基于牛顿--拉夫逊算法的电力系统潮流计算

电力系统分析课外作业研究报告

蒋博宇

2012104623

（三峡大学电气与新能源学院 20121096 班）

摘要：潮流计算是电力系统中的一种最基本的计算。牛顿—拉夫逊算法是经典的潮流计算算法之一，具有较好的收敛性。本文介绍了潮流计算、牛顿法潮流计算的原理、过程，给出了简单电力系统网络，并运用 MATLAB 编写潮流计算程序进行求解计算。最后，制作了一个简单的潮流计算 GUI 界面。

关键字：电力系统；牛顿—拉夫逊算法；潮流计算；用户图形界面(GUI)

Calculation of Power Flow for Power System Based On Newton-Raphson Algorithm

JIANG Bo-Yu

2012104623

Class of 20121096

(College of Electrical Engineering & New Energy ,China Three Gorges University)

Abstract: The calculation of power flow is one of the most basic calculations for power system. And Newton-Raphson algorithm is one of the classical algorithms of calculation of power flow, and it has good convergence. This article introduces the calculation of power flow and the principle and process of calculation of power flow using Newton-Raphson. Besides, this article has a example of simple power system network, and wrote the MATLAB code to solve the calculation of power flow. In the end, I have made a simple GUI of calculation of power flow.

Key Words: power system; Newton—Raphson algorithm; calculation of power flow; GUI

1. 电力系统潮流计算简介

电力系统潮流计算是电力系统分析中的一种最基本的计算，是对复杂电力系统和故障条件下稳态运行状态的计算。潮流计算的目的是求取电力系统在给定的运行状态下的计算，即节点上电压（幅值和相角）和功率的分布，用以检查系统中各元件是否过负荷、各点电压是否满足要求、功率的分布和分配是否合理。潮流计算的结果可用于如电力系统稳态研究、安全估计或者最优潮流等，同时对潮流计算的模型和方法有直接的影响。[1--2]

随着计算机技术的高速发展，电力系统潮流计算的方法也随之发生了巨大的变化。20 世纪 50 年代，普遍采用以节点导纳矩阵为基础的高斯—赛德尔法，该方法原理简单，对计算机内存需求小，但是收敛性较差；20 世纪 60 年代初期转向以阻抗矩阵为基础的阻抗法，改善了系统潮流计算的收敛性问题，但是占计算

机内存多，每次迭代的计算量大；60 年代以后陆续提出了牛顿—拉夫逊法、P—Q 分解法，这些方法在收敛性、计算机内存要求、运算速度都有了明显的改进，至今仍然被广泛的沿用。文献[3]中又指出了利用遗传算法、粒子群算法、退火算法等智能算法进行潮流计算的新方法；文献[4] 将矩阵求逆运算的松弛方法应用于电力系统潮流计算，提出了一种新的电力系统潮流的并行松弛牛顿计算方法。文献[5]提出了一种基于内点非线性规划的潮流计算模型和方法，基于 L1 范数的计算原理，将潮流方程转化为求解一个新的非线性规划模型 L1lF，并结合现代内点算法进行求解。

2. 牛顿—拉夫逊潮流算法的原理

牛顿—拉夫逊(Newton-Raphson)法，是求解非线性代数方程的一种有效且收敛速度快的迭代计算方法。下面将阐明其原理和计算方法，考虑 n 维非线性代数方程组[6]：

$$\begin{cases} f_1(x_1,x_2,\cdots x_n)=0 \\ f_2(x_1,x_2,\cdots x_n)=0 \\ \quad\vdots \\ f_n(x_1,x_2,\cdots x_n)=0 \end{cases} \tag{1}$$

假设变量的初始值给定为 $x_1^{(0)},x_2^{(0)},\cdots,x_n^{(0)}$，并设 $\Delta x_1^{(0)},\Delta x_2^{(0)},\cdots,\Delta x_n^{(0)}$ 分别为各变量的修正值，则有：

$$\begin{cases} f_1(x_1^{(0)}+\Delta x_1^{(0)},x_2^{(0)}+\Delta x_2^{(0)},\cdots x_n^{(0)}+\Delta x_n^{(0)})=0 \\ f_2(x_1^{(0)}+\Delta x_1^{(0)},x_2^{(0)}+\Delta x_2^{(0)},\cdots x_n^{(0)}+\Delta x_n^{(0)})=0 \\ \quad\vdots \\ f_n(x_1^{(0)}+\Delta x_1^{(0)},x_2^{(0)}+\Delta x_2^{(0)},\cdots x_n^{(0)}+\Delta x_n^{(0)})=0 \end{cases} \tag{2}$$

将式(2)在初始值附近展开成泰勒级数，并略去修正量的二次及以上的高次项，可以得到：

$$\begin{cases} f_1(x_1^{(0)},x_2^{(0)},\cdots x_n^{(0)})+(\dfrac{\partial f_1}{\partial x_1}\Big|_0\Delta x_1^{(0)}+\dfrac{\partial f_1}{\partial x_2}\Big|_0\Delta x_2^{(0)}\cdots+\dfrac{\partial f_1}{\partial x_n}\Big|_0\Delta x_n^{(0)})=0 \\ f_2(x_1^{(0)},x_2^{(0)},\cdots x_n^{(0)})+(\dfrac{\partial f_2}{\partial x_1}\Big|_0\Delta x_1^{(0)}+\dfrac{\partial f_2}{\partial x_2}\Big|_0\Delta x_2^{(0)}\cdots+\dfrac{\partial f_2}{\partial x_n}\Big|_0\Delta x_n^{(0)})=0 \\ \quad\vdots \\ f_n(x_1^{(0)},x_2^{(0)},\cdots x_n^{(0)})+(\dfrac{\partial f_n}{\partial x_1}\Big|_0\Delta x_1^{(0)}+\dfrac{\partial f_n}{\partial x_2}\Big|_0\Delta x_2^{(0)}\cdots+\dfrac{\partial f_n}{\partial x_n}\Big|_0\Delta x_n^{(0)})=0 \end{cases} \tag{3}$$

式中：$\dfrac{\partial f_i}{\partial x_j}\Big|_0$ 为函数 $f_i(x_1,x_2,\cdots x_n)$ 对自变量 x_j 的偏导数在 $(x_1^{(0)},x_2^{(0)},\cdots x_n^{(0)})$ 处的取值。

将(3)式表示成矩阵形式可得：

$$\begin{bmatrix} f_1(x_1^{(0)},x_2^{(0)},\cdots x_n^{(0)}) \\ f_2(x_1^{(0)},x_2^{(0)},\cdots x_n^{(0)}) \\ \vdots \\ f_3(x_1^{(0)},x_2^{(0)},\cdots x_n^{(0)}) \end{bmatrix} = -\begin{bmatrix} \frac{\partial f_1}{\partial x_1}\Big|_0 & \frac{\partial f_1}{\partial x_2}\Big|_0 & \cdots & \frac{\partial f_1}{\partial x_n}\Big|_0 \\ \frac{\partial f_2}{\partial x_1}\Big|_0 & \frac{\partial f_2}{\partial x_2}\Big|_0 & \cdots & \frac{\partial f_2}{\partial x_n}\Big|_0 \\ \vdots & \vdots & & \\ \frac{\partial f_n}{\partial x_1}\Big|_0 & \frac{\partial f_n}{\partial x_2}\Big|_0 & \cdots & \frac{\partial f_n}{\partial x_n}\Big|_0 \end{bmatrix} \begin{bmatrix} \Delta x_1^{(0)} \\ \Delta x_2^{(0)} \\ \vdots \\ \Delta x_n^{(0)} \end{bmatrix} \tag{4}$$

并用向量 $x, x^{(0)}, \Delta x^{(0)}$ 分别表示由变量、变量初值和修正量所组成的向量，得到修正方程式：

$$f(x^{(0)}) = -J^{(0)}\Delta x^{(0)} \tag{5}$$

式(5)中的矩阵 J 为向量 $f(x)$ 对变量 x 的一阶导数，即式(4)中右端的系数矩阵，这一矩阵称为雅克比矩阵。于是可以得到修正后的 x 值：

$$\begin{cases} x_1^{(1)} = x_1^{(0)} + \Delta x_1^{(0)} \\ x_2^{(1)} = x_2^{(0)} + \Delta x_2^{(0)} \\ \vdots \\ x_n^{(1)} = x_n^{(0)} + \Delta x_n^{(0)} \end{cases} \tag{6}$$

然后，以 $x_1^{(1)}, x_2^{(1)}, \cdots, x_n^{(1)}$ 作为新的初值，重新形成并求解新的修正方程式，便可以解出 $\Delta x_1^{(1)}, \Delta x_2^{(1)}, \cdots, \Delta x_n^{(1)}$。依次类推。

于是，得出多维的非线性代数方程的牛顿法迭代格式为：

$$\begin{cases} \Delta x^{(k)} = -[J^{(k)}]^{-1} f(x^{(k)}) \\ x^{(k+1)} = x^{(k)} + \Delta x^{(k)} \end{cases}; k = 0,1,2,\cdots; \tag{7}$$

迭代的收敛判据通常采用：

$$\max_i |f_i(x^{(k)})| < \varepsilon \tag{8}$$

式(8)中 $|f_i(x^{(k)})|$ 表示向量 $f(x^{(k)})$ 中各分量绝对值最大者；ε 为给定的容许误差。

2.1 潮流计算中节点的分类

在潮流计算中，给定的量应该是负荷吸收的功率、发电机发出的功率或者发电机的电压。按照给定量种类的不同，可以将节点分为以下三类：

(1)PQ 节点

给定节点的注入有功功率 P 和注入无功功率 Q。这类节点对应于实际系统中的纯负荷节点(如变电所母线)、有功和无功功率都给定的发电机节点以及联络节点(注入有功和无功功率等于 0)

(2)PV 节点

给定节点的注入有功功率 P 和节点电压的有效值 U，待求量是节点的注入无功功率 Q 和电压的相位 θ。这类节点通常为发电机节点，其有功功率个给定而且有比较大的无功容量，它们能依靠自动电压调节器的作用使母线电压保持为给定值。

(3)平衡节点

在潮流计算中，必须设置一个平衡节点，其电压有效值为给定值，电压相位为$\theta=0$，即系统中其他各节点的电压相位都以它为参考；而其注入的有功功率和无功功率都是待求量。平衡节点的注入有功功率必须平衡全系统的有功功率和有功损耗，所以其值不能加以给定。

进行计算时，平衡节点是必不可少的；PQ 节点是大量的；PV 节点是少量，甚至没有的。[6-7]

2.2 极坐标形式的牛顿—拉夫逊潮流算法

2.2.1 节点的功率平衡方程

由 2.1 可以知道，在电力系统潮流计算中，将全部的节点分成 PQ 节点、PV 节点和平衡节点。设系统中有n个节点，其中有m个 PQ 节点，$n-m-1$个 PV 节点，1 个平衡节点。为了方便叙述，假定节点按照先 PQ 节点，再 PV 节点，最后平衡节点的顺序进行编号，如下表 1 所示：

表 1：系统中节点的编号、种类、个数

Table1：The number & type & amount of node in power system

节点编号	种类	个数
$1,2,\cdots,m$	PQ 节点	m
$m+1,m+2,\cdots,n-1$	PV 节点	$n-m-1$
n	平衡节点	1

假定所有节点都含有发电机和负荷，分别对各类节点列出电压用极坐标形式表示的功率方程。

(1)PQ 节点功率方程式

$$\begin{cases} P_{Gi}-P_{Li}=U_i\sum_{j\in i}U_j(G_{ij}\cos\theta_{ij}+B_{ij}\sin\theta_{ij}) \\ Q_{Gi}-Q_{Li}=U_i\sum_{j\in i}U_j(G_{ij}\sin\theta_{ij}-B_{ij}\cos\theta_{ij}) \end{cases};\ i=1,2,\cdots,m \qquad (9)$$

式中：P_{Gi}、P_{Li}、Q_{Gi}和Q_{Li} $(i=1,2,\cdots,m)$都是给定值，而U_i和θ_i $(i=1,2,\cdots,m)$都待求。

(2)PV 节点的功率方程式

$$\begin{cases} P_{Gi}-P_{Li}=U_i\sum_{j\in i}U_j(G_{ij}\cos\theta_{ij}+B_{ij}\sin\theta_{ij}) \\ Q_{Gi}-Q_{Li}=U_i\sum_{j\in i}U_j(G_{ij}\sin\theta_{ij}-B_{ij}\cos\theta_{ij}) \end{cases};\ i=m+1,\cdots,n-1 \quad (10)$$

式中：P_{Gi}、P_{Li}和U_i $(i=m+1,\cdots,n-1)$都是给定值，而$Q_{Gi}-Q_{Li}$和θ_i $(i=m+1,\cdots,n-1)$待求。

(3)平衡节点的功率方程式

$$\begin{cases} P_{Gi}-P_{Li}=U_i\sum_{j\in i}U_j(G_{ij}\cos\theta_{ij}+B_{ij}\sin\theta_{ij}) \\ Q_{Gi}-Q_{Li}=U_i\sum_{j\in i}U_j(G_{ij}\sin\theta_{ij}-B_{ij}\cos\theta_{ij}) \end{cases};\ i=n \qquad (11)$$

式中：U_n和θ_n为给定值，且一般取$\theta_n=0$，即平衡节点的电压相位为参考，

而 P_{Li} 和 $Q_{Gi}-Q_{Li}$ 为待求量。

综合以上三类节点的功率方程式，可以看出，在潮流计算中实际需要求解的非线性方程组为

$$\begin{cases} P_{Gi}-P_{Li}=U_i\sum_{j\in i}U_j(G_{ij}\cos\theta_{ij}+B_{ij}\sin\theta_{ij});i=1,2,\cdots,n-1 \\ Q_{Gi}-Q_{Li}=U_i\sum_{j\in i}U_j(G_{ij}\sin\theta_{ij}-B_{ij}\cos\theta_{ij});i=1,2,\cdots,m \end{cases} \tag{12}$$

它包含 $n-1$ 个有功功率方程和 m 个无功功率方程，总共 $n+m-1$ 个。

2.2.2 Jacobi 矩阵的求解

将式(12)改写成式(13)形式，

$$\begin{cases} \Delta P_i(x)=P_{Gi}-P_{Li}-U_i\sum_{j\in i}U_j(G_{ij}\cos\theta_{ij}+B_{ij}\sin\theta_{ij});i=1,2,\cdots,n-1 \\ \Delta Q_i(x)=Q_{Gi}-Q_{Li}-U_i\sum_{j\in i}U_j(G_{ij}\sin\theta_{ij}-B_{ij}\cos\theta_{ij});i=1,2,\cdots,m \end{cases} \tag{13}$$

Jacobi 矩阵如下式(14)所示：

$$\begin{bmatrix} \Delta P_1(x) \\ \vdots \\ \Delta P_{n-1}(x) \\ \hdashline \Delta Q_1(x) \\ \vdots \\ \Delta Q_m(x) \end{bmatrix} = -\left[\begin{array}{ccc:ccc} H_{11} & \cdots & H_{1,n-1} & N_{11} & \cdots & N_{1,n-1} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ H_{n-1,1} & \cdots & H_{n-1,n-1} & N_{n-1,1} & \cdots & N_{n-1,n-1} \\ \hdashline M_{11} & \cdots & M_{1,n-1} & L_{11} & \cdots & L_{1,n-1} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ M_{m,1} & \cdots & M_{n-1,n-1} & L_{n-1,1} & \cdots & L_{n-1,n-1} \end{array}\right] \begin{bmatrix} \Delta\theta_1 \\ \vdots \\ \Delta\theta_{n-1} \\ \Delta U_1/U_1 \\ \vdots \\ \Delta U_m/U_m \end{bmatrix} \tag{14}$$

对于式(14) Jacobi 矩阵由虚线分开的各分块矩阵之非对角元素 $(i\neq j)$ 有：

$$\begin{cases} H_{ij}=\dfrac{\partial\Delta P_i}{\partial\theta_j}=-U_iU_j(G_{ij}\sin\theta_{ij}-B_{ij}\cos\theta_{ij}) \\ N_{ij}=\dfrac{\partial\Delta P_i}{\partial U_j}U_j=-U_iU_j(G_{ij}\cos\theta_{ij}+B_{ij}\sin\theta_{ij}) \\ M_{ij}=\dfrac{\partial\Delta Q_i}{\partial\theta_j}=U_iU_j(G_{ij}\cos\theta_{ij}+B_{ij}\sin\theta_{ij}) \\ L_{ij}=\dfrac{\partial\Delta Q_i}{\partial U_j}U_j=-U_iU_j(G_{ij}\sin\theta_{ij}-B_{ij}\cos\theta_{ij}) \end{cases} \tag{15}$$

对于各分块矩阵的对角元素 $(j=i)$ 有

$$\begin{cases} H_{ii}=\dfrac{\partial \Delta P_i}{\partial \theta_i}=U_i\sum\limits_{\substack{j\in i\\ j\neq i}}U_j(G_{ij}\sin\theta_{ij}-B_{ij}\cos\theta_{ij}) \\ N_{ii}=\dfrac{\partial \Delta P_i}{\partial U_i}U_i=-U_i\sum\limits_{\substack{j\in i\\ j\neq i}}U_j(G_{ij}\cos\theta_{ij}+B_{ij}\sin\theta_{ij})-2U_i^2G_{ii} \\ M_{ii}=\dfrac{\partial \Delta Q_i}{\partial \theta_i}=-U_i\sum\limits_{\substack{j\in i\\ j\neq i}}U_j(G_{ij}\cos\theta_{ij}+B_{ij}\sin\theta_{ij}) \\ L_{ii}=\dfrac{\partial \Delta Q_i}{\partial U_i}U_i=-U_i\sum\limits_{\substack{j\in i\\ j\neq i}}U_j(G_{ij}\sin\theta_{ij}-B_{ij}\cos\theta_{ij})+2U_i^2B_{ii} \end{cases} \tag{16}$$

将 Jacobi 矩阵改写成

$$\begin{bmatrix}\Delta P^{(k)}\\ \Delta Q^{(k)}\end{bmatrix}=-\begin{bmatrix}H^{(k)} & N^{(k)}\\ M^{(k)} & L^{(k)}\end{bmatrix}\begin{bmatrix}\Delta\theta^{(k)}\\ \Delta U'^{(k)}\end{bmatrix} \tag{17}$$

其中：

$$\begin{cases}\Delta\theta^{(k)}=[\Delta\theta_1^{(k)}\ \ \Delta\theta_2^{(k)}\ \cdots\ \Delta\theta_{n-1}^{(k)}]\\ \Delta U'^{(k)}=[\Delta U_1^{(k)}/U_1^{(k)}\ \ \Delta U_2^{(k)}/U_2^{(k)}\ \cdots\ \ \Delta U_{n-1}^{(k)}/U_{n-1}^{(k)}]\end{cases} \tag{18}$$

对修正方程式(17)进行求解后，可以得到修正量$\Delta\theta^{(k)}$和$\Delta U'^{(k)}$，从而可以得出：

$$\begin{cases}\theta_i^{(k+1)}=\theta_i^{(k)}+\Delta\theta_i^{(k)};i=1,2,\cdots,n-1\\ U_i^{(k+1)}=U_i^{(k)}+U_i^{(k)}\times(\dfrac{\Delta U_i^{(k)}}{U_i^{(k)}});i=1,2,\cdots,m\end{cases} \tag{19}$$

式(16)和式(19)组成牛顿法计算潮流的迭代格式，可见，形成 Jacobi 矩阵和求解修正方程式是求解牛顿法潮流计算中的主体。

牛顿法潮流计算的判据一般取：

$$\max_i|\Delta P_i^{(k)}|<\varepsilon\wedge\max_i|\Delta Q_i^{(k)}|<\varepsilon \tag{20}$$

式中：ε为节点功率的不平衡量的允许误差，其取值范围一般为$10^{-7}\sim10^{-3}$。

极坐标形式下的牛顿—拉夫逊算法步骤如下所示：

(1) 输入原始数据。其包括系统的信息，各线路和变压器所在节点的编号和等值电路中的参数；各负荷所在节点的编号及其所取用的有功功率和无功功率；PQ、PV 以及平衡节点的已知数据；

(2) 形成节点导纳矩阵；

(3) 给定各 PQ 节点的电压初始值和除平衡节点外各节点电压相位的初始值$\theta_i^{(0)}$并组成待求的初始向量$U^{(0)},\theta^{(0)}$；

(4) 置迭代次数$k=0$；

(5) 应用$U^{(k)},\theta^{(k)}$及 PV 节点和平衡节点所给定的电压，按式(13)计算各 PQ 节点的有功功率和无功功率$\Delta P_i^{(k)}$和$\Delta Q_i^{(k)}$以及各 PV 节点的有功功率误差$\Delta P_i^{(k)}$，并组成功率误差向量$\Delta P^{(k)}$和$\Delta Q^{(k)}$；

(6) 按照式(20)中的收敛判据判断最大的功率误差是否小于容许值，如果满足则转向第(11)步，否则进行下一步；

(7) 应用$U^{(k)},\theta^{(k)}$按式(15)和式(16)计算 Jacobi 矩阵元素；

(8) 解修正方程式(17)，得到修正量$\Delta\theta^{(k)}$和$\Delta U'^{(k)}$；

(9) 应用式(19)计算各节点电压和相位的修正值，即新的初值；

(10) 置迭代次数$k=k+1$，返回第(5)步进行下一轮迭代；

(11) 按式(11)计算平衡节点的发电机有功功率和无功功率，用式(10)计算各PV节点发电机的无功功率；元件两端的功率、电流和损耗，最后输出计算结果。

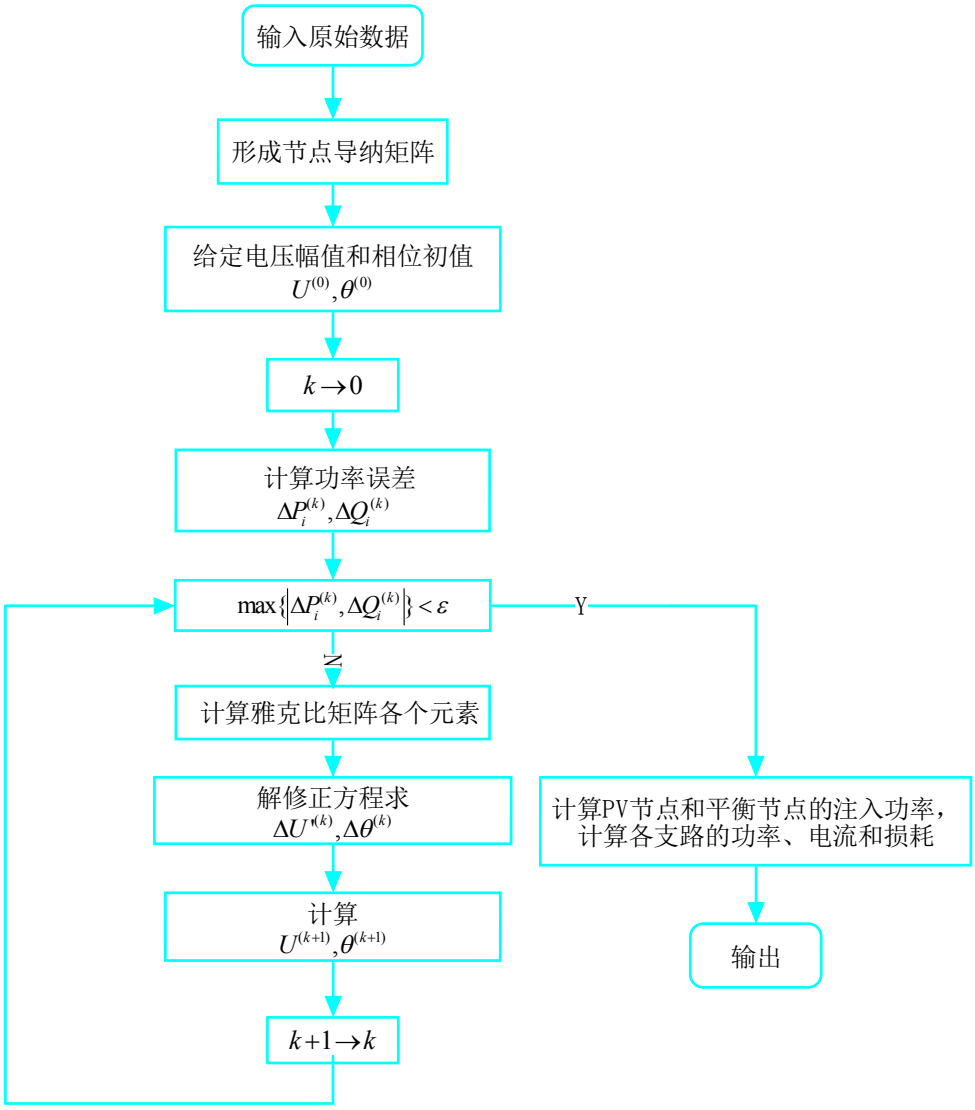

图 1：极坐标坐标形式下的牛顿—拉夫逊法程序流程图

Fig.1：The flow chart of Newton Laphson algorithm in polar coordination form

2.3 直角坐标形式的牛顿—拉夫逊潮流算法

在采用直角坐标时，节点电压可表示为：

$$\dot{V}_i=e_i+jf_i \tag{21}$$

节点导纳矩阵元素则表示为：

$$Y_{ij}=G_{ij}+jB_{ij} \tag{22}$$

假设有m个PQ节点，第i个节点的给定功率设为P_{is}和Q_{is}，则可得方程：

$$\begin{cases}\Delta P_i = P_{is} - e_i\sum_{j=1}^{n}(G_{ij}e_j - B_{ij}f_j) - f_i\sum_{j=1}^{n}(G_{ij}f_j + B_{ij}e_j) = 0 \\ \Delta Q_i = Q_{is} - f_i\sum_{j=1}^{n}(G_{ij}e_j - B_{ij}f_j) + e_i\sum_{j=1}^{n}(G_{ij}f_j + B_{ij}e_j) = 0\end{cases};(i = 1,2,\cdots,m) \quad (23)$$

假设系统中的第$m+1,m+2,\cdots,n-1$号节点为PV节点，则对其中每一个节点列写方程：

$$\begin{cases}\Delta P_i = P_{is} - e_i\sum_{j=1}^{n}(G_{ij}e_j - B_{ij}f_j) - f_i\sum_{j=1}^{n}(G_{ij}f_j + B_{ij}e_j) = 0 \\ \Delta V_i^2 = V_{is}{}^2 - V_i^2 = V_{is}{}^2 - (e_i^2 + f_i^2) = 0\end{cases} \quad (24)$$

$$(i = m+1, m+2, \cdots, n-1)$$

第n号节点为平衡节点，其电压$V_n = e_n + jf_n$是给定的，故不参加迭代。

写出修正方程式：

$$\Delta W = -J\Delta V \quad (25)$$

式中

$$\Delta W = [\Delta P_i \ \ \Delta Q_i \ \ \cdots \ \ \Delta P_m \ \ \Delta Q_m \ \ \Delta P_{m+1} \ \ \Delta V_{m+1}{}^2 \ \ \cdots \ \ \Delta P_{n-1} \ \ \Delta V_{n-1}{}^2]^T$$

$$\Delta V = [\Delta e_i \ \ \Delta f_i \ \ \cdots \ \ \Delta e_m \ \ \Delta f_m \ \ \Delta e_{m+1} \ \ \Delta f_{m+1} \ \ \cdots \ \ \Delta e_{n-1} \ \ \Delta f_{n-1}]^T$$

下面将求解Jacobi矩阵中的各个元素，当$i \neq j$时

$$\begin{cases}\dfrac{\partial \Delta P_i}{\partial e_j} = -\dfrac{\partial \Delta Q_i}{\partial f_j} = -(G_{ij}e_j + B_{ij}f_i) \\ \dfrac{\partial \Delta P_i}{\partial f_j} = \dfrac{\partial \Delta Q_i}{\partial e_j} = B_{ij}e_j - G_{ij}f_i \\ \dfrac{\partial \Delta V_i^2}{\partial e_j} = \dfrac{\partial \Delta V_i^2}{\partial f_j} = 0\end{cases} \quad (26)$$

当$j = i$时

$$\begin{cases}\dfrac{\partial \Delta P_i}{\partial e_j} = -\sum_{k=1}^{n}(G_{ik}e_k - B_{ik}f_k) - G_{ii}e_i - B_{ii}f_i \\ \dfrac{\partial \Delta P_i}{\partial f_j} = -\sum_{k=1}^{n}(G_{ik}f_k + B_{ik}e_k) + B_{ii}e_i - G_{ii}f_i \\ \dfrac{\partial \Delta Q_i}{\partial e_j} = \sum_{k=1}^{n}(G_{ik}f_k + B_{ik}e_k) + B_{ii}e_i - G_{ii}f_i \\ \dfrac{\partial \Delta Q_i}{\partial f_j} = -\sum_{k=1}^{n}(G_{ik}e_k - B_{ik}f_k) + G_{ii}e_i + B_{ii}f_i \\ \dfrac{\partial \Delta V_i^2}{\partial e_j} = -2e_i \\ \dfrac{\partial \Delta V_i^2}{\partial f_j} = -2f_i\end{cases} \quad (27)$$

将修正方程写分块矩阵的形式

$$\begin{bmatrix}\Delta W_1\\ \Delta W_2\\ \vdots\\ \Delta W_{n-1}\end{bmatrix}=-\begin{bmatrix}J_{11} & J_{12} & \cdots & J_{1,n-1}\\ J_{21} & J_{22} & \cdots & J_{2,n-1}\\ \vdots & \vdots & \cdots & \vdots\\ J_{n-1,1} & J_{n-1,2} & \cdots & J_{n-1,n-1}\end{bmatrix}\begin{bmatrix}\Delta V_1\\ \Delta V_2\\ \vdots\\ \Delta V_{n-1}\end{bmatrix} \tag{28}$$

式中，ΔW_i 和 ΔV_i 都是二维向量；J_{ij} 是 2×2 阶方阵。

$$\Delta V_i=\begin{bmatrix}\Delta e_i\\ \Delta f_i\end{bmatrix}$$

对于 PQ 节点：

$$\Delta W_i=\begin{bmatrix}\Delta P_i\\ \Delta Q_i\end{bmatrix}$$

$$J_{ij}=\begin{bmatrix}\dfrac{\partial\Delta P_i}{\partial e_j} & \dfrac{\partial\Delta P_i}{\partial f_j}\\ \dfrac{\partial\Delta Q_i}{\partial e_j} & \dfrac{\partial\Delta Q_i}{\partial f_j}\end{bmatrix} \tag{29}$$

对于 PV 节点：

$$\Delta W_i=\begin{bmatrix}\Delta P_i\\ \Delta V_i^2\end{bmatrix}$$

$$J_{ij}=\begin{bmatrix}\dfrac{\partial\Delta P_i}{\partial e_j} & \dfrac{\partial\Delta P_i}{\partial f_j}\\ \dfrac{\partial\Delta V_i^2}{\partial e_j} & \dfrac{\partial\Delta V_i^2}{\partial f_j}\end{bmatrix} \tag{30}$$

直角坐标形式下的牛顿—拉夫逊法程序流程图如图 2 所示：

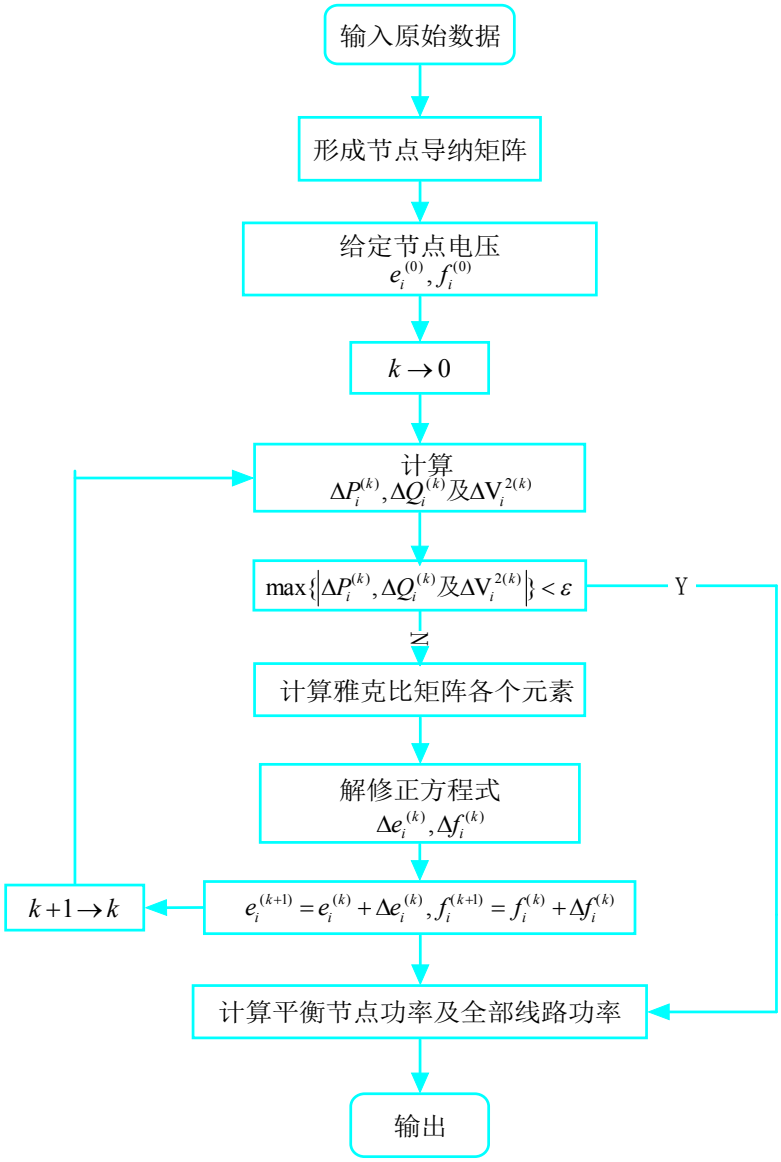

图 2：直角坐标形式下的牛顿—拉夫逊法程序流程图

Fig.2：The flow chart of Newton Laphson algorithm in rectangular form

3. MTLAB 及其用户图形界面（GUI）简介

3.1 MATLAB 简介

MATLAB 是美国 MathWorks 公司出品的商业数学软件，用于算法开发、数据可视化、数据分析以及数值计算的高级技术计算语言和交互式环境，主要包括 MTLAB 和 Simulink 两大部分。MATLAB 是矩阵实验室 (Matrix Laboratory) 的简称，是一种功能强大、效率高并且便于进行科学和工程计算的交互式软件包。MATLAB 作为一种新的计算机语言，不像学习其他高级语言如 Basic、Fortran、C 等那样难于掌握。

3.2 MATLAB 用户图形界面（GUI）简介

图形用于界面（Graphical User Interfaces，GUI）[8]是提供人机交互的工具和方法。GUI 是包含图形对象（如窗口、图标、菜单和文本）的用户界面。以某种方式选择或激活这些对象时，通常会引起动作或者发生变化。一个设计优秀的 GUI 能够非常直观的让用户知道如何操作 MATLAB 界面，并且了解设计者的开发意图。MATLAB 的 GUI 为开发者提供了一个不脱离 MATLAB 的开发环境，有助于 MATLAB 程序的 GUI 集成。这样可以使开发者不必理会一大堆烦杂的代码，简化程序，但是同样可以实现向决策者提供图文并茂的界面，甚至达到多媒体的效果。可以说 MATLAB 提供了一个简便的开发环境，可以让开发者快速上手，提高了开发者的工作效率。

GUIDE 就是图形用户界面开发环境（Graphical User Interface Development Environment），它向用户提供了一系列的创建用户图形界面的工具。这些工具大大简化了 GUI 设计和生成的过程。GUIDE 可以完成的任务有如下两点：(1)输出 GUI；(2)GUI 编程。GUIDE 实际上是一套 MATLAB 工具集，它主要由七部分组成：版面设计器、属性编辑器、菜单编辑器、调整工具、对象浏览器、Tab 顺序编辑器、M 文件编辑器。

4. 潮流计算实例描述

本次潮流计算的算例来自于华中科技大出版社出版的何仰赞著作的《电力系统分析》(第三版)(下册)P61 页例 11-5，其简单的电力系统如下图 3 所示：

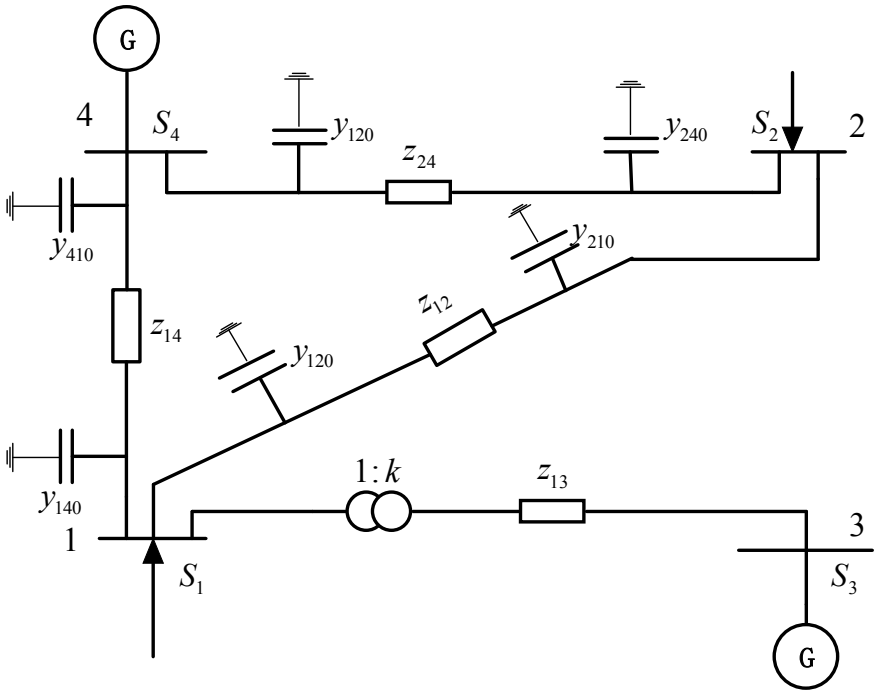

图 3：潮流计算的电力系统

Fig.3: The power system of calculation of power flow

在图 3 所示的简单电力系统中，网络各元件参数的标幺值如下：

$$z_{12} = 0.10 + j0.40$$
$$y_{120} = y_{210} = j0.01528$$
$$z_{13} = j0.3, k = 1.1$$
$$z_{14} = 0.12 + j0.50$$
$$y_{140} = y_{410} = j0.01920$$
$$z_{24} = 0.08 + j0.4$$
$$y_{240} = y_{420} = j0.01413$$

系统中节点 1、2 为 PQ 节点，节点 3 为 PV 节点，节点 4 为平衡节点，已给定

$$P_{1s} + jQ_{1s} = -0.30 - j0.18$$
$$P_{2s} + jQ_{2s} = -0.55 - j0.13$$
$$P_{3s} = 0.5, V_{3r} = 1.10, \quad V_{4s} = 1.05\angle 0°$$

容许误差为$\varepsilon = 10^{-5}$。试用牛顿法计算潮流分布。

5. 潮流计算结果展示与分析

首先，对于上面所给出的潮流计算实例中含有变压器，因此应该将变压器等效为π型等效电路，下面举例说明：

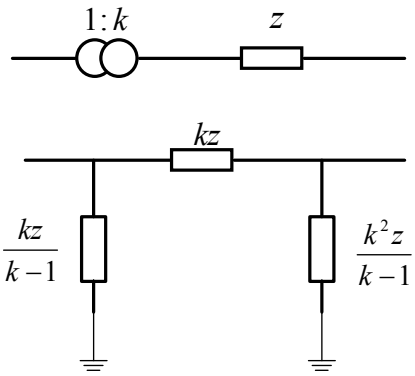

图 4：变压器π型等效电路实例

Fig.4：The example of transformer equivalent circuit

然后采用直角坐标形式的牛顿—拉夫逊潮流计算算法，按照算法的流程图，通过 MATLAB 软件编写潮流计算程序,运行程序后发现，第 5 次迭代后收敛，即节点电压的不平衡量的最大值小于容许误差，节点导纳矩阵、收敛后的节点电压、平衡节点功率如下表 2-5 所示，同时 Jacobi 矩阵、线路功率也相应求出，请运行 MATLAB 程序查看，此处暂不列出。

表 2：迭代收敛后复数形式的节点电压

Tab.2：The node voltage in plural form after iterative convergence

节点编号	节点 1	节点 2	节点 3	节点 4
节点电压	1.0913 - 0.0352i	1.0384 - 0.1174i	1.0939 + 0.1159i	1.0500 + 0.0000i

表 3：迭代收敛后相角形式的节点电压

Tab.3：The node voltage in phase angle form after iterative convergence

节点编号	节点 1	节点 2	节点 3	节点 4
节点电压幅值	1.0919	1.0451	1.1000	1.0500
节点电压相角	–1.8480°	-6.4522°	6.0483°	0°

表 4：迭代收敛后的节点功率

Tab.4：The node power after iterative convergence

节点编号	节点 1	节点 2	节点 3	节点 4
节点功率	-0.3000 - 0.1800i	-0.5500 - 0.1300i	0.5000 + 0.0000i	0.0156 - 0.0212i

表 5：节点导纳矩阵

Tab.4：The matrix of node admittance

节点编号	节点 1	节点 2	节点 3	节点 4
节点 1	1.0421 - 6.9368i	-0.5882 + 2.3529i	0.0000 + 3.0303i	-0.4539 + 1.8911i
节点 2	-0.5882 + 2.3529i	1.0690 - 4.7274i	0.0000 + 0.0000i	-0.4808 + 2.4038i
节点 3	0.0000 + 3.0303i	0.0000 + 0.0000i	0.0000 - 2.7548i	0.0000 + 0.0000i
节点 4	-0.4539 + 1.8911i	-0.4808 + 2.4038i	0.0000 + 0.0000i	0.9488 - 4.2757i

此外，还做出了各个节点电压的幅值与迭代次数的关系曲线图，如下图 5 所示：

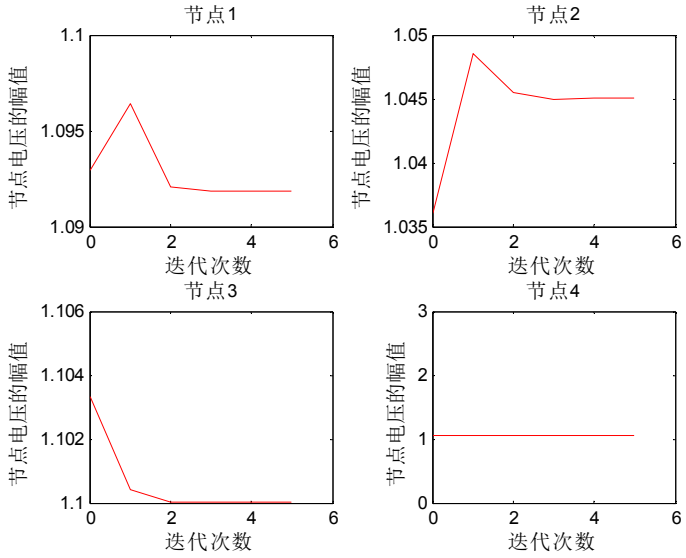

图 5：各个节点电压的幅值与迭代次数的曲线图

Fig.5：The relationship between the amplitude of each node voltage and iterations

通过图 5 可以看出，节点 1、2、3 在前两次迭代时电压波动比较大，但是从第 3 次迭代以后逐渐趋于稳定，最后基本不变，而整个潮流计算过程中，平衡节点(节点 4)的电压始终保持不变。

最后，我还尝试用 MATLAB 中的用户图形界面(GUI)制作了一个简单的基于牛顿—拉夫逊法潮流计算的 GUI 界面，将其展示如下图 6-8 所示：

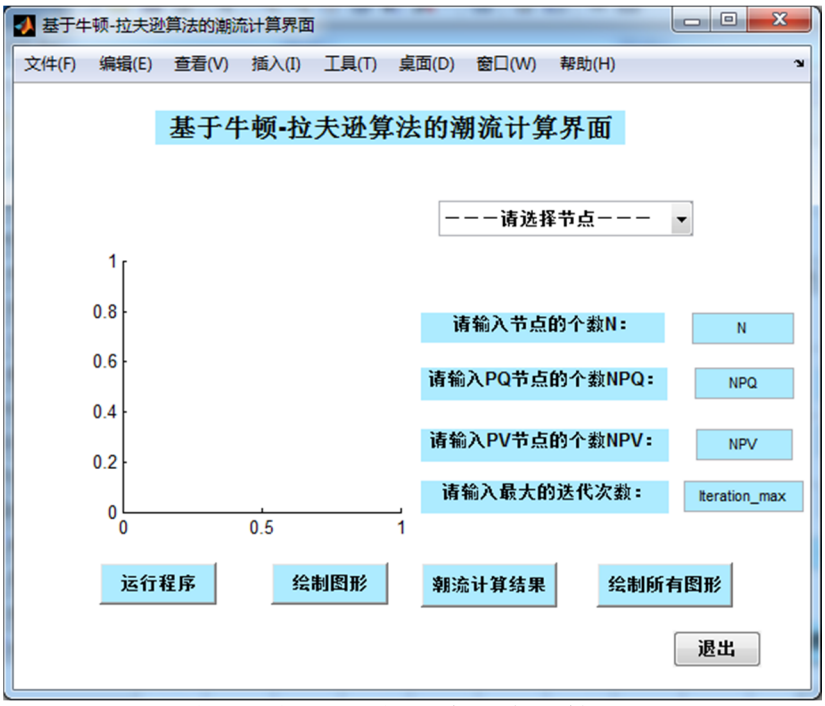

图 6：基于牛顿—拉夫逊法潮流计算 GUI 界面

Fig.6：The GUI of Calculation of Power Flow Based On Newton-Raphson Algorithm

图 7：基于牛顿—拉夫逊法潮流计算 GUI 界面

Fig.7：The GUI of Calculation of Power Flow Based On Newton-Raphson Algorithm

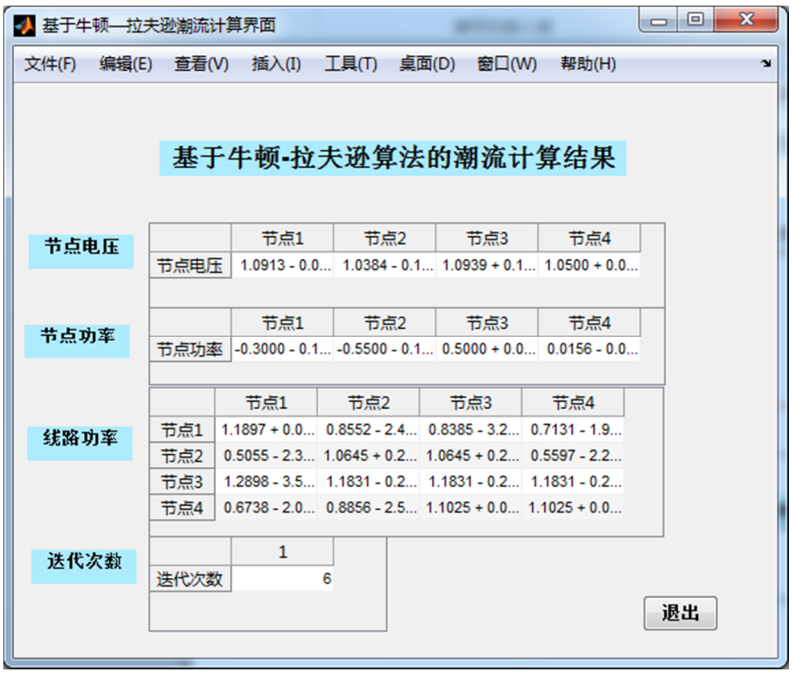

图 8：基于牛顿—拉夫逊法潮流计算 GUI 界面

Fig.8：The GUI of Calculation of Power Flow Based On Newton-Raphson Algorithm

6. 心得体会

经过一个多星期的坚持不懈和钻研，我终于完成这个电力系统潮流计算课外作业，顿时心中小有成就感。这一个星期对于我而言，是艰辛的、充实的、快乐的。因为在大学里又一次让我感受到自己为了一件事情，如此的努力和投入，我觉得这样的课外作业对于一个人毅力以及计算机应用(包括 MATLAB 的编程能力，word 的编辑排版能力等等)都有很大的益处。

下面谈一谈整个过程所遇到的困难以及自己的心得体会。因为自己对于 MATLAB 软件还是比较熟悉，有一定的基础，选择好算例后，马上进行了潮流计算的编写，遇到的第一个问题的是节点导纳矩阵的求解，由于所求得简单电力系统网络还有对地电容，同时支路还有变压器，于是先将变压器等效为 π 型等效电路，再用循环求解节点导纳矩阵，之后遇到的小问题是迭代收敛的判定依据上程序出了点小问题，节点功率的不平衡量应该加上绝对值再取最大，最后遇到最大的问题应该是 Jacobi 矩阵内各个元素的求解部分程序的编写，开始是还没理解透彻，之后是尝试用元胞数组 cell 来表示分块矩阵，但是失败了，还是选择老老实实用简单的循环，仔细的检查、调试，试了一遍又一遍，程序也编了好几个，最后还是发现并解决问题，顿时心里特别的激动，高兴的难以抑制，整整一个星期空闲时间全部花在了潮流计算的程序上，那一刻真的觉得都值了。最后就是利用 MATLAB 制作 GUI 界面，由于对于 GUI 还是比较的熟悉，同时所做的 GUI 也比较简单，相对于程序而言没有遇到太大的困难。

短短一周的时间，我自己心里清楚所编的程序并不能全部的适用于其他的算例，但是只要经过简单的修改还是能够适用，所做的 GUI 同样也是，如果所采用的程序的通用性很好，完全可以做出比较好的潮流计算 GUI 界面。

在我看来，收获总是很付出成正比的，只有付出了，才有可能得到回报。而对于做事，我认为，做完一件事情，或许是很简单的，但是去做好任何一件事情

确实要付出时间、精力的。

最后，感谢李老师给我们机会去完成这个课外作业，非常感谢！

7. 参考文献

[1] 何仰赞，温增银等，电力系统分析 下册 (第三版)[M]，华中科技大学出版社，2002 03，P29
[2] 郗忠梅，李有安，赵法起等，基于 Matlab 的电力系统的潮流计算[J]，山东农业大学学报(自然科学版)，2010,41(2)，P291-294
[3] 秦景，牛卢璐，路一平等，电力系统潮流计算的最新进展，河北建筑工程学院学报，2008,12,26(4)，P61-62
[4] 汪芳宗，电力系统潮流的并行松弛牛顿计算方法[J]，电力系统自动化，1998,12,22(12)，P16-19
[5] 阳育德，韦化，基于 L1 范数和现代内点理论的电力系统潮流计算，广西大学学报，2007,32,6
[6] 夏道止，电力系统分析(第二版)[M]，中国电力出版社，2010,12
[7] 陈珩，电力系统稳态分析(第三版)[M]，中国电力出版社，2007，06
[8] 罗华飞，MATLAB GUI 设计学习手记[M]，北京航空航天大学出版社，2009,08

8. 附录

8.1 附录一 潮流计算 MATLAB 程序

```
%% ------------------------------------------------------
%   电力系统分析课外作业：基于牛顿-拉夫逊算法的电力系统的潮流计算
%   程序编写者：蒋博宇
%   编写时间：2014年12月13日-2014年12月19日
%   ------------------------------------------------------
clear,clc;
%   ------------------------------------------------------
%   输入节点之间的阻抗，计算生成节点导纳矩阵
%   ------------------------------------------------------
fprintf('请输入节点之间的阻抗：\n')
node_impe_mat=[0 0.1+0.4i 0.3i 0.12+0.5i;0.1+0.4i 0 0 0.08+0.4i;0.3i 0 0
0;0.12+0.5i 0.08+0.4i 0 0]

for n=1:4
    for m=1:4
        if node_impe_mat(n,m)==0
            node_tem_mat(n,m)=0;
        else
            node_tem_mat(n,m)=1/node_impe_mat(n,m);
```

```
            end
        end
end
%将节点1和节点3之间的变压器等效为pi型等效电路，计算其等效后的导纳
turns_ratio_k=1.1;%变压器变比
node_tem_mat_y(1,1)=node_tem_mat(1,3)*(1-turns_ratio_k)/turns_ratio_k;
node_tem_mat_y(3,1)=node_tem_mat(1,3)*(1-turns_ratio_k)/(turns_ratio_k^2);
node_tem_mat(1,3)=node_tem_mat(1,3)/turns_ratio_k;
node_tem_mat(3,1)=node_tem_mat(3,1)/turns_ratio_k;
node_tem_mat_y(2,1)=0;
node_tem_mat_y(4,1)=0;
%求出节点之间对地电容形成的导纳矩阵
fprintf('请输入节点之间的对地电容导纳的原始数据：\n')
node_C_mat=[0 0.01528i 0 0.01920i;0.01528i 0 0 0.01413i;0 0 0 0;0.01920i 0.01413
0 0]
for n=1:4
        for m=1:4
            node_C_mat_y(n,1)=sum(node_C_mat(n,:));
        end
end
%求出导纳矩阵node_Y_mat(n,m)
for n=1:4
        for m=1:4
            if n~=m
                node_Y_mat(n,m)=-node_tem_mat(n,m);
            else
                for k=1:4

node_Y_mat(n,m)=sum(node_tem_mat(n,:))+node_tem_mat_y(n,1)+node_C_mat_y(
n,1);
                end
            end
        end
end
%  求节点导纳矩阵的实部和虚部矩阵
node_G_mat=real(node_Y_mat);
node_B_mat=imag(node_Y_mat);
node_Y_mat;%  节点导纳矩阵
%设置总节点的个数，以及PQ、PV节点的个数
N=4;
NPQ=2;
NPV=1;
Iteration=0;%置迭代次数的初始值为0
Iteration_max=input('\n\n    请输入最大迭代次数后回车(可从零开始)
```

```
Iteration_max=\n');
D_wucha=1e-5;
% 给定PQ、PV节点的初始值
node_power_P=[-0.3 -0.55 0.5];
node_power_Q=[-0.18 -0.13 0];
node_power_V=[0 0 1.10];
% 给定节点电压的初始值
node_vol_e=[1.0 1.0 1.1 1.05];
node_vol_f=[0 0 0 0];
% 利用循环进行牛顿—拉夫逊潮流计算多次迭代
for Iteration=0:Iteration_max
    % 先计算NPQ个PQ节点的修正量D_node_P,D_node_Q;
    %------------------------------------------------------------
    % 给D_node_P,D_node_Q附初始值
    for i=1:NPQ
        D_node_P(i)=node_power_P(i);
        D_node_Q(i)=node_power_Q(i);
    end
    % 计算D_node_P,D_node_Q
    for i=1:NPQ
        for j=1:N

D_node_P(i)=D_node_P(i)-node_vol_e(i)*node_G_mat(i,j)*node_vol_e(j)+node_vol
_e(i)*node_B_mat(i,j)*node_vol_f(j)-node_vol_f(i)*node_G_mat(i,j)*node_vol_f(i)-
node_vol_f(i)*node_B_mat(i,j)*node_vol_e(j);

D_node_Q(i)=D_node_Q(i)-node_vol_f(i)*node_G_mat(i,j)*node_vol_e(j)+node_vol
_f(i)*node_B_mat(i,j)*node_vol_f(j)+node_vol_e(i)*node_G_mat(i,j)*node_vol_f(i)
+node_vol_e(i)*node_B_mat(i,j)*node_vol_e(j);
        end
    end

    % 再计算NPV个PV节点的D_node_P、D_node_V2
    %------------------------------------------------------------
    for i=(NPQ+1):(N-1)
        D_node_P(i)=node_power_P(i);
    end
    for i=(NPQ+1):(N-1)
        for j=1:N

D_node_P(i)=D_node_P(i)-node_vol_e(i)*node_G_mat(i,j)*node_vol_e(j)+node_vol
_e(i)*node_B_mat(i,j)*node_vol_f(j)-node_vol_f(i)*node_G_mat(i,j)*node_vol_f(i)-
node_vol_f(i)*node_B_mat(i,j)*node_vol_e(j);
            D_node_V2(i)=node_power_V(i)^2-node_vol_e(i)^2-node_vol_f(i)^2;
```

```
        end
    end

    % 形成D_node_W=[D_node_P1,D_node_Q1,...] 矩阵
    %-----------------------------------------------------------
    % 先将D_node_P间隔的输入D_node_W中
    a=1;
    for i=1:(N-1)
        D_node_W(a)=D_node_P(i);
        a=a+2;
    end
    % 再将D_node_Q间隔的输入D_node_W中
    a=2;
    for i=1:NPQ
        D_node_W(a)=D_node_Q(i);
        a=a+2;
    end
    % 最后将 D_node_V输入D_node_W中
    a=2*NPQ+2;
    for i=(NPQ+1):(NPQ+NPV)
        D_node_W(a)=D_node_V2(i);
        a=a+2;
    end
    % 判断是否满足小于的容许误差，如果满足误差条件则跳出循环，否则继续运行
    %-----------------------------------------------------------
    if max(abs(D_node_W))<D_wucha
        fprintf('\n 迭代是收敛的，第%d次迭代后收敛，迭代终止。\n',Iteration-1);
        break;
    end
    % 计算Jacbi矩阵中的各个元素
    %-----------------------------------------------------------
    Jacbi_mat=zeros(N-1);
    for i=1:N-1
        for j=1:N-1
            if i<=NPQ            %先计算Jacbi矩阵中PQ节点的各个元素
                if i==j

Jacbi_mat(2*i-1,2*j-1)=-node_G_mat(i,i)*node_vol_e(i)-node_B_mat(i,i)*node_vol_f(i);

Jacbi_mat(2*i-1,2*j)=node_B_mat(i,i)*node_vol_e(i)-node_G_mat(i,i)*node_vol_f(i);
```

```
Jacbi_mat(2*i,2*j-1)=node_B_mat(i,i)*node_vol_e(i)-node_G_mat(i,i)*node_vol_f(i)
;

Jacbi_mat(2*i,2*j)=node_G_mat(i,i)*node_vol_e(i)+node_B_mat(i,i)*node_vol_f(i);
                    elseif i~=j

Jacbi_mat(2*i-1,2*j-1)=-node_G_mat(i,j)*node_vol_e(i)-node_B_mat(i,j)*node_vol_
f(i);

Jacbi_mat(2*i-1,2*j)=node_B_mat(i,j)*node_vol_e(i)-node_G_mat(i,j)*node_vol_f(i)
;

Jacbi_mat(2*i,2*j-1)=node_B_mat(i,j)*node_vol_e(i)-node_G_mat(i,j)*node_vol_f(i)
;

Jacbi_mat(2*i,2*j)=node_G_mat(i,j)*node_vol_e(i)+node_B_mat(i,j)*node_vol_f(i);
                    end
                end
                if i>NPQ            %再计算Jacbi矩阵中PV节点的各个元素
                    if i==j

Jacbi_mat(2*i-1,2*j-1)=-node_G_mat(i,i)*node_vol_e(i)-node_B_mat(i,i)*node_vol_
f(i);

Jacbi_mat(2*i-1,2*j)=node_B_mat(i,i)*node_vol_e(i)-node_G_mat(i,i)*node_vol_f(i)
;
                        Jacbi_mat(2*i,2*j-1)=-2*node_vol_e(i)
                        Jacbi_mat(2*i,2*j)=-2*node_vol_f(i)
                    elseif i~=j

Jacbi_mat(2*i-1,2*j-1)=-node_G_mat(i,j)*node_vol_e(i)-node_B_mat(i,j)*node_vol_
f(i);

Jacbi_mat(2*i-1,2*j)=node_B_mat(i,j)*node_vol_e(i)-node_G_mat(i,j)*node_vol_f(i)
;
                        Jacbi_mat(2*i,2*j-1)=0;
                        Jacbi_mat(2*i,2*j)=0;
                    end
                end
            end
        end
        for i=1:NPQ
            for k=1:N
```

```
Jacbi_mat(2*i-1,2*j-1)=Jacbi_mat(2*i-1,2*j-1)-node_G_mat(i,k)*node_vol_e(k)+nod
e_B_mat(i,k)*node_vol_f(k);

Jacbi_mat(2*i-1,2*j)=Jacbi_mat(2*i-1,2*j)-node_G_mat(i,k)*node_vol_f(k)-node_B
_mat(i,k)*node_vol_e(k);

Jacbi_mat(2*i,2*j-1)=Jacbi_mat(2*i,2*j-1)+node_G_mat(i,k)*node_vol_f(k)+node_B
_mat(i,k)*node_vol_e(k);

Jacbi_mat(2*i,2*j)=Jacbi_mat(2*i,2*j)-node_G_mat(i,k)*node_vol_e(k)+node_B_ma
t(i,k)*node_vol_f(k);
            end
        end
        % 求出D_node_V矩阵
        D_node_V=-Jacbi_mat\D_node_W';
        % 修正、刷新node_vol_e(i)、node_vol_f(i)
        for i=1:N-1
            node_vol_e(i)=node_vol_e(i)+D_node_V(2*i-1);
            node_vol_f(i)=node_vol_f(i)+D_node_V(2*i);
            %将节点电压转换为复数形式显示
            node_V(i)=complex(node_vol_e(i),node_vol_f(i));
        end
            node_V(N)=complex(node_vol_e(N),node_vol_f(N));
            %format long g; %将数据显示为长整型科学计数
            fprintf('\n   第%d次迭代（共%d个节点）',Iteration,N);
            node_V
         % 求节点电压的幅值和相角
      for i=1:N
            node_yxz_V(i)=sqrt(node_vol_e(i)^2+node_vol_f(i)^2);
            node_yxz_V_angle(i)=atan(node_vol_f(i)./node_vol_e(i))*180./pi;
      end
            node_yxz_V;
            node_1(Iteration+1)=node_yxz_V(1);
            node_2(Iteration+1)=node_yxz_V(2);
            node_3(Iteration+1)=node_yxz_V(3);
            node_4(Iteration+1)=node_yxz_V(4);
end
        %-------------------------------------------------------------------
        % 计算平衡节点的功率
        node_power_P(N)=0;
        node_power_Q(N)=0;
            for i=N
                for j=1:N
                node_power_P(i)=
```

```
node_power_P(i)+node_vol_e(i)*(node_G_mat(i,j)*node_vol_e(i)-node_B_mat(i,j)*
node_vol_f(i));

node_power_Q(i)=node_power_Q(i)+node_vol_e(i)*(-node_G_mat(i,j)*node_vol_f(i
)-node_B_mat(i,j)*node_vol_e(i));
                end
            end
    %   将各个节点的功率转化为复数形式
    for i=1:N
            node_power_S(i)=node_power_P(i)+node_power_Q(i)*sqrt(-1);
    end

            node_power_P, node_power_Q ,node_power_S

    % 计算全部的线路功率
    for i=1:N
            for j=1:N

line_S(i,j)=conj(node_V(i))*conj(node_V(i))-conj(node_V(j))*conj(node_tem_mat(i,j
));
            end
    end
         line_S
    % 求节点电压的幅值和相角
    for i=1:N
             node_yxz_V(i)=sqrt(node_vol_e(i)^2+node_vol_f(i)^2);
             node_yxz_V_angle(i)=atan(node_vol_f(i)./node_vol_e(i))*180./pi;
    end
    %-----------------------------------------------------------------
    % 绘制节点电压幅值与迭代次数的曲线
    figure(1);
    set(gcf,'Numbertitle','off','name','节点电压幅值与迭代次数曲线');
    node_plot=0:1:5;
    subplot(2,2,1)
    plot(node_plot,node_1,'r');
    xlabel('迭代次数');
    ylabel('节点电压的幅值');
    title('节点1');
    subplot(2,2,2)
    plot(node_plot,node_2,'r');
    xlabel('迭代次数');
    ylabel('节点电压的幅值');
    title('节点2');
    subplot(2,2,3)
```

```
plot(node_plot,node_3,'r');
xlabel('迭代次数');
ylabel('节点电压的幅值');
title('节点3');
subplot(2,2,4)
plot(node_plot,node_4,'r');
xlabel('迭代次数');
ylabel('节点电压的幅值');
title('节点4');
```