PQ法潮流计算

# 初始化数据

clc,clear

base\_MVA = 100.0; %基准容量

error\_tol = 0.00001; %误差精度

circle\_count = 0; %循环次数

% Bus

% 1.节点号bus number 2.节点类型type 3.发电机功率 4.负荷功率

% 5.电压初始值 6.PV节点电压给定值

% 节点类型标号：1为PQ节点、2为PV节点，3为平衡节点

% 发电机、负荷功率为有名值、电压初始值为标幺值

% Branch

% 1.支路首端编号 2.支路末端编号 3.支路阻抗

% 两个矩阵都已经放入到对应的mat文件里了

% 接线1 对应plan3.PWB

load plan3.mat

% %接线2 对应plan4.PWB

% load plan4.mat

%下面数据可以同时在PQ和NR法成功收敛

% load success\_data.mat

slack\_bus\_voltage = 1.05; %平衡节点电压幅值

slack\_bus\_ang = 0; %平衡节点电压相角

%节点数

node\_num = size(Bus,1);

% 电压最终值直角坐标系表达式

u = zeros(node\_num,1);

% 电压最终值极坐标系表达式

u\_amp = zeros(node\_num,1);

u\_ang = zeros(node\_num,1);

% 节点功率最终值

S= [];

% 节点功率

S\_node = zeros(node\_num,1);

% 节点电流

I = zeros(node\_num,node\_num);

# 求节点导纳矩阵

Y = zeros(node\_num,node\_num);

for i = 1:size(Branch,1)

p = Branch(i,1);

q = Branch(i,2);

node\_data\_y = 1 ./ Branch(i,3);

% 修改节点自导纳

Y(p,p) = Y(p,p) + node\_data\_y;

% 修改连接节点自导纳

Y(q,q) = Y(q,q) + node\_data\_y;

% 修改互导纳

Y(p,q) = Y(p,q) - node\_data\_y;

Y(q,p) = Y(p,q);

end

%分解出导纳矩阵的实部和虚部

G = real(Y);

B = imag(Y);

# 读取数据形成节点电压以及电压初始状态

PQ\_num = sum(Bus(:,2) == 1); % 统计PQ节点数

PV\_num = sum(Bus(:,2) == 2); % 统计PV节点数

B1 = B(1:(end-1),1:(end-1)); % B'

B2 = B(1:PQ\_num,1:PQ\_num); % B''

dP = zeros(PQ\_num+PV\_num,1);

P = zeros(PQ\_num+PV\_num,1);

P\_set = zeros(PQ\_num+PV\_num,1);

dtheta = zeros(PQ\_num+PV\_num,1); % 角度和P有关

dQ = zeros(PQ\_num,1);

Q = zeros(PQ\_num,1);

Q\_set = zeros(PQ\_num,1);

dU = zeros(PQ\_num,1); % 电压变化量和Q有关

% 电压和角度是节点的基本特征，也是潮流计算需要求解的东西

theta = zeros(node\_num,1);

U = zeros(node\_num,1);

for i = 1:node\_num

node\_type = Bus(i,2);

% 平衡节点

if node\_type == 3

U(i) = slack\_bus\_voltage;

theta(i) = deg2rad(slack\_bus\_ang);

% PV节点

elseif node\_type == 2

generation\_MW = real(Bus(i,3));%发电机有功

load\_MW = real(Bus(i,4));%负载有功

desire\_voltage = Bus(i,6);%PV节点电压给定值

P\_set(i) = (generation\_MW - load\_MW)/base\_MVA;

U(i) = desire\_voltage;

theta(i) = 0;

% PQ节点，初始电压为1, node\_type == 1

else

generation\_MW = real(Bus(i,3));%发电机有功

generation\_MVAR = imag(Bus(i,3));%发电机无功

load\_MW = real(Bus(i,4));%负载有功

load\_MVAR = imag(Bus(i,4));%负载无功

P\_set(i) = (generation\_MW - load\_MW)/base\_MVA;

Q\_set(i) = (generation\_MVAR - load\_MVAR)/base\_MVA;

U(i) = 1;

theta(i) = 0;

end

end

# 循环

dP\_mat = zeros(50,PQ\_num+PV\_num);

dQ\_mat = zeros(50,PQ\_num);

total\_time = zeros(50,1);

countP = 1;

countQ = 1;

circle\_status = true;%循环标志位

converge\_status = false;%收敛情况

%启动计时器

tic

while circle\_status

if circle\_count > 20

disp("潮流不收敛")

%停止迭代

circle\_status = false;

converge\_status = false;

break

end

circle\_count = circle\_count + 1;

% 根据当前节点电压计算先计算P

for i = 1:node\_num

node\_type = Bus(i,2);

if node\_type ~= 3

% 计算功率的不平衡量

sum = 0;

for j = 1:node\_num

theta\_ij = theta(i) - theta(j);

sum = sum + U(j)\*(G(i,j)\*cos(theta\_ij) + B(i,j)\*sin(theta\_ij));

end

P(i) = U(i) \* sum;

end

end

% 计算偏差量

dP = P\_set - P;

% 记录迭代结果和时间

dP\_mat(countP,:) = dP;

countP = countP +1;

% 如果不满足精度

if max(abs(dP(:))) > error\_tol

% P误差不满足

dP\_flag = false;

% P不满足条件Q也要重新计算

dQ\_flag = false;

dPU = dP ./ U(1:end-1);

try

dUtheta = linsolve(B1,-dPU);

dtheta = dUtheta ./ U(1:end-1);

% 叠加修正量

for n = 1:(node\_num-1)

theta(n) = theta(n) + dtheta(n);

end

catch

disp('该方程无解！！');

circle\_status = false;

converge\_status = false;

end

else

dP\_flag = true;

if dQ\_flag == true

% 满足精度要求，终止循环

circle\_status = false;

converge\_status = true;

end

end

if circle\_status == true

% 开始计算dQ

for i = 1:node\_num

node\_type = Bus(i,2);

if node\_type == 1

% 计算功率的不平衡量

sum = 0;

for j = 1:node\_num

theta\_ij = theta(i) - theta(j);

sum = sum + U(j)\*(G(i,j)\*sin(theta\_ij) - B(i,j)\*cos(theta\_ij));

end

Q(i) = U(i) \* sum;

end

end

% 计算偏差量

dQ = Q\_set - Q;

%记录时计算时间

dQ\_mat(countQ,:) = dQ;

toc

total\_time(countQ) = toc;

if countQ > 1

total\_time(countQ) = total\_time(countQ) + total\_time(countQ-1);

end

tic;

countQ = countQ +1;

% 判断是否收敛

if max(abs(dQ(:))) > error\_tol

% Q误差不满足

dQ\_flag = false;

% Q不满足P也要重新计算

dP\_flag = false;

dQU = dQ ./ U(1:end-(1+PV\_num));

try

dU = linsolve(B2,-dQU);

% 叠加修正量

for n = 1:PQ\_num

U(n) = U(n) + dU(n);

end

catch

disp('该方程无解！！');

circle\_status = false;

converge\_status = false;

end

else

dQ\_flag = true;

if dP\_flag == true

% 满足精度要求，终止循环

circle\_status = false;

converge\_status = true;

else

circle\_status = true;

end

end

end

end

if converge\_status

disp('循环结束！循环次数为：')

disp(circle\_count)

% 计算各节点最终的电压

for i = 1:node\_num

temp\_amp = U(i);

temp\_ang = rad2deg(theta(i));

u(i) = U(i) \* complex(cos(theta(i)),sin(theta(i)));

u\_amp(i) = temp\_amp;

u\_ang(i) = temp\_ang;

end

% 计算最终各节点功率

for i = 1:node\_num

I = 0;

Ui = U(i) \* complex(cos(theta(i)),sin(theta(i)));

% 计算支路的首端功率

for j = 1:node\_num

% 计算节点注入的共轭值

Uj = U(j) \* complex(cos(theta(j)),sin(theta(j)));

I = I + Y(i,j)\*Uj;

end

% 计算各节点的功率 S = 电压 X 注入电流的共轭值

S\_node(i) = Ui\*conj(I)\*base\_MVA;

end

end

% 停止计时器并输出时间

toc

elapsed\_time = toc;

disp(['代码运行时间：', num2str(elapsed\_time), ' 秒']);

S\_node;

u\_amp;

u\_ang;

p = real(S\_node);

q = imag(S\_node);

if converge\_status

disp('迭代次数为：')

disp(circle\_count)

disp('各节点电压幅值为：')

disp(u\_amp)

disp('各节点电压相角为：')

disp(u\_ang)

disp('各节点有功为：')

disp(p)

disp('各节点无功为：')

disp(q)

end

total\_time = nonzeros(total\_time);

dP\_mat = dP\_mat(1:circle\_count,:);

dQ\_mat = dQ\_mat(1:circle\_count,:);

subplot(1,2,1);

plot(dP\_mat,'-','LineWidth',1.3);

title('ΔP收敛情况')

xlabel('迭代次数');

ylabel('修正量');

legend({'ΔP1','ΔP2','ΔP3','ΔP4','ΔP5','ΔP6'},'Location', 'northwest');

subplot(1,2,2);

plot(dP\_mat,'-','LineWidth',1.3);

title('ΔQ收敛情况')

xlabel('迭代次数');

ylabel('修正量');

legend({'ΔQ1','ΔQ2','ΔQ3','ΔQ4','ΔQ5'},'Location', 'northwest')