**电力系统分析课程设计**

课题：直角形式的牛顿-拉夫逊法潮流计算设计

专 业： 电气工程及其自动化

班 级： 电气15-4

姓 名： 张享彬、洪玉泉

学 号： 41、19

指导教师： 任红卫

**绪论**

**潮流计算是电力系统重要的计算，可以检验电力系统方案能否满足要求，可以预知各种负荷的变化和结构的改变对系统的影响。目前潮流计算方法主要有牛顿-拉夫逊法、pq分解法、高斯-塞德尔迭代法等；近年来快速解耦算法，因为其速度快和低内存越来越受欢迎，但是由于牛顿法对大多数系统准确可靠，学习并深入理解牛顿法很有必要。**

**摘要**

**牛顿-拉夫逊法（简称N-R法）是牛顿17世纪提出的一种在实数域和复数域上近似求解方程的方法，实质是局部线性化。牛顿-拉夫逊法的基本要求有三方面，一是可靠性、收敛性，二是计算速度和内存，三是方便性和灵活性。它的优点是收敛快，可靠性高，缺点是对内存要求高，对初始解要求高。**

**一、设计目的**

（一）掌握牛顿-拉夫逊法的基本原理

（二）熟悉牛顿-拉夫逊法潮流计算的步骤

（三）培养电力系统潮流计算机编程能力，掌握计算机潮流计算的相关知识。

**二、设计要求**

（一）编写潮流计算程序

（二）在计算机上调试通过

（三）运行程序并计算出正确结果

**三、总体设计**

（一）设计的总体原理框图（小四号字、宋体）

由电路图输入原始数据

形成导纳矩阵纳矩阵

计算不平衡量

计算雅可比矩阵

解修正方程组

迭代是否收敛？

N

Y

计算平衡节点功率、线路功率

（二）工作原理

牛顿-拉夫逊法是求解非线性方程常用的方法。其基本原理是将非线性方程组化为线性方程组求解。

**（1）**

**（2）**

**（3）**

**（4）**

**（5）**





**当**不满足设定的误差时修正近似解并且回代到第三步，直到误差满足要求时停止，此时的值就是符合要求的，其中k表示迭代的次数。在潮流计算中，通常表示第k次迭代后的电压值，表示第k次的电压修正量，称函数的雅可比矩阵,称为不平衡量的列向量，在潮流计算中称为节点功率或者电压平方不平衡量。

牛顿型潮流计算的核心问题是建立和求解修正方程式。假定网络中有n个PQ节点, m个PV节点,一个平衡节点，依次编号1，2，3….n+m;

=

上式的，，分别为节点注入功率和电压平方不平衡量；



式中的雅可比矩阵分别为

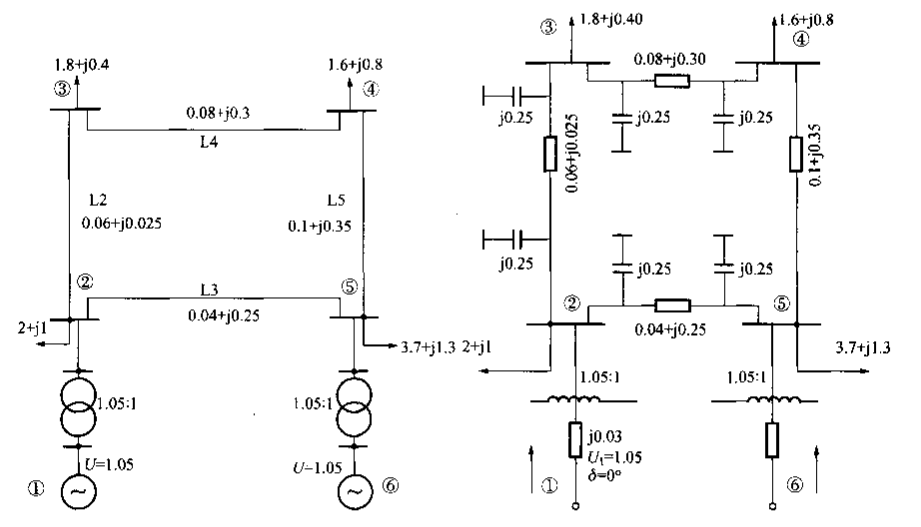
j!=i时： j=i时：

**四、仿真结果分析**

(一)输入原始数据（、、、、、 、）

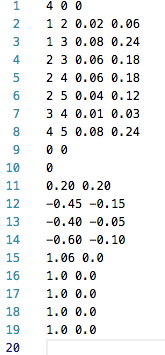
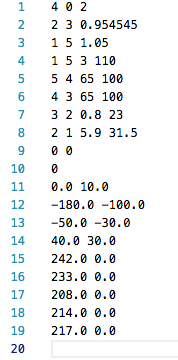
本实验的网络图如下



（注：电路图中pv节点6的有功功率为5，电压为1.1，5-6之间阻抗为0.015j,平衡节点的电压为1.1）



课本例4-2 p120页网络图数据输入示例 课本例4-1图输入数据示例

（输入说明：第一行包含三个正整数；第一个数表示pq节点的个数n,第二个表示pv节点的个数m，第三个表示变压器个数tnum。接下来的tnum行写入所以变压器变比k，第一、第二个数分别是变压器高低次侧所连节点号，第三个数为变比；（高低次侧应与变比对应）

然后输入线路的阻抗，格式为：节点号 节点号 阻抗，输入以 0 0 结束；接着输入各个节点的对地电容（不包含变压器等效电容），第一行输入含有对地电容的节点个数，

然后输入节点电容；输入节点电容后输入节点的输入功率，个数为pq与pv节点个数只和，先输入所以的pq节点，再输入pv节点；最后输入初始电压，第一行为平衡节点电压；余下n+m行是pq节点电压、pv节点电压； 整个输入过程要注意节点与数据的对应；第一个节点为平衡节点，pv节点在最后；）

在计算程序中，需要将输入数据以类似“102.txt”文本格式放在.cpp所在位置所在目录中，如果不知道，可以放在制定目录，如（D://a.txt）,并将计算程序读入文件的地址修改，在附录代码中有注释。

（二）形成导纳矩阵（Y=G+Bj）

由于c++在处理复数方面的能力较差，因此使用的是两个二维数组表示G和B；

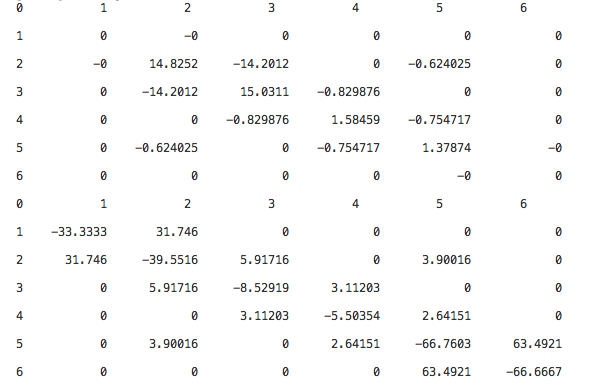


图1

理论上，仅包含变压器和线路的电力系统模型都能求出导纳矩阵；如下面图2图3分别是根据教材中的例题4-3、例题4-1网络系统求得的导纳矩阵，与教材133页和116页所示的相差不大，验证了程序的正确性。

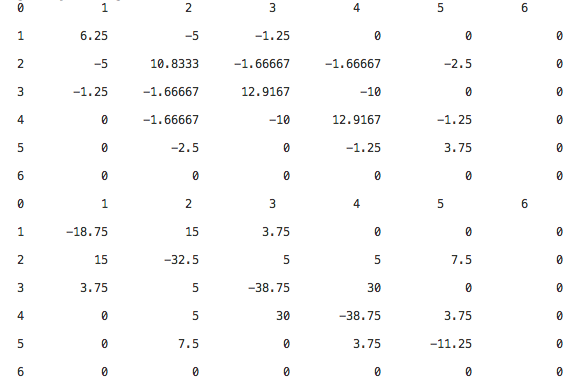
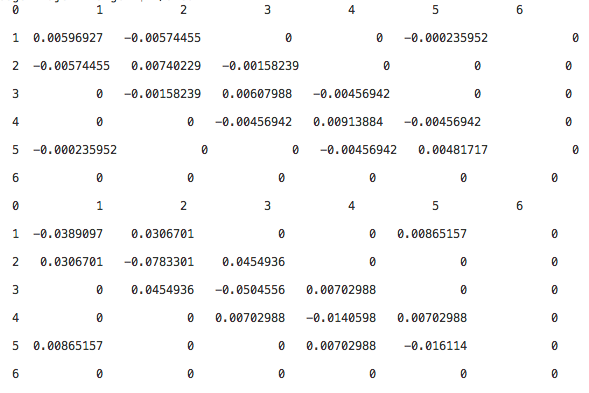


图2



图三

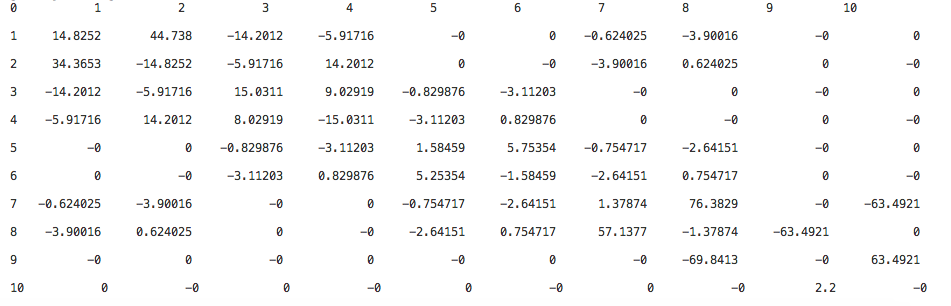
（三）计算不平衡量（、、）

求得功率的不平衡量：因为第一个节点为平衡节点，所以编号从2开始

mac os:Users:xiangbin:Desktop:107.png

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
| -2 | 4.18632 | -1.8 | 0.1 | -1.6 | -0.55 | -3.7 | 8.32264 | 5 | 0 |

(四)计算雅可比矩阵（J）

****

（五）解修正方程

算得第一次迭代的电压不平衡量：

**mac os:Users:xiangbin:Desktop:114.png**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0.162123 | 0.0152685 | -0.0936411 | 0.144102 | 0 |
|  | -0.117409 | -0.172651 | -0.340388 | 0.0657566 | 0.0382391 |
| i | 2 | 3 | 4 | 5 | 6 |

如果发现上述计算结果与matlab有偏差，不一定是程序出错，可能是由计算精度引起的误差，这仅会影响迭代的次数，在相同的误差情况下（），对最终所求的节点电压、线路功率影响有限。🈶️

（六）迭代

经过迭代后最终电压不平衡量满足要求，最终电压、平衡节点功率、线路功率如图4、图5所示

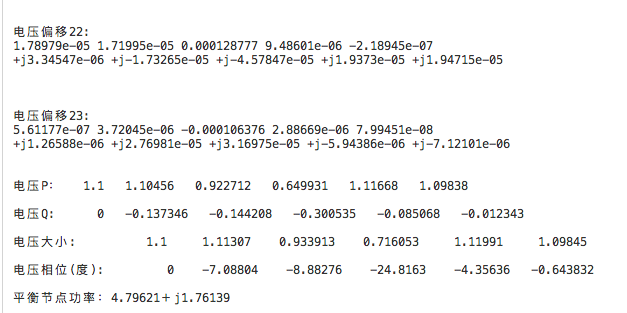


图4

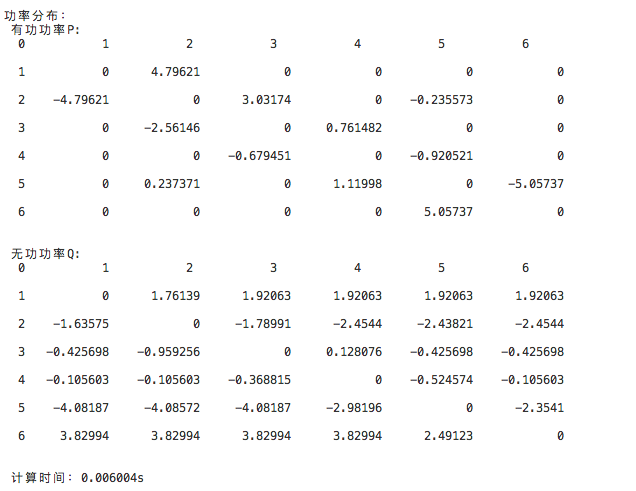
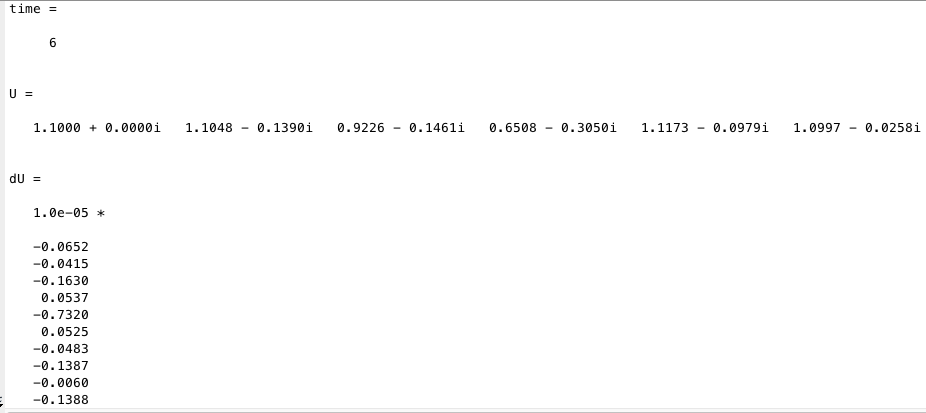


图5

图6为使用matlab计算出来的节点电压及线路功率



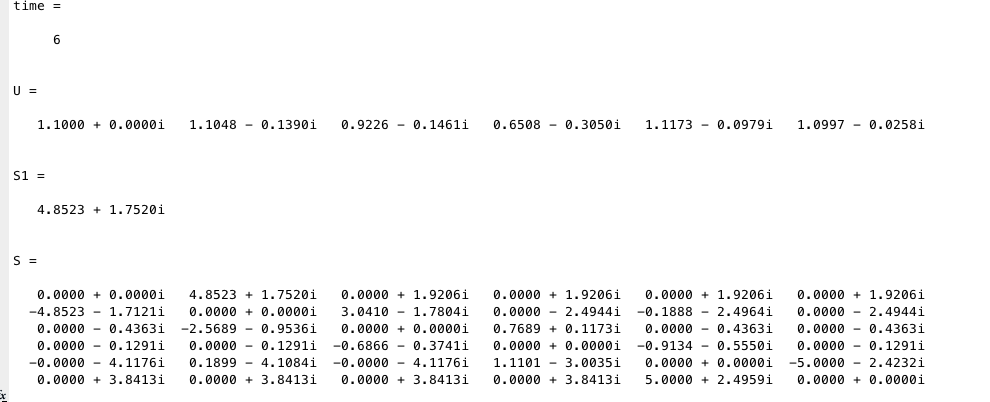


图6

通过比较可得，通过matlab和c++程序潮流计算的结果相差不大，在1%～5%之间。

图7、图8分别为教材例题4-3、4-4潮流计算所得的节点电压、平衡节点功率、线路功率

**五、结论**

1、设计过程中遇到的问题及解决方法

（1）在写线性方程组求解函数时一定要注意边界问题，还要注意错误的输入。

（1）在写数据输入函数时遇到大量问题，例如没有设置结束标志、将变压器变比与节点编号颠倒、在计算变压器对地导纳时弄错公式，解决的方法是一定要熟悉相关计算才开始写代码，写代码的前提是有思路。

（3）在输入数据会出现很多错误，而这些错误很难发现，结果不对一直以为是代码问题，检查半天发现仅仅是输入问题，所以输入数据时有必要检查数据读取的正确性。

（4）在形成雅可比矩阵函数时，要小心pv节点的r和s的处理。在迭代时有可能出现不收敛，在不确定收敛之前最好使用固定循环次数。还有结果出现问题不一定是代码原因，也可能是初值设置不恰当，在这点上很难检查出来。最好先找几个简单的电力网络检验代码是否正确。

2、设计体会

本次电力系统稳态分析实验是花费有效时间最长的一次，潮流计算程序一共四百行左右，其中求解线性方程组函数175行，这175在matlab只需要3行就解决。程序的函数有二十个左右，修改次数近百次。造成这些的原因是对编程语言不够熟悉、对牛顿法的逻辑结构的数据处理不够透切。同样让我懂得了思路才是最重要的，c++只是一个工具，只要脑中存在解决问题的具体方法，无论是c++还是matlab或者其他工具，都不影响问题的解决。

3、对设计的建议

不知道是线性方程组求解的精度问题还是代码的结构问题，电路中的节点1即平衡节点的电压为1.05时不收敛，节点6即pv节点为1.05也不能收敛，改为1.1后才能收敛。还有建立雅可比矩阵时以H N J L 还是–J H L -N组合方式对某些系统都可以潮流计算并收敛

**七、附录**

**//c++**潮流计算程序

使用该程序只要只要将数据输入文件放在正确的文件夹

//#include"LinearEquationSolve.h"

#include<cstdio>

#include<iostream>

#include<fstream>

#include<algorithm>

#include<cmath>

#include<time.h>

const double PI=3.14159246;

using namespace std;

#define N 20

ifstream fin("102.txt");//放在代码文件 .cpp所在文件夹，否则使用类似”D://102.txt”

//求解线性方程组

bool inversematrix(double (\*s)[N],int n);

void zeroline(int j,int i,int n,double(\*a)[N]);

double determinant(int m,int n,double(\*a)[N]);

bool abjustzero(int set,int end,double(\*a)[N]);

bool matrixch(int n,double(\*a)[N]);

bool matrixnubmul(int n,double b,double(\*a)[N]);

bool matrixmul(double (\*p)[N],double (\*y)[N],int a,int b,int c);

double pc(const int n,int h,int k,double(\*a)[N]);

void ax(int n,double(\*a)[N]);

bool output(int n,double(\*a)[N],double \*u,double \*out);

void show(int n,double(\*p)[N]){

for(int i=1;i<=n;i++){

for(int j=1;j<=n;j++){

printf("%lf ",p[i][j]);

}

cout<<endl;

}

}

void zeroline(int j,int i,int n,double(\*a)[N]){

double k;

k=-a[j][i]/a[i][i];

for(int t=i;t<=n;t++){

a[j][t]+=a[i][t]\*k;

}

}

double determinant(int m,int n,double(\*a)[N]){

double p=1.0;

if(((n-m)==1)){

p=a[m][m]\*a[n][n]-a[m][n]\*a[n][m];

}

else{

for(int i=m;i<=n;i++){

if(a[i][i]==0){

if(abjustzero(i,n,a)==false){

return 0;}

}

for(int j=i+1;j<=n;j++){

if(a[j][i]!=0) {

zeroline(j,i,n,a);

}

}

}

for(int i=1;i<=n;i++){

p\*=a[i][i];

}}

return p;

}

bool abjustzero(int set,int end,double(\*a)[N]){

int q=0;

if(a[set][set]==0){

for(int t=1;t<=end;t++){

if(a[set][t]!=0){

int p;

q=1;

for(int u=1;u<=end;u++){

p=a[u][set];

a[u][set]=-a[u][t];

a[u][t]=p;

}

break;

}

}

if(q==0) return false;

}

return true;

}

bool matrixch(int n,double(\*a)[N]){

//int a[n];

double p;

for(int i=1;i<=n;i++){

for(int j=i+1;j<=n;j++){

p=a[i][j];

a[i][j]=a[j][i];

a[j][i]=p;

}

}

return true;

}

bool matrixnubmul(int n,double b,double(\*a)[N]){

for(int i=1;i<=n;i++){

for(int j=1;j<=n;j++){

a[i][j]\*=b;

}

}

return true;

}

bool matrixmul(double (\*p)[N],double \*y,int a,int b,double \*r){

double s=0;

for(int i=1;i<=a;i++){

s=0.0;

for(int j=1;j<=b;j++){

s+=p[i][j]\*y[j];

}

r[i]=s;

}

return true;

}

bool inversematrix(double (\*s)[N],int n){

double p[N][N],k;

for(int i=1;i<=n;i++){

for(int j=1;j<=n;j++){

p[i][j]=s[i][j];

}

}

k=determinant(1,n,p);

//show(n,p);

//cout<<k<<endl;

if(k!=0.0){

ax(n,s);

matrixnubmul(n,1.0/k,s);

//show(n,s);

}

else {

return false;

}

return true;

}

double pc(const int n,int h,int k,double(\*a)[N]){

double s[N][N];

int p=0,b=0;

for(int i=1;i<=n;i++){

if(i!=h){

p++;

b=0;

for(int j=1;j<=n;j++){

//b=1;

if(j!=k){

b=b+1;

s[p][b]=a[i][j];

}

}

}

}

//show(n-1,s);

if(((h+k)%2)==0){

return determinant(1,n-1,s);

}

else{

return -determinant(1,n-1,s);

}

}

void ax(int n,double(\*a)[N]){

double t[N][N];

for(int i=1;i<=n;i++){

for(int j=1;j<=n;j++){

t[j][i]=pc(n,i,j,a);

}

}

for(int i=1;i<=n;i++){

for(int j=1;j<=n;j++){

a[i][j]=t[i][j];

}

}

}

bool output(int n,double(\*a)[N],double \*u,double \*out){

if(!inversematrix(a,n)) {

return false;

}

else {

matrixmul(a,u,n,n,out);

}

return true;

}

//以上为线性方程组求解

double G[N][N]={0},B[N][N]={0},e[N],f[N],P[N]={0.0},Q[N]={0.0},dut[N],a[N],b[N];

double pline[N][N],qline[N][N];

void show2(int n,double (\*a)[N]){

//ofstream fout("out.txt");

for(int i=0;i<=n;i++){

for(int j=0;j<=n;j++){

if(j==0){cout<<" "<<i<<" ";}

else{

cout.width(10);

if(i==0){cout<<j<<" ";}

else{

//printf("%lf,",a[i][j]);

cout<<a[i][j]<<" ";}

}

}

cout<<endl<<endl;

}

}

void show1(int n,double \*a){

for(int j=1;j<=n;j++){

//printf("%lf,",a[j]);

cout.width(6);

cout<<a[j]<<" ";

}

//fout<<endl<<endl;

cout<<endl<<endl;

}

void acquirepqab(int n,int m,double \*lp,double \*lq,double \*pq){

int k=1;

for(int i=1;i<=n+1+m;i++){

lp[i]=lq[i]=0.0;

a[i]=b[i]=0.0;

}

//show2(6,G);show2(6,B);

//show1(6,e);show1(6,f);

for(int i=2;i<=n+1;i++){

for(int j=1;j<=n+1+m;j++){

lp[i]+=e[i]\*(G[i][j]\*e[j]-B[i][j]\*f[j])+f[i]\*(G[i][j]\*f[j]+B[i][j]\*e[j]);

lq[i]+=f[i]\*(G[i][j]\*e[j]-B[i][j]\*f[j])-e[i]\*(G[i][j]\*f[j]+B[i][j]\*e[j]);

}

}

for(int i=2;i<=n+1;i++){

lp[i]=P[i]-lp[i];lq[i]=Q[i]-lq[i];

pq[k++]=lp[i];

pq[k++]=lq[i];

}

for(int i=n+2;i<=n+m+1;i++){

for(int j=1;j<=n+m+1;j++){

lp[i]+=e[i]\*(G[i][j]\*e[j]-B[i][j]\*f[j])-f[i]\*(G[i][j]\*f[j]+B[i][j]\*e[j]);

}

int du;

du=dut[i-n-1]\*dut[i-n-1]-e[i]\*e[i]-f[i]\*f[i];

lp[i]=P[i]-lp[i];

pq[k++]=lp[i];

pq[k++]=du;

}

for(int i=2;i<=n+m+1;i++){

for(int j=1;j<=n+m+1;j++){

if(j!=i){

a[i]+=G[i][j]\*e[j]-B[i][j]\*f[j];

b[i]+=G[i][j]\*f[j]+B[i][j]\*e[j];

}

}

a[i]+=G[i][i]\*e[i]-B[i][i]\*f[i];

b[i]+=G[i][i]\*f[i]+B[i][i]\*e[i];

}

}

void acquireJ(int n,int m,double (\*H)[N],double (\*N1)[N],double (\*J)[N],double (\*L)[N]){

for(int i=2;i<=n+1;i++){

for(int j=2;j<=n+m+1;j++){

if(i==j){

H[i][j]=-B[i][j]\*e[i]+G[i][j]\*f[i]+b[i];

N1[i][j]=G[i][j]\*e[i]+B[i][i]\*f[i]+a[i];

J[i][j]=-G[i][j]\*e[i]+B[i][j]\*f[i]+a[i];

L[i][j]=-B[i][j]\*e[i]+G[i][j]\*f[i]-b[i];

}

else{

H[i][j]=-B[i][j]\*e[i]+G[i][j]\*f[i];

N1[i][j]=G[i][j]\*e[i]+B[i][i]\*f[i];

J[i][j]=-G[i][j]\*e[i]+B[i][j]\*f[i];

L[i][j]=-B[i][j]\*e[i]+G[i][j]\*f[i];

}

}

}

for(int i=n+2;i<=n+m+1;i++){

for(int j=2;j<=n+m+1;j++){

if(i!=j){

J[i][j]=-G[i][j]\*e[i]+B[i][j]\*f[i];

H[i][j]=-B[i][j]\*e[i]+G[i][j]\*f[i];

//N1[i][j]=G[i][j]\*e[i]+B[i][i]\*f[i];

N1[i][j]=0.0;L[i][j]=0.0;}

else{

J[i][j]=-G[i][j]\*e[i]+B[i][j]\*f[i]+a[i];

H[i][j]=-B[i][j]\*e[i]+G[i][j]\*f[i]+b[i];

// N1[i][j]=G[i][j]\*e[i]+B[i][i]\*f[i]+a[i];

L[i][i]=2.0\*e[i];N1[i][i]=2.0\*f[i];

}

}

}

}

void acquirejmat(int n,int m,double (\*jj)[N],double (\*H)[N],double (\*N1)[N],double (\*J)[N],double (\*L)[N]){

int k=1,p=1;

for(int i=2;i<=n+m+1;i++){

for(int j=2;j<=n+m+1;j++){

jj[p][k++]=-J[i][j];

jj[p][k++]=H[i][j];

}

k=1;p++;

for(int j=2;j<=n+m+1;j++){

jj[p][k++]=L[i][j];

jj[p][k++]=-N1[i][j];

}

k=1;p++;

}

}

void show3(int s){

printf("\n电压P：");

show1(s,e);

printf("电压Q: ");

show1(s,f);

printf("电压大小: ");

for(int i=1;i<=s;i++){

cout.width(10);

cout<<sqrt(e[i]\*e[i]+f[i]\*f[i])<<" ";

}

printf("\n\n电压相位(度):");

for(int i=1;i<=s;i++){

cout.width(10);

cout<<atan2(f[i],e[i])\*180.0/PI<<" ";

}

for(int i=1;i<=s;i++){

P[1]+=G[1][i]\*e[i]-B[1][i]\*f[i];

Q[1]+=-(B[1][i]\*e[i])-(G[1][i]\*f[i]);

}

P[1]=P[1]\*e[1]-Q[1]\*f[1];

Q[1]=P[1]\*f[1]+Q[1]\*e[1];

cout<<endl<<endl<<"平衡节点功率：" <<P[1]<<"＋j"<<Q[1]<<endl;

///\*

for(int i=1;i<=s;i++){

for(int j=1;j<=s;j++){

if(i!=j){

double as[N][N],ad[N][N];

as[i][j]=(B[i][0]\*f[i]-e[i]\*G[i][0]-G[i][j]\*(e[i]-e[j])-B[i][j]\*(f[j]-f[i]));

ad[i][j]=(e[i]\*B[i][0]+f[i]\*G[i][0]+(e[i]-e[j])\*B[i][j]-G[i][j]\*(f[j]-f[i]));

pline[i][j]=(as[i][j]\*e[i]-ad[i][j]\*f[i]);

qline[i][j]=as[i][j]\*f[i]+ad[i][j]\*e[i];

}

}

}

cout<<endl<<"功率分布："<<endl;

cout<<" "<<"有功功率P:"<<endl;

show2(s,pline);

cout<<endl;

cout<<" "<<"无功功率Q:"<<endl;

show2(s,qline);

cout<<endl;

}

void inputdata(int \*n,int \*m,double(\*g)[N],double(\*b)[N],double \*p0,double \*q0){

int i,j,tnum=0,sp,sd=0;

double y,u;

fin>>\*n>>\*m>>tnum;

sp=\*n+\*m+1;

double k[sp][sp];

int kp[sp][2];

//输入变压器信息

for(int t1=1;t1<=tnum;t1++){

double kt;

fin>>kp[t1][0]>>kp[t1][1]>>kt;

k[kp[t1][0]][kp[t1][1]]=kt;

}

//形成导纳矩阵；

while(fin>>i>>j&& i!=0 && j!=0){

fin>>y>>u;

g[i][j]=-y/(y\*y+u\*u);

b[i][j]=u/(y\*y+u\*u);

for(int ko=1;ko<=tnum;ko++){

if(i==kp[ko][0]&&j==kp[ko][1]){

g[i][0]=g[i][j]\*(1.0-k[i][j])/k[i][j]/k[i][j];

b[i][0]=b[i][j]\*(1.0-k[i][j])/k[i][j]/k[i][j];

g[j][0]=g[i][j]\*(k[i][j]-1.0)/k[i][j];

b[j][0]=b[i][j]\*(k[i][j]-1.0)/k[i][j];

g[i][j]/=k[i][j];

b[i][j]/=k[i][j];

break;

}

if(j==kp[ko][0]&&i==kp[ko][1]){

g[j][0]=g[i][j]\*(1.0-k[j][i])/k[j][i]/k[j][i];

b[j][0]=b[i][j]\*(1.0-k[j][i])/k[j][i]/k[j][i];

g[i][0]=g[i][j]\*(k[j][i]-1.0)/k[j][i];

b[i][0]=b[i][j]\*(k[j][i]-1.0)/k[j][i];

g[i][j]/=k[j][i];

b[i][j]/=k[j][i];

break;

}

}

g[j][i]=g[i][j];

b[j][i]=b[i][j];

}

//输入对地电容节点；

fin>>sd;

while(sd--){

int i=1;

double c0=0.0;

fin>>i>>c0;

b[i][0]-=c0;

}

//形成导纳矩阵对角线；

for(int i=1;i<=sp;i++){

for(int j=0;j<=sp;j++){

if(i!=j){

g[i][i]-=g[i][j];

b[i][i]-=b[i][j];

}

}

}

//输入节点功率；

for(int i=2;i<=\*n+1;i++){

fin>>p0[i]>>q0[i];

}

for(int i=\*n+2;i<=sp;i++){

fin>>p0[i]>>dut[i-\*n-1];

}

//输入节点电压；

for(int i=1;i<=sp;i++){

fin>>e[i]>>f[i];}

}

int main(){

double lp[N],lq[N],le[N],lf[N];

double h[N][N],n1[N][N],j1[N][N],l[N][N];

double jj[N][N],pq[N];

//double pline[N][N],qline[N][N];//线路功率；

double out[N]={0.0};

int n,m,s,k=0,c=1,kp=1,kkl=1;

clock\_t uyt,uio;

double hjk;

uyt=clock();

double \*er,\*ew;

inputdata(&n,&m,G,B,P,Q);

s=n+m+1;//show2(6,G);

//show2(6,B);

do

{

kp=1;c=1;

acquirepqab(n,m,lp,lq,pq);//show1(10,pq);

acquireJ(n,m,h,n1,j1,l);

acquirejmat(n,m,jj,h,n1,j1,l);//show2(10,jj);

output(2\*(n+m),jj,pq,out);//show1(10,out);

//cout<<endl<<endl<<"节点电压"<<k<<": "<<endl;

for(int i=1;i<=s;i++){

//cout<<e[i]<<" ";

}

//cout<<endl;

for(int i=1;i<=s;i++){

// cout<<"+j"<<f[i]<<" ";

}

//cout<<endl<<endl<<"电压偏移"<<k<<": "<<endl;

for(int i=2;i<=s;i++){

// cout<<le[i]<<" ";

}

//cout<<endl;

for(int i=2;i<=s;i++){

//cout<<"+j"<<lf[i]<<" ";

}

//cout<<endl<<endl;

for(int i=2;i<=n+m+1;i++){

le[i]=out[c++];lf[i]=out[c++];}

for(int i=2;i<=n+m+1;i++){

e[i]+=le[i]; f[i]+=lf[i];}

//show1(6,e);

er=max\_element(out+1,out+2\*(n+m));

ew=min\_element(out+1,out+2\*(n+m));

k++;

}

while(\*er>1.0e-5||\*ew<-1.0e-5);

//while(kkl--);

show3(s);

uio=clock();

hjk=(double)(uio-uyt)/CLOCKS\_PER\_SEC;

cout<<" 计算时间："<<hjk<<"s"<<endl<<endl;

return 0;

}