6);   % Storing the off-nominal trans-ratio

bus_type=(busdata(:,2)); % Storing the bus-type(s)
v_bus_initial=(busdata(:,3)); % Storing the initial bus-voltage(s) in (p.u.)
P_gen=((busdata(:,5))*(MVA_base)); % Storing the active power generated at bus(s)
Q_gen=((busdata(:,6))*(MVA_base)); % Storing the reactive power generated at bus(s)
S_gen=((P_gen)+(1i)*(Q_gen)); % Storing the complex power generated at bus(s)
P_load=((busdata(:,7))*(MVA_base)); % Storing the active power at load(s) at bus(s)
Q_load=((busdata(:,8))*(MVA_base)); % Storing the reactive power at load(s) at bus(s)
S_load=((P_load)+(1i)*(Q_load)); % Storing the complex power at load(s) at bus(s)
Q_min=((busdata(:,9))*(MVA_base)); % Storing the minimum Q-range of the PV-bus(s)
Q_max=((busdata(:,10))*(MVA_base)); % Storing the maximum Q-range of the PV-bus(s)
B_shunt=((busdata(:,11))); % Storing the data corresponding to the static shunt capacitance
z_elementary=((r)+(1i)*(x)); % Calculating the 'z'
b_elementary=((1i)*(b)); % Calculating the 'b'
nbus = max(max(fb),max(tb));    % no. of buses
nbranch = length(fb);           % no. of branches
z=zeros(nbus,nbus); % Initialising the elementary [z]
b=zeros(nbus,nbus); % Initialising the elementary [b]
Ybus=zeros(nbus,nbus); % Initialising the elementary [Y] (Bus-admittance matrix)
tolerence_checker=1; % Initialising the 'tolerence_checker' as '1'
iteration=0; % Initialising the 'iteration' as '0'
V_new=v_bus_initial; % Initialising the 'V_new[]' as 'V_initial[]'
v_scheduled=v_bus_initial; % Initialising the 'v_scheduled[]' as 'V_initial[]' (to get the scheduled
voltages at the PV-bus(s))
V_new_accelerated=zeros(1,nbus); % Initialising the 'V_new_accelerated[]'
V_new_del=zeros(1,nbus); % Initialising the 'v_new_del[]' (to get the difference(s) of bus-voltage(s) of
two(02) consecutive iteration(s))
difference=zeros(1,nbus); % Iniialising the 'difference[]'
real_diff=zeros(1,nbus); % Initialising the 'real_diff[]' (to store the real part of the difference)
imag_diff=zeros(1,nbus); % Initialising the 'imag_diff[]' (to store the imaginary part of the difference)
delta=zeros(1,nbus); % Initialising the 'delta[]' (for getting the angle at PV-bus(s))
E_new=zeros(1,nbus); % Initialising the 'E_new[]' (to store the real part of the voltage at PV-bus)
F_new=zeros(1,nbus); % Initialising the 'F_new[]' (to store the imaginary part of the voltage at PV-bus)
Q_intermediate=zeros(1,nbus); % Initialising the 'Q_intermediate'
Q_final=zeros(1,nbus); % Initialising 'Q_final[]'
complex_flow_line=zeros(1,nbranch); % Initialising 'complex_flow_line[]'
line_flows=zeros(nbus,nbus); % Initialising 'line_flows[]'
active_flow_line=zeros(1,nbranch); % Initialising 'active_flow_line[]'
reactive_flow_line=zeros(1,nbranch); % Initialising 'reactive_flow_line[]'
bus_power_injection=zeros(1,nbus); % Initialising 'bus_power_injection[]'
bus_power_mismatch=zeros(1,nbus); % Initialising 'bus_power_mismatch[]'
line_loss=zeros(nbus,nbus); % Initialising 'line_loss[]'
flow_count=1; % Initialising the 'flow_count' as (1) //(for tracking the total no. of lines)
sum_line_loss=((0.0)+(1i)*(0.0)); % Initialising 'sum_line_loss' as '0'
sum=zeros(1,nbus); % Initialising the 'sum' (used in calculating the digaonal [Ybus] elements)
shunt_fb_onr=zeros(1,nbranch); % storing the ((a^2)/(1-a)) as seen from the FB(From Bus)
shunt_tb_onr=zeros(1,nbranch); % storing the ((a)/(a-1)) as seen from the TB(To Bus)
P_injected_sum=zeros(nbus,1); % Initialising 'P_injected_sum[]' for intermediate calculation(s) of injected
(P)
Q_injected_sum=zeros(nbus,1); % Initialising 'Q_injected_sum[]' for intermediate calculation(s) of injected
(Q)
P_injected_bus=zeros(nbus,1); % for storing the intermediate 'active power injection(s)' at different
bus(s)
Q_injected_bus=zeros(nbus,1); % for storing the intermediate 'reactive power injection(s)' at different
bus(s)
partial_P_delta=zeros(nbus-1,nbus-1); % Initialsing 'partial_P_delta[]'corresponding to the [J11]
partial_Q_delta=zeros(nbus-nPV-1,nbus-1); % Initialsing 'partial_Q_delta[]'corresponding to the [J21]
partial_P_vol_mag=zeros(nbus-1,nbus-nPV-1); % Initialising 'partial_P_vol_mag[]'corresponding to the [J12]
partial_Q_vol_mag=zeros(nbus-nPV-1,nbus-nPV-1); % Initialising 'partial_Q_vol_mag[]'corresponding to the
[J12]
J=zeros(((2*(nbus-1))-nPV),((2*(nbus-1))-nPV)); % Initialising 'J[]' i.e. THE COMPLETE JACOBIAN MATRIX
inj_active_pow_mismatch_vector=zeros(nbus-1,1); % Initialising the 'inj_active_pow_mismatch_vector[]'
inj_reactive_pow_mismatch_vector=zeros(nbus-nPV-1,1); % Initialising the
'inj_reactive_pow_mismatch_vector[]'
inj_pow_mismatch_vector=zeros(((2*(nbus-1))-nPV),1); % Initialising the 'inj_pow_mismatch_vector[]'
correction_vector=zeros(((2*(nbus-1))-nPV),1); % Initialising the 'correction_vector[]'
```

```
correction_voltage_angle=zeros((nbus-1),1); % Initialising the 'correction_voltage_angle[]'
correction_voltage_magnitude=zeros((nbus-1-nPV),1); % Initialising the 'correction_voltage_magnitude[]'
mismatch_active_vector_element_count=1; % Initialising the 'mismatch_active_vector_element_count' as (1)
mismatch_reactive_vector_element_count=1; % Initialising the 'mismatch_reactive_vector_element_count' as
(1)
mismatch_vector_element_count=0;% Initialising the 'mismatch_active_vector_element_count' as (0)
real_mismatch=zeros(nbus-1,1);
imag_mismatch=zeros(nbus-1-nPV,1);

for u=1:((2*(nbus-1))-nPV)
    correction_vector(u)=1.0; % Initialising the correction vector element(s) as (1.0); so that loop can be
started
end

% Forming the elementary z-matrix & b-matrix
for u=1:nbranch
    z(fb(u),tb(u))=z_elementary(u); % Storing the mutual element(s) (Z(i,j))
    z(tb(u),fb(u))=z(fb(u),tb(u)); % Storing the similar element(s) in (Z(j,i))
    b(fb(u),tb(u))=b_elementary(u); % Storing the mutual element(s) (B(i,j))
    b(tb(u),fb(u))=b(fb(u),tb(u)); % Storing the similar element(s) in (B(j,i))
end

% Modification(s) in the element(s) of 'z' & 'b' due to OFF-NOMINAL TRANS RATIO
for u=1:nbranch
    if((a(u,1))~=1.0)
        shunt_fb_onr(1,u)=(((a(u,1))^(2))/(1-(a(u,1)))); % Calculating the (a^2/(1-a))
        b(fb(u),tb(u))=(1/((shunt_fb_onr(1,u))*(z(fb(u),tb(u)))))+(b(fb(u),tb(u))); % Modifying the element
'b' matrix w.r.t. 'fb'
        shunt_tb_onr(1,u)=((a(u,1))/((a(u,1))-1)); % Calculating the (a/(a-1))
        b(tb(u),fb(u))=(1/((shunt_tb_onr(1,u))*(z(fb(u),tb(u))))); % Modifying the element 'b' matrix
w.r.t. 'tb'
        z(fb(u),tb(u))=z(fb(u),tb(u))*(a(u,1)); % Modifying the 'z' matrix
        z(tb(u),fb(u))=z(fb(u),tb(u)); % Storing the similar element(s) in (Z(j,i))
    end
end

% Including the effect(s) of static shunt-capacitance
for u=1:nbus
    if(B_shunt(u)~=0.0)
        sum(1,u)=B_shunt(u); % Updating the 'sum(1,u)' with the data for static shunt capacitance
    end
end

%%  ------------------------- STEP-1 - CALCULATION OF BUS-ADMITTANCE MATRIX -------------------------
for u=1:nbus
    for j=1:nbus
      if(z(u,j)==0.0)
          Ybus(u,j)=0.0; % No connection between (u) & (j)
      else
          Ybus(u,j)=-(1/z(u,j)); % Calculating the mutual-admittances
      end
    end
end

for u=1:nbus
    for j=1:nbus
      sum(1,u)=sum(1,u)+b(u,j)-Ybus(u,j);
      if(j==nbus)
          Ybus(u,u)=sum(1,u); % Calculating the self-admittances
      end
    end
end

fprintf('THE BUS-ADMITTANCE IS GIVEN BELOW\n'); %DISPLAYING THE RESULTS GOT FROM STEP-1
disp(Ybus);
Gbus=real(Ybus); % Generating the conducatance matrix[G] by taking the real part of the [Ybus] element(s);
(y=g+jb)
Bbus=imag((+1)*(Ybus)); % Generating the susceptance matrix[B] by taking the imaginary part of the [Ybus]
element(s); (y=g+jb)

% ----------------------------------------------END OF STEP-1------------------------------------------
%%
while((tolerence_checker>0)&&(iteration<10))
    mismatch_vector_element_count=0;% Resetting the 'mismatch_active_vector_element_count' as (0)
    mismatch_active_vector_element_count=1; % Resetting the 'mismatch_active_vector_element_count' as (1)
    mismatch_reactive_vector_element_count=1; % Resetting the 'mismatch_reactive_vector_element_count' as
(1)
    P_injected_sum=zeros(nbus,1); % Resetting 'P_injected_sum[]' for intermediate calculation(s) of
injected (P)
    Q_injected_sum=zeros(nbus,1); % Resetting 'Q_injected_sum[]' for intermediate calculation(s) of
injected (Q)
    tolerence_checker=0; % resetting the 'tolerence_checker' to (0)

    %% --------------------------------------------STARTING OF STEP-2-----------------------------------------
--
    % CALCULATING THE ACTIVE POWER INJECTION(s) AND REACTIVE POWER INJECTION(s) AT DIFFEENT BUS(s) AND
    % THE MISMATCH VECTOR FOR THE ACTIVE POWER INJECTION(s) & REACTIVE POWER INJECTION(s)
```

```
    % (ACTIVE POWER INJECTION MISMATCH=(SCHEDULED ACTIVE POWER INJECTION-CALCULATED ACTIVE POWER
INJECTION))
    % (REACTIVE POWER INJECTION MISMATCH=(SCHEDULED REACTIVE POWER INJECTION-CALCULATED REACTIVE POWER
INJECTION))
    %-----------------------------------------------------------------------------------------------------
--
    P_injected_scheduled=((P_gen-P_load)/(MVA_base)); % calculating the scheduled injected active power at
each bus
    Q_injected_scheduled=((Q_gen-Q_load)/(MVA_base)); % calculating the scheduled injected reactive power
at each bus

    for u=1:nbus
        for j=1:nbus
            if((u~=j)&&(abs(Ybus(u,j)))~=0.0)

P_injected_sum(u,1)=((P_injected_sum(u,1))+(((abs((V_new(u,1))*(V_new(j,1))*(Ybus(u,j)))))*(cos((angle(Ybus(
u,j)))+(angle(V_new(j,1)))-(angle(V_new(u,1))))))))); % calculating the mutual sumation part
            end
            if((u~=j)&&((abs(Ybus(u,j)))~=0.0)&&(bus_type(u)~=2))

Q_injected_sum(u,1)=((Q_injected_sum(u,1))+(((abs((V_new(u,1))*(V_new(j,1))*(Ybus(u,j)))))*(sin((angle(Ybus(
u,j)))+(angle(V_new(j,1)))-(angle(V_new(u,1))))))))); % calculating the mutual sumation part
            end
            if(j==nbus)
                P_injected_bus(u,1)=((((abs(V_new(u,1)))^(2))*(Gbus(u,u)))+(P_injected_sum(u,1))); %
calculating the injecetd 'P' at bus-(u)
            end
            if((j==nbus)&&(bus_type(u)~=2))
                Q_injected_bus(u,1)=(-((((abs(V_new(u,1)))^(2))*(Bbus(u,u)))+(Q_injected_sum(u,1)))); %
calculating the injecetd 'Q' at bus-(u)
            end
        end
    end

    for u=1:nbus
        if(bus_type(u)~=1)

inj_active_pow_mismatch_vector(mismatch_active_vector_element_count,1)=((P_injected_scheduled(u,1))-
(P_injected_bus(u,1))); % calculating the mismatch vector for active power injecttion
            mismatch_active_vector_element_count=mismatch_active_vector_element_count+1; % updating the
value of 'mismatch_vector_element_count'
        end
        if((bus_type(u)~=1)&&(bus_type(u)~=2))

inj_reactive_pow_mismatch_vector(mismatch_reactive_vector_element_count,1)=((Q_injected_scheduled(u,1))-
(Q_injected_bus(u,1))); % calculating the mismatch vector for reactive power injecttion
            mismatch_reactive_vector_element_count=mismatch_reactive_vector_element_count+1; % updating the
value of 'mismatch_vector_element_count'
        end
    end

    for u=2:nbus
        if(mismatch_vector_element_count<=((mismatch_active_vector_element_count-
1)+(mismatch_reactive_vector_element_count-1)))
            inj_pow_mismatch_vector((u-1),1)=inj_active_pow_mismatch_vector(u-1,1); % storing the mismatch
in the active power
            mismatch_vector_element_count=mismatch_vector_element_count+1; % updating the
'mismatch_vector_element_count'
        end
        if((mismatch_vector_element_count<((mismatch_active_vector_element_count-
1)+(mismatch_reactive_vector_element_count-1)))&&((u-1)<=(nbus-1-nPV)))
            inj_pow_mismatch_vector((u+nbus-2),1)=inj_reactive_pow_mismatch_vector(u-1,1); % storing the
mismatch in the reactive power
            mismatch_vector_element_count=mismatch_vector_element_count+1; % updating the
'mismatch_vector_element_count'
        end
    end
    fprintf('\nThe mismatch-vector is :\n');
    disp(inj_pow_mismatch_vector); % displaying the 'inj_pow_mismatch_vector[]'

    for u=1:nbus-1
        real_mismatch(u,1)=inj_pow_mismatch_vector(u,1); % storing the real-power mismatch(s)
        if((u+nbus-1)<=((2*(nbus-1))-nPV))
            imag_mismatch(u,1)=inj_pow_mismatch_vector(u+nbus-1,1); % storing the imaginary-power
mismatch(s)
        end
    end

    for u=1:((2*(nbus-1))-nPV)
        if((abs(inj_pow_mismatch_vector(u,1)))>(tolerence))
            tolerence_checker=tolerence_checker+1;
        end
    end
    %----------------------------------------------END OF STEP-2------------------------------------------

    %% ---------------------------------------------STARTING OF STEP-3--------------------------------------
--
    % STEP-3: CALCULATES THE [J] OR THE JACOBIAN MATRIX ([J]=[[J11] [J12] : [J21] [J22]])
```

```
    % UNDER THIS STEP SUB-STEP HAS BEEN PRFORMED FOR CALCULATING THE [J11], [J12], [J21], [J22] IN
INDIVIDUAL
    %-----------------------------------------------------------------------------------------------
--
    % STEP-(3.a): CALCULATING THE [J11]
    %-----------------------------------------------------------------------------------------------
--
    sum_pv=0; % initialising 'sum_pv' for getting the [J] elements dependent on PV-bus (Q)
    for u=1:nbus-1
        for j=1:nbus-1
            if((u+1)~=(j+1))
                partial_P_delta(u,j)=(-
((abs((V_new(u+1,1))*(V_new(j+1,1))*(Ybus(u+1,j+1))))*(sin((angle(Ybus(u+1,j+1)))+(angle(V_new(j+1,1)))-
(angle(V_new(u+1,1))))))); % calculating the off-diagonal element(s)
            else
                if((bus_type(u+1))~=2)
                    partial_P_delta(u,j)=(-
((((abs(V_new(u+1,1)))^(2))*(Bbus(u+1,u+1)))+(Q_injected_bus(u+1,1)))); % calculating the diagonal
element(s)
                else
                    for k=1:nbus
                        if(((u+1)~=(k))&&(abs(Ybus(u+1,k))~=0.0))
                            %sum_pv=sum_pv-partial_P_delta(u-1,k-1);

sum_pv=sum_pv+((abs((V_new(u+1,1))*(V_new(k,1))*(Ybus(u+1,k)))*(sin((angle(Ybus(u+1,k))+(angle(V_new(k,1)))
-(angle(V_new(u+1,1))))))));
                        end
                    end
                    partial_P_delta(u,j)=sum_pv; % calculating the (partial_P_delta(3,3)) for the PV-bus
                    sum_pv=0; % resetting the 'sum_pv' to (0)
                end
            end
        end
    end
    fprintf('\n The [J11] is : \n');
    disp(partial_P_delta); % diplaying the [J11]
    %-----------------------------------------------------------------------------------------------
--
    % STEP-(3.b): CALCULATING THE [J21]
    %-----------------------------------------------------------------------------------------------
--
    m=1;n=1;element_count=0;
    for u=2:nbus
        for j=2:nbus
            if((m~=n)&&((bus_type(u))~=2)&&(u~=j))
                partial_Q_delta(m,n)=(-
((abs((V_new(u,1))*(V_new(j,1))*(Ybus(u,j))))*(cos((angle(Ybus(u,j)))+(angle(V_new(j,1)))-
(angle(V_new(u,1))))))); % calculating the off-diagonal element(s)
                n=n+1;
                element_count=element_count+1;
            elseif((m~=n)&&((bus_type(u))~=2)&&(u==j))
                partial_Q_delta(m,n)=((P_injected_bus(u,1))-(((abs(V_new(u,1)))^(2))*(Gbus(u,u))));
                n=n+1;
                element_count=element_count+1;
            elseif((m==n)&&((bus_type(u))~=2)&&(u~=j))
                partial_Q_delta(m,n)=(-
((abs((V_new(u,1))*(V_new(j,1))*(Ybus(u,j))))*(cos((angle(Ybus(u,j)))+(angle(V_new(j,1)))-
(angle(V_new(u,1))))))); % calculating the off-diagonal element(s)
                n=n+1;
                element_count=element_count+1;
            elseif((m==n)&&((bus_type(u))~=2)&&(u==j))
                partial_Q_delta(m,n)=((P_injected_bus(u,1))-(((abs(V_new(u,1)))^(2))*(Gbus(u,u))));
                n=n+1;
                element_count=element_count+1;
            end
        end
        if((j==nbus)&&(element_count>0))
            m=m+1;n=1;element_count=0;
        end
    end
    fprintf('\n The [J21] is : \n');
    disp(partial_Q_delta); % diplaying the [J21]
    %-----------------------------------------------------------------------------------------------
--
    % STEP-(3.c): CALCULATING THE [J12]
    %-----------------------------------------------------------------------------------------------
--
    m=1;n=1;element_count=0;
    for u=2:nbus
        for j=2:nbus
            if((m==n)&&(bus_type(j)~=2)&&(u~=j)&&(bus_type(u)~=2))
                %partial_P_vol_mag(m,n)=(-(partial_Q_delta(u-nPV-1,j-1))); % calculating the off-diagonal
element(s)

partial_P_vol_mag(m,n)=((abs((V_new(u,1))*(V_new(j,1))*(Ybus(u,j))))*(cos((angle(Ybus(u,j)))+(angle(V_new(j
,1)))-(angle(V_new(u,1))))));
                n=n+1;
                element_count=element_count+1;
```

```
            elseif((m==n)&&(bus_type(j)~=2)&&(u==j)&&(bus_type(u)~=2))
                partial_P_vol_mag(m,n)=((P_injected_bus(u,1))+(((abs(V_new(u,1)))^(2))*(Gbus(u,u))));
                n=n+1;
                element_count=element_count+1;
            elseif((m==n)&&(bus_type(j)~=2)&&(u~=j)&&(bus_type(u)==2))

partial_P_vol_mag(m,n)=((abs((V_new(u,1))*(V_new(j,1))*(Ybus(u,j))))*(cos((angle(Ybus(u,j)))+(angle(V_new(j
,1)))-(angle(V_new(u,1))))));
                n=n+1;
                element_count=element_count+1;
            elseif((m~=n)&&(bus_type(j)~=2)&&(u~=j)&&(bus_type(u)~=2))

partial_P_vol_mag(m,n)=((abs((V_new(u,1))*(V_new(j,1))*(Ybus(u,j))))*(cos((angle(Ybus(u,j)))+(angle(V_new(j
,1)))-(angle(V_new(u,1))))));
                n=n+1;
                element_count=element_count+1;
            elseif((m~=n)&&(bus_type(j)~=2)&&(u==j)&&(bus_type(u)~=2))
                partial_P_vol_mag(m,n)=((P_injected_bus(u,1))+(((abs(V_new(u,1)))^(2))*(Gbus(u,u))));
                n=n+1;
                element_count=element_count+1;
            elseif((m~=n)&&(bus_type(j)~=2)&&(u~=j)&&(bus_type(u)==2))

partial_P_vol_mag(m,n)=((abs((V_new(u,1))*(V_new(j,1))*(Ybus(u,j))))*(cos((angle(Ybus(u,j)))+(angle(V_new(j
,1)))-(angle(V_new(u,1))))));
                n=n+1;
                element_count=element_count+1;
            end
        end
        if((j==nbus)&&(element_count>0))
            m=m+1;n=1;element_count=0;
        end
    end
    fprintf('\n The [J12] is : \n');
    disp(partial_P_vol_mag); % diplaying the [J12]
    %---------------------------------------------------------------------------------------------------
--
    % STEP-(3.d): CALCULATING THE [J22]
    %---------------------------------------------------------------------------------------------------
--
    m=1;n=1;element_count=0;
    for u=2:nbus
        for j=2:nbus
            if((m~=n)&&(u~=j)&&((bus_type(u))~=2)&&((bus_type(j))~=2))
                partial_Q_vol_mag(m,n)=(-
((abs((V_new(u,1))*(V_new(j,1))*(Ybus(u,j))))*(sin((angle(Ybus(u,j)))+(angle(V_new(j,1)))-
(angle(V_new(u,1)))))));
                n=n+1;
                element_count=element_count+1;
            elseif((m~=n)&&(u==j)&&((bus_type(u))~=2)&&((bus_type(j))~=2))
                partial_Q_vol_mag(m,n)=((Q_injected_bus(u,1))-(((abs(V_new(u,1)))^(2))*(Bbus(u,u))));
                n=n+1;
                element_count=element_count+1;
            elseif((m==n)&&(u~=j)&&((bus_type(u))~=2)&&((bus_type(j))~=2))
                %partial_Q_vol_mag(m,n)=partial_P_delta(u-nPV-1,j-nPV-1);
                partial_Q_vol_mag(m,n)=(-
((abs((V_new(u,1))*(V_new(j,1))*(Ybus(u,j))))*(sin((angle(Ybus(u,j)))+(angle(V_new(j,1)))-
(angle(V_new(u,1)))))));
                n=n+1;
                element_count=element_count+1;
            elseif((m==n)&&(u==j)&&((bus_type(u))~=2)&&((bus_type(j))~=2))
                partial_Q_vol_mag(m,n)=((Q_injected_bus(u,1))-(((abs(V_new(u,1)))^(2))*(Bbus(u,u))));
                n=n+1;
                element_count=element_count+1;
            end
        end
        if((j==nbus)&&(element_count>0))
            m=m+1;n=1;element_count=0;
        end
    end
    fprintf('\n The [J22] is : \n');
    disp(partial_Q_vol_mag); % diplaying the [J22]
    %---------------------------------------------------------------------------------------------------
--
    % STEP-(3.e): CALCULATING THE COMPLETE [J]
    %---------------------------------------------------------------------------------------------------
--
    J=[partial_P_delta partial_P_vol_mag ; partial_Q_delta partial_Q_vol_mag];
    fprintf('\nThe complete JACOBIAN matrix is : \n');
    disp(J);
    %----------------------------------------------END OF STEP-3------------------------------------------

    %% -------------------------------------------STARTING OF STEP-4--------------------------------------
--
    % STEP-4: CALCULATING THE CORRECTION VECTOR BY ((inv(J))*(MISMATCH-VECTOR)) & THE UPDATED VALUES OF THE
    % STATE VARIABLES i.e. (|v_i|,DELTA_i)
    %---------------------------------------------------------------------------------------------------
--
    for u=1:(2*(nbus-1))
```

```
        correction_vector(u,1)=0.0; % Initialising the 'correction_vector' as (0.0)
    end
    correction_vector=(inv(J))*(inj_pow_mismatch_vector); % calculating the 'correction_vector[]'
    fprintf('\n Inverse of the [J] matrix : \n');
    disp(inv(J)); % displaying the inverse of the [J] matrix
    for u=1:(nbus-1)
        correction_voltage_angle(u,1)=correction_vector(u,1); % storing the correction(s) in the voltage
angle(s)
        if(u<=(nbus-1-nPV))
            correction_voltage_magnitude(u,1)=correction_vector((u+nbus-1),1); % storing the correction(s)
in the voltage angle(s)
        end
    end
    correction_voltage_magnitude_count=1; % initialising the 'correction_voltage_magnitude_count' as (1)
    for u=1:(nbus-1)
        voltage_angle_updated=angle((V_new(u+1,1)))+(correction_voltage_angle(u,1));  % calculating the
updated voltage-angle
        if(bus_type(u+1)~=2)

voltage_magnitude_updated=abs((V_new(u+1,1)))+((correction_voltage_magnitude(correction_voltage_magnitude_c
ount,1))*(abs((V_new(u+1,1))))); % calculating the updated voltage-magnitude
            correction_voltage_magnitude_count=correction_voltage_magnitude_count+1; % updating
'correction_voltage_magnitude_count' by (1)
        end
        if(bus_type(u+1)~=2)

V_new(u+1,1)=voltage_magnitude_updated*((cos(voltage_angle_updated))+(sin(voltage_angle_updated)*(1i))); %
calculating the updated voltage
        end
        if(bus_type(u+1)==2)

V_new(u+1,1)=abs(V_new(u+1,1))*((cos(voltage_angle_updated))+(sin(voltage_angle_updated)*(1i))); %
calculating the updated voltage
        end
    end
    %-----------------------------------------------------------------------------------------------------
--
    iteration=iteration+1; % updating the value of iteration-count by (1)
    fprintf('\n The updated state variables after the (%d)-th iteration is :\n',iteration);
    fprintf('    BUS        RECTANGULAR-FORM                   POLAR-FORM\n');
    fprintf('-----------------------------------------------------------------------\n');
    for u=1:nbus
        fprintf('    %d      (%f)+j(%f)      (%f)at an
angle(%f)deg.\n',u,real(V_new(u,:)),imag(V_new(u,:)),abs(V_new(u,:)),((180/pi)*(angle(V_new(u,:)))));

%         figure(u);
%         subplot(2,1,1);
%         plot(abs(V_new(u,:)), '-o');
%         xlabel('Iteration');
%         ylabel('Voltage amplitude (p.u.)');
%
%         subplot(2,1,2);
%         plot(((180/pi)*(angle(V_new(u,:)))), '-o r');
%         xlabel('Iteration');
%         ylabel('Voltage angle (degree)');
    end
end

%% STEP-4 - CALCULATING THE LINE-FLOW(s) & THE BUS-INJECTION(s) & THE BUS-POWER  MISMATCH(s) & THE LINE-
LOSS(s)
%-----------------------------------------------------------------------------------------------------------
--
fprintf('The Line-flows are as follows : \n');
fprintf('Line-code      Active-flow(p.u.)      Reactive-flow(p.u.)\n');
fprintf('-------------------------------------------------------------\n');
for u=1:nbus
   for j=1:nbus
       if((u~=j)&&(Ybus(u,j))~=0.0)
           complex_flow_line(1,flow_count)=((conj(V_new(u,1))*((V_new(u,1))-(V_new(j,1))))*(-
Ybus(u,j)))+(conj(V_new(u,1))*(V_new(u,1))*(b(u,j))); % Calculating the line-flows
           line_flows(u,j)=conj(complex_flow_line(1,flow_count)); % Storing the power flow in the line-flow
matrix
           active_flow_line(1,flow_count)=real(complex_flow_line(1,flow_count)); % P.U. active power flow
from (u->j)
           reactive_flow_line(1,flow_count)=(-(imag(complex_flow_line(1,flow_count)))); % P.U. reactive
power flow from (u->j)
           fprintf('  %d->%d',u,j); % Printing the line-code
           fprintf('            %f
%f\n',active_flow_line(1,flow_count),reactive_flow_line(1,flow_count)); % Printing the active & reactive
power flows
           bus_power_injection(1,u)=bus_power_injection(1,u)+conj(complex_flow_line(1,flow_count)); %
Calculating the bus-power-injection(s)
           bus_power_mismatch(1,u)=((S_gen(u,1)-S_load(u,1))/MVA_base)-(bus_power_injection(1,u)); %
Calculating the bus-power-mismach
           flow_count=flow_count+1; % Updating the flow_count by (1)
       end
   end
end
```

```
fprintf('The bus-power injection(s) are :\n');
fprintf('Bus-code        Power-injection(p.u.)\n');
fprintf('----------------------------------------\n');
for u=1:nbus
    fprintf('  %d             (%f)+j(%f)\n',u,real(bus_power_injection(u)),imag(bus_power_injection(u))); %
Displaying the bus-power injections
end
fprintf('The bus-power mismatch(s) are :\n');
fprintf('Bus-code        Bus-power-mismatch(p.u.)\n');
fprintf('----------------------------------------\n');
for u=2:nbus
    fprintf('  %d             (%f)+j(%f)\n',u,real(bus_power_mismatch(u)),imag(bus_power_mismatch(u))); %
Displaying the bus-power injections
end
for u=1:nbus
    for j=1:nbus
        if((u<j)&&(abs(line_flows(u,j))~=0.0))
            line_loss(u,j)=line_flows(u,j)+line_flows(j,u); % Calculating the individual line-loss
            real_line_loss=abs(real(line_loss(u,j))); % Absolute value of the active-line-loss
            imag_line_loss=abs(imag(line_loss(u,j))); % Absolute value of the reactive-line-loss
            line_loss(u,j)=(real_line_loss)+(imag_line_loss)*(1i);
            sum_line_loss=sum_line_loss+line_loss(u,j); % Calculating the total line-loss
        end
    end
end
fprintf('Individual line-loss is as follows = \n');
fprintf('Line-code            Line-loss  \n');
fprintf('-----------------------------------\n');
for u=1:nbus
    for j=1:nbus
        if((u<j)&&(abs(line_flows(u,j))~=0.0))
            fprintf('  %d->%d      (%f)+j(%f)\n',u,j,real(line_loss(u,j)),imag(line_loss(u,j)));
        end
    end
end
fprintf('Summation of all the losses = (%f)+j(%f)\n',real(sum_line_loss),imag(sum_line_loss)); % Displaying
the summation of all the line-losses
```

خروجی کد به صورت زیر است:

Command Window

```
The complete JACOBIAN matrix is :
   1.0e+03 *

    0.0393   -0.0226   -0.0159         0         0    0.0010   -0.0005         0         0
   -0.0115   -0.1849         0         0    0.1964    0.3189         0         0    0.1099
   -0.0077         0    0.0757   -0.0511         0         0    0.3328   -0.0597         0
         0         0   -0.0572   -0.0686    0.1258         0    0.0538    0.4588   -0.0402
         0    0.1971         0    0.0292   -0.2263    0.1087         0    0.1288    0.5874
    0.0194    0.0905         0         0   -0.1099    0.5997         0         0    0.1964
    0.0139         0   -0.0574    0.0597         0         0    0.6380   -0.0511         0
         0         0   -0.0538    0.0136    0.0402         0   -0.0572    1.1848    0.1258
         0   -0.1087         0   -0.1288    0.2375    0.1971         0    0.0292    0.8542


 Inverse of the [J] matrix :
    0.5470    0.7320    0.2306    0.6549    0.6990   -0.3823   -0.2035   -0.3167   -0.4095
    0.5503    0.7737    0.2406    0.6967    0.7404   -0.4041   -0.2141   -0.3367   -0.4334
    0.5065    0.7090    0.2277    0.6279    0.6747   -0.3701   -0.1986   -0.3038   -0.3959
    0.5520    0.7790    0.2386    0.6887    0.7409   -0.4066   -0.2121   -0.3337   -0.4347
    0.5500    0.7740    0.2398    0.6931    0.7368   -0.4043   -0.2132   -0.3350   -0.4314
   -0.0001    0.0012    0.0002    0.0008    0.0003    0.0011   -0.0002   -0.0003   -0.0005
   -0.0183   -0.0255   -0.0070   -0.0226   -0.0243    0.0133    0.0081    0.0111    0.0142
   -0.0029   -0.0043   -0.0009   -0.0040   -0.0042    0.0023    0.0010    0.0028    0.0023
    0.0005    0.0006   -0.0001   -0.0003    0.0012   -0.0007    0.0001   -0.0000    0.0005
```

Command Window

```
 The updated state variables after the (10)-th iteration is :
    BUS        RECTANGULAR-FORM                   POLAR-FORM
-----------------------------------------------------------------------
    1      (1.000000)+j(0.000000)      (1.000000)at an angle(0.000000)deg.
    2      (0.146985)+j(-1.039661)      (1.050000)at an angle(-81.953008)deg.
    3      (-0.304701)+j(-0.881564)      (0.932737)at an angle(-109.067055)deg.
    4      (-0.419060)+j(-4.301418)      (4.321783)at an angle(-95.564403)deg.
    5      (2.237199)+j(-3.640186)      (4.272706)at an angle(-58.425789)deg.
    6      (0.390057)+j(1.215563)      (1.276612)at an angle(72.209306)deg.
The Line-flows are as follows :
Line-code      Active-flow(p.u.)      Reactive-flow(p.u.)
-----------------------------------------------------------
  1->2           2.725501              -0.564955
  1->4           14.850275              -3.133649
  2->1           0.373511              5.833275
  2->3           2.856297              -5.874915
  2->4           -2.374787              -8.348284
  3->2           0.240161              11.603360
  3->6           49.721427              86.846207
  4->1           19.702288              62.564056
  4->2           35.173025              62.192058
  4->5           -15.413720              -6.439173
  5->4           16.608917              28.550312
  5->6           44.813592              130.165479
  6->3           65.387789              120.350382
  6->5           38.233274              15.166537
```

Command Window

```
  5->6           44.813592              130.165479
  6->3           65.387789              120.350382
  6->5           38.233274              15.166537
The bus-power injection(s) are :
Bus-code        Power-injection(p.u.)
---------------------------------------
  1             (17.575776)+j(-3.698604)
  2             (0.855021)+j(-8.389924)
  3             (49.961588)+j(98.449567)
  4             (39.461593)+j(118.316942)
  5             (61.422508)+j(158.715791)
  6             (103.621064)+j(135.516919)
The bus-power mismatch(s) are :
Bus-code        Bus-power-mismatch(p.u.)
---------------------------------------
  2             (0.044979)+j(8.189924)
  3             (-50.361588)+j(-98.599567)
  4             (-39.861593)+j(-118.466942)
  5             (-61.922508)+j(-158.915791)
  6             (-103.921064)+j(-135.616919)
Individual line-loss is as follows =
Line-code             Line-loss
----------------------------------
  1->2      (3.099012)+j(5.268320)
  1->4      (34.552563)+j(59.430408)
  2->3      (3.096457)+j(5.728446)
```

Command Window

```
  1             (17.575776)+j(-3.698604)
  2             (0.855021)+j(-8.389924)
  3             (49.961588)+j(98.449567)
  4             (39.461593)+j(118.316942)
  5             (61.422508)+j(158.715791)
  6             (103.621064)+j(135.516919)
The bus-power mismatch(s) are :
Bus-code        Bus-power-mismatch(p.u.)
----------------------------------------
  2             (0.044979)+j(8.189924)
  3             (-50.361588)+j(-98.599567)
  4             (-39.861593)+j(-118.466942)
  5             (-61.922508)+j(-158.915791)
  6             (-103.921064)+j(-135.616919)
Individual line-loss is as follows =
Line-code            Line-loss
-----------------------------------
  1->2      (3.099012)+j(5.268320)
  1->4      (34.552563)+j(59.430408)
  2->3      (3.096457)+j(5.728446)
  2->4      (32.798238)+j(53.843775)
  3->6      (115.109216)+j(207.196589)
  4->5      (1.195197)+j(22.111139)
  5->6      (83.046866)+j(145.332016)
Summation of all the losses = (272.897549)+j(498.910692)
>>
```