研究生学位课：

现代电力系统分析

任课教师：王继东

参考书目：

1、诸骏伟，电力系统分析（上册），水利电力出版社，1995

2、夏道止，电力系统分析（下册），水利电力出版社，1995

3、王锡凡，现代电力系统分析，科学出版社，2003

第一章　电力系统潮流计算

第一节　概　　述

作为研究电力系统稳态运行情况的一种基本电气计算，电力系统常规潮流计算的任务是根据给定的网络结构及运行条件（网络结构包括线路、变电站、电源点的位置等；运行条件是指负荷的大小及电源出力等），求出整个网络的运行状态，其中包括各母线的电压、网络中的功率分布以及功率损耗等等。

✓ 潮流计算的应用分为离线和在线应用两类。

离线应用主要有：

- 规划及运行方式研究
- 静态及暂态稳定计算
- 故障分析及优化计算

在线应用主要有：

- 随着现代化的调度控制中心的建立，为了对电力系统进行实时安全监控，需要根据实时数据库所提供的信息，随时判断系统当前的运行状态并对预想事故进行安全分析，这就需要进行广泛的潮流计算，并且对计算速度等还提出了更高的要求，从而产生了潮流的在线计算。

- ✓ 由上可见，潮流计算是电力系统中应用最为广泛、最基本和最重要的一种电气计算
- ✓ 潮流计算问题在数学上一般是属于多元非线性代数方程组的求解问题，必须采用迭代计算方法。
- ✓ 自从20世纪50年代中期开始利用电子计算机进行潮流计算以来，潮流计算是电力系统各种问题中投入研究力量最多的领域之一，出现了大量的研究成果。这些成果：
  - ➢ 开拓了各种特殊性质的潮流计算问题
  - ➢ 更多的是属于为了提高计算性能而陆续提出的各种具体算法。

- 对于一个潮流算法，其基本要求可归纳成以下四个方面

  (1) 计算速度；

  (2) 计算机内存占用量；

  (3) 算法的收敛可靠性；

  (4) 程序设计的方便性以及算法扩充移植等的通用灵活性。

- 这四点要求也成为本章后面评价各种潮流算法性能时所依据的主要标准。

- ✓本章在对潮流计算问题的数学模型进行简单的回顾以后，将首先转入三种最基本的潮流算法：
  - ➢高斯—塞德尔法
  - ➢牛顿法
  - ➢快速解耦法的讨论
- ✓这三种算法的基本原理在大学本科的《电力系统分析》教材中已作过介绍，但鉴于这些方法的重要性，将在大学本科《电力系统分析》教材的基础上作进一步的讨论。

- ✓牛顿法的特点是将非线性方程线性化。20世纪70年代后期，有人提出采用更精确的模型，即将泰勒级数的高阶项也包括进来，希望以此提高算法的性能，这便产生了保留非线性的潮流算法。
- ✓为了解决病态潮流计算，出现了将潮流计算表示为一个无约束非线性规划问题的模型，并称之为最小化潮流计算法。本章第六、七两节将分别介绍这两类算法，

✓一些实际用于生产的潮流程序往往在上述基本潮流的框架内再加入模拟实际系统运行控制特点的自动调整计算功能，如潮流控制，分接头调整等，这部分内容将在本章第八节中予以介绍。

✓20世纪60年代中期，结合电力系统经济调度工作的开展，针对经典的经济调度方法的不足，开辟了一个新的研究领域，称之为最优潮流问题。这种以非线性规划作为计算模型的潮流问题能够统筹兼顾电力系统的经济性、安全性和电能质量，因而受到很大的重视，发展很快，其应用领域正在不断扩大。我们将在本章第九节中以较大的篇幅讨论这种潮流问题。

- 和交流输电比较，直流输电具有不少固有的特点。20世纪70年代以后，随着晶闸管（可控硅）换流器的问世，促进了直流输电的迅速发展，一批批新的线路正在建设或已经投运，我国也已经建成了葛洲坝--上海、天生桥—广东、三峡--广东等高压直流±500kV输电工程，因此研究交直流系统的潮流计算就成为十分必要。
- 最后，本章将简单介绍几种特殊用途的潮流计算问题。
  - 直流潮流
  - 随机潮流
  - 三相潮流

第二节 潮流计算问题的数学模型

- ✓电力系统是由发电机、变压器、输电线路及负荷等组成，其中发电机及负荷是非线性元件，但在进行潮流计算时，一般可用接在相应节点上的一个电流注入量代表，因此潮流计算所用的电力网络系由变压器、输电线路、电容器、电抗器等静止线性元件所构成，并用集中参数表示的串联或并联等值支路来模拟。

✓结合电力系统的特点，对这样的线性网络进行分析，普遍采用的是节点法，节点电压与节点电流之间的关系为：

$$\dot{I} = Y\ \dot{U}$$

$$\dot{U} = Z\ \dot{I}$$

✓其展开式分别为：

$$\dot{I}_i = \sum_{j=1}^{n} Y_{ij}\ \dot{U}_j \qquad (i = 1, 2, \cdots, n)$$

$$\dot{U}_i = \sum_{j=1}^{n} Z_{ij}\ \dot{I}_j \qquad (i = 1, 2, \cdots, n)$$

✓但是在工程实际中，已知的节点注入量往往不是节点电流而是节点功率，为此必须应用联系节点电流和节点功率的关系式

$$\dot{I}_i = \frac{P_i - \mathrm{j}Q_i}{\overset{*}{U}_i} \quad (i = 1, 2, \cdots, n)$$

✓将上式代入电压、电流展开式，

$$\dot{I}_i = \sum_{j=1}^{n} Y_{ij} \dot{U}_j \quad (i = 1, 2, \cdots, n)$$

$$\dot{U}_i = \sum_{j=1}^{n} Z_{ij} \dot{I}_j \quad (i = 1, 2, \cdots, n)$$

$$\sum_{j=1}^{n} Y_{ij}\dot{U}_i = \frac{P_i - jQ_i}{\dot{U}_i^*} \qquad (i = 1,2,\cdots, n)$$

或
$$\dot{U}_i = \sum_{j=1}^{n} Z_{ij}\frac{P_j - jQ_j}{\dot{U}_j^*} \qquad (i = 1,2,\cdots, n)$$

✓ 这就是潮流计算问题最基本的方程式，是一个以节点电压为变量的非线性代数方程组。由此可见，采用节点功率作为节点注入量是造成方程组呈非线性的根本原因。由于方程组为非线性的，因此必须采用数值计算方法、通过迭代来求解。而根据在计算中对这个方程组的不同应用和处理，就形成了不同的潮流算法。

- 对于电力系统中的每个节点，要确定其运行状态，需要有四个变量：
  - 有功注入
  - 无功注入
  - 电压模值
  - 电压相角
- n个节点总共有4n个运行变量要确定。n个复数方程式，如果将实部与虚部分开，则形成2n个实数方程式，由此仅可以解得2n个未知运行变量。为此在计算潮流以前，必须将另外2n个变量作为已知量而预先给以指定。也即对每个节点，要给定其两个变量的值作为已知条件，而另两个变量作为待求量。

- ✓ 按照电力系统的实际运行条件，根据预先给定的变量的不同，电力系统中的节点又可分成
  - ➢ PQ节点
  - ➢ PV节点
  - ➢ 平衡节点
- ✓ 对应于这些节点，分别对其注入的有功、无功功率，有功功率及电压模值以及电压模值和相角加以指定；并且对平衡节点来说，其电压相角一般作为系统电压相角的基准：即 $\theta = 0$

✓交流电力系统中的复数电压变量可以用两种坐标形式来表示：

$$\dot{U}_i = U_i e^{\mathrm{j}\theta_i}$$

或

$$\dot{U}_i = e_i + \mathrm{j}f_i$$

✓复数导纳为

$$Y_{ij} = G_{ij} + \mathrm{j}B_{ij}$$

✓将上三式代入以导纳矩阵为基础的潮流方程式，并将实部与虚部分开，可得到两种形式的潮流方程。

$$\sum_{j=1}^{n} Y_{ij}\dot{U}_i = \frac{P_i - \mathrm{j}Q_i}{\dot{U}_i^*} \quad (i = 1,2,\cdots, n)$$

✓潮流方程的直角坐标形式为：

$$P_i = e_i \sum_{j \in i} \left(G_{ij} e_j - B_{ij} f_j \right) +$$

$$f_i \sum_{j \in i} \left(G_{ij} f_j + B_{ij} e_j \right) \quad (i = 1, 2, \cdots, n)$$

$$Q_i = f_i \sum_{j \in i} \left(G_{ij} e_j - B_{ij} f_j \right) -$$

$$e_i \sum_{j \in i} \left(G_{ij} f_j + B_{ij} e_j \right) \quad (i = 1, 2, \cdots, n)$$

✓ 潮流方程的极坐标形式为：

$$P_i = U_i \sum_{j \in i} U_j \left(G_{ij} \cos \theta_{ij} + B_{ij} \sin \theta_{ij} \right)$$

$$(i = 1, 2, \cdots, n)$$

$$Q_i = U_i \sum_{j \in i} U_j \left(G_{ij} \sin \theta_{ij} - B_{ij} \cos \theta_{ij} \right)$$

$$(i = 1, 2, \cdots, n)$$

✓ 以上各式中$j \in i$表示$\sum$号后的标号为j的节点必须直接和节点i相联，并包括j=i的情况。这两种形式的潮流方程通称为节点功率方程，是牛顿—拉夫逊法等潮流算法所采用的主要数学模型。

- ✓ 对于以上潮流方程中的有关运行变量，还可以按其性质的不同再加以分类，这对于进行例如灵敏度分析以及最优潮流的研究等，都是比较方便的。

- ✓ 每个节点的**注入功率**是该节点的**电源输入功率** P_{Gi}、Q_{Gi}和**负荷需求功率**P_{Li}，Q_{Li}的代数和。
- ✓ 负荷需求的功率取决于用户，是无法控制的，所以称之为**不可控变量或扰动变量**。
- ✓ 而某个电源所发的有功、无功功率则是可以由运行人员控制或改变的变量，是**自变量或称为控制变量**。
- ✓ 至于各个节点的电压模值或相角，则属于随着控制变量的改变而变化的**因变量或状态变量**。
- ✓ 当系统中各个节点的电压模值及相角都知道以后，则整个系统的**运行状态**也就完全确定了。

✓若以p, u, x分别表示扰动变量、控制变量、状态变量，则潮流方程可以用更简洁的方式表示为：

$$f\left(x, u, p\right) = 0$$

✓根据上式，潮流计算的含义就是针对某个扰动变量p，根据给定的控制变量u，求出相应的状态变量x。

第三节　高斯—塞德尔法

✓以导纳矩阵为基础，并应用高斯—塞德尔迭代的算法是在电力系统中最早得到应用的潮流计算方法。

$$\dot{U}_i = \frac{1}{Y_{ii}}\left[\frac{P_i - jQ_i}{\dot{U}_i^*} - \sum_{\substack{j=1 \\ j\neq i}}^{n} Y_{ij}\dot{U}_j\right] \qquad (i = 2,3,\cdots,n)$$

- 高斯--塞德尔迭代格式

$$\dot{U}_i^{(k+1)} = \frac{1}{Y_{ii}}\left[\frac{P_i - jQ_i}{\dot{U}_i^{*(k)}} - Y_{i1}\dot{U}_1 - \left(\sum_{j=2}^{i-1} Y_{ij}\dot{U}_j^{(k+1)} + \sum_{j=i+1}^{n} Y_{ij}\dot{U}_j^{(k)}\right)\right]$$

$$(i = 2, 3, \cdots, n)$$

- 从一组假定的电压初值出发，依次进行迭代计算，迭代收敛的判据是

$$\max_i \left|\dot{U}_i^{(k+1)} - \dot{U}_i^{(k)}\right| < \varepsilon$$

✓优点：原理简单，程序设计十分容易。导纳矩阵是一个对称且高度稀疏的矩阵，因此占用内存非常节省。就每次迭代所需的计算量而言，是各种潮流算法中最小的，并且和网络所包含的节点数成正比关系。

✓缺点：

✓本算法的主要缺点是收敛速度很慢。

✓因为根据迭代计算公式，各节点电压在数学上是松散耦合的，也即经过一次迭代，每个节点电压值的改进只能影响到和这个节点直接相联的少数几个节点电压的修正，所以节点电压向最后收敛解点的接近非常缓慢。

✓算法达到收敛所需的迭代次数与所计算网络的节点数目有密切关系，迭代次数将随着所计算网络节点数的增加而直接上升，从而导致了计算量大急剧增加。因此在用于较大规模电力系统的潮流计算时，速度显得非常缓慢。

✓为了提高算法收敛速度，常用的一种方法是在迭代过程中加入加速因子

$$\dot{U}_i^{(k+1)} = \dot{U}_i^{(k)} + \alpha(\dot{U}_i'^{(k+1)} - \dot{U}_i^{(k)})$$

✓ $\dot{U}_i'^{(k+1)}$ 是通过迭代计算求得的节点 i 电压的第（k+1）次迭代值；

- ✓病态条件系统，计算往往会发生收敛困难
  - ✓节点间相位角差很大的重负荷系统；
  - ✓包含有负电抗支路(如某些三绕组变压器或线路串联电容等)的系统；
  - ✓具有较长的辐射形线路的系统；
  - ✓长线路与短线路接在同一节点上，而且长短线路的长度比值又很大的系统。
- ✓ 此外，平衡节点所在位置的不同选择，也会影响到收敛性能。
- ✓ 目前高斯—塞德尔法已很少使用

第四节　牛顿—拉夫逊法

一、牛顿—拉夫逊法的一般概念

牛顿—拉夫逊法在数学上是求解非线性代数方程式的有效方法。其要点是把非线性方程式的求解过程变成反复地对相应的线性方程式进行求解的过程，即通常所称的逐次线性化过程。

✓ 对于非线性代数方程组

$$f(x)=0$$

即

$$f_i(x_1, x_2, \ldots, x_n)=0 \quad (i=1,2,\ldots n)$$

✓ 在待求量x的某一个初始估计值，$x^{(0)}$附近，将上式展开成泰勒级数并略去二阶及以上的高阶项，得到如下的经线性化的方程组

$$f\ (x^{(0)})+ f'(x^{(0)})\Delta x^{(0)} = 0$$

✓ 上式称之为牛顿法的修正方程式。

✓由此可以求得第一次迭代的修正量

$$\Delta x^{(0)} = -[f'(x^{(0)})]^{-1} f(x^{(0)})$$

✓将 $\Delta x^{(0)}$和$x^{(0)}$ 相加，得到变量的第一次改进值$x^{(1)}$。接着就从$x^{(1)}$出发，重复上述计算过程。因此从一定的初值$x^{(0)}$出发，应用牛顿法求解的迭代格式为：

$$f'(x^{(k)})\Delta x^{(k)} = -f(x^{(k)})$$

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$$

$$f'(x^{(k)})\Delta x^{(k)} = -f(x^{(k)})$$

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$$

- ✓ 上两式中：f’(x)是函数f(x)对于变量x的一阶偏导数矩阵，即雅可比矩阵J；k为迭代次数。
- ✓ 由上两式可见，牛顿法的核心便是反复形成并求解修正方程式。
- ✓ 牛顿法当初始估计值$x^{(0)}$和方程的精确解足够接近时，收敛速度非常快，具有平方收敛特性。

二、牛顿潮流算法的修正方程式

- ✓ 在将牛顿法用于求解电力系统潮流计算问题时，由于所采用$f(x)$的数学表达式以及复数电压变量采用的坐标形式的不同，可以形成牛顿潮流算法的不同形式。
- ✓ 以下讨论用得最为广泛的$f(x)$采用功率方程式模型，而电压变量则分别采用极坐标和直角坐标的两种形式。

（一）极坐标形式

✓令 $\dot{U}_i = U_i \angle \theta_i$ 则采用极坐标形式的潮流方程是：

对每个PQ节点及PV节点

$$P_i^S - U_i \sum_{j \in i} U_j \left(G_{ij} \cos \theta_{ij} + B_{ij} \sin \theta_{ij} \right) = \Delta P_i = 0$$

对每个PQ节点

$$Q_i^S - U_i \sum_{j \in i} U_j \left(G_{ij} \sin \theta_{ij} - B_{ij} \cos \theta_{ij} \right) = \Delta Q_i = 0$$

✓将上述方程式在某个近似解附近用泰勒级数展开，并略去二阶及以上的高阶项后，得到以矩阵形式表示的修正方程式为：

$$\begin{matrix} n-1 \\ n-m-1 \end{matrix}\begin{bmatrix} \Delta P \\ \cdots \\ \Delta Q \end{bmatrix} = -\begin{bmatrix} H & N \\ \cdots & \cdots \\ M & L \end{bmatrix}\begin{bmatrix} \Delta\theta \\ \cdots \\ \Delta U/U \end{bmatrix}\begin{matrix} n-1 \\ n-m-1 \end{matrix}$$

✓式中：n为节点总数；m为PV节点数，雅可比矩阵是(2n-m-2)阶非奇异方阵。

(二)直角坐标形式

令 $\dot{U}_i = e_i + jf_i$ 在这里，潮流方程的组成与上不同，对每个节点，都有二个方程式，所以在不计入平衡节点方程式的情况下，总共有$2(n-1)$个方程式。

✓对每个PQ节点有：

$$P_i^{s} - \sum_{j \in i} [e_i (G_{ij} e_j - B_{ij} f_j) + f_i (G_{ij} f_j + B_{ij} e_j)] = \Delta P_i = 0$$

$$Q_i^{s} - \sum_{j \in i} [f_i (G_{ij} e_j - B_{ij} f_j) - e_i (G_{ij} f_j + B_{ij} e_j)] = \Delta Q_i = 0$$

✓对每个PV节点，除了有与上式相同的有功功率方程式之外，还有

$$(U_i^s)^2-(e_i^2+f_i^2)=\Delta U_i^2=0$$

✓采用直角坐标形式的修正方程式为

$$\begin{matrix}n-1\\ n-m-1\\ m\end{matrix}\begin{bmatrix}\Delta P\\ \Delta Q\\ \Delta U^2\end{bmatrix}=-\left[\begin{array}{c|c}H & N\\ \hline M & L\\ \hline R & S\end{array}\right]\begin{matrix}\begin{bmatrix}\Delta e\\ \Delta f\end{bmatrix}\begin{matrix}n-1\\ n-1\end{matrix}\end{matrix}$$

✓ 仔细分析以上两种类型的修正方程式，可以看出两者具有以下的共同特点。

(1) 修正方程式的数目分别为2(n-1)-m及2(n-1)个，在PV节点所占比例不大时，两者的方程式数目基本接近2(n-1)个。

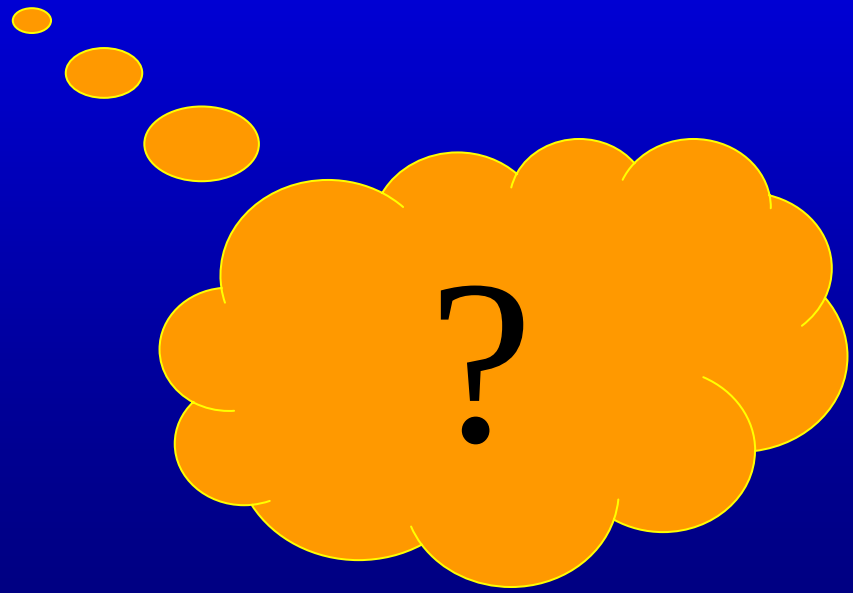
(2) 雅可比矩阵的元素都是节点电压的函数，每次迭代，雅可比矩阵都需要重新形成。

(3)分析雅可比矩阵的非对角元素的表示式可见，某个非对角元素是否为零决定于相应的节点导纳矩阵元素Y_{ij}是否为零。因此如将修正方程式按节点号的次序排列，并将雅可比矩阵分块，把每个2X2阶子阵作为分块矩阵的元素，则按节点号顺序而构成的分块雅可比矩阵将和节点导纳矩阵具有同样的稀疏结构，是一个高度稀疏的矩阵。

(4)和节点导纳矩阵具有相同稀疏结构的分块雅可比矩阵在位置上对称，但雅可比矩阵不是对称阵。

三、修正方程式的处理和求解

- ✓ 在本节的开头就已提到，牛顿算法的核心就是反复形成并求解修正方程式。
- ✓ 因此如何有效地处理修正方程式就成为提高牛顿法潮流程序计算速度并降低内存需量的关键所在。



- ✓从算法的发展过程来看，在20世纪50年代末就已经提出了牛顿法潮流的雏形。
  - ➢先是用迭代法求解修正方程式，但遇到迭代法本身不收敛的问题。
  - ➢用高斯消去法等直接法求解，但如前所分析，修正方程式的数目在$2(n-1)$左右，如果不利用雅可比矩阵的稀疏特性，当网络节点数增加为N倍，存储雅可比矩阵的内存量将正比于N^2倍，利用直接法求解修正方程的计算量将正比于N^3倍地增长。
- ✓这就限制了牛顿法潮流程序的解题规模，从而使得这种方法的推广应用一度止步不前。

- ✓其后正是人们注意到了雅可比矩阵高度稀疏的特点，求解修正方程式时采用了稀疏程序设计技巧，并且发展了一套在消元过程中旨在尽量保持其稀疏性、以减少内存需量并提高计算速度的有效方法(即著名的最优顺序消去法)，才使牛顿法真正得到了突破，因而在20世纪60年代中期以后被普遍采用。

✓ 结合修正方程式的求解，目前在实用的牛顿法潮流程序中所包含的程序特点主要有以下三个方面，这些程序特点对牛顿法潮流程序性能的提高起着决定性的作用。

(1) 对于稀疏矩阵，在计算机中以“压缩”方式只储存其非零元素，且只有非零元素才参加运算。

(2)修正方程式的求解过程，采用对包括了修正方程常数项的增广矩阵以按行消去而不是传统的按列消去的方式进行消元运算。由于消元运算系按行进行，因此可以不需先形成整个增广矩阵，然后进行消元运算，而是采取边形成、边消元、边存储的方式，即每形成增广矩阵的一行便马上进行消元，并且消元结束后便随即将结果送内存存储。

✓图1-1是增广矩阵按行消元的示意图，图中表示了五阶增广矩阵的前四行，其中1-3行已完成了消元运算且已经存放在内存中，接着要进行的是第四行的消元运算，即消去对角元以左的三个元素。在具体的程序中，待消行是放在一个专用的工作数组中进行消元运算的。

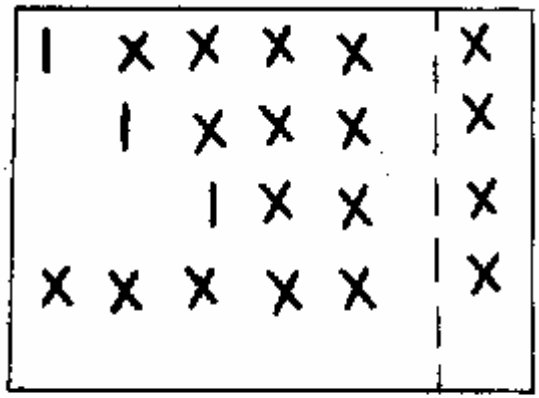


图 1-1　增广阵按行消元示意图

✓这种按行消元做法的好处：

- 是对于消元过程中新注入的非零元素，当采用“压缩”存储方式时，可以方便地按序送入内存，不需要预留它们的存放位置。
- 特别值得注意的是由于不必一次形成整个雅可比矩阵，且常数项的消元运算已和矩阵的消元过程同时进行，因此这种牛顿潮流算法求解修正方程式时，所需的矩阵存储量只是消元运算结束时所得到的用以进行回代的上三角矩阵而已。

(3) 消元的最优顺序或节点编号优化

经过消元运算得到的上三角矩阵一般仍属稀疏阵，但由于消元过程中在原来是零元素的位置上有新元素注入，使得它的稀疏度比原来雅可比矩阵的上三角有所降低。但分析表明，注入元素的多少和消元的顺序或节点编号有关。节点编号优化的作用即在于找到一种网络节点的重新编号方案，使得按此构成的节点导纳矩阵以及和它相应的雅可比矩阵在高斯消元或三角分解过程中出现的注入元素数目能大大减少。节点编号优化通常有三种方法：

- **静态法**——按各节点静态连接支路数的多少顺序编号；
- **半动态法**——按各节点动态连接支路数的多少顺序编号；
- **动态法**——按各节点动态增加支路数的多少顺序编号。

✓三种节点编号优化方法：动态法效果最好，但优化本身所需计算量也最多，而静态法则反之。对于牛顿法潮流计算来说，一般认为，采用**半动态法**似乎是较好的选择。

四、牛顿潮流算法的性能和特点

- ✓ 牛顿潮流算法突出的优点是**收敛速度快**，若选择到一个较好的**初值**，算法将具有**平方收敛特性**，一般迭代4—5次便可以收敛到一个非常精确的解。而且其迭代次数与所计算网络的规模基本无关。
- ✓ 牛顿法也具有良好的**收敛可靠性**，对于上节中提到的对以节点导纳矩阵为基础的高斯—塞德尔法呈**病态**的系统，牛顿法均能可靠地收敛。

- ✓牛顿法所需的内存量及每次迭代所需时间均较前述的高斯—塞德尔法为多，并与程序设计技巧有密切关系。
- ✓牛顿法的可靠收敛取决于有一个良好的启动初值。如果初值选择不当，算法有可能根本不收敛或收敛到一个无法运行的解点上。
- ✓对于正常运行的系统，各节点电压一般均在额定值附近，偏移不会太大，并且各节点间的相位角差也不大，所以对各节点可以采用统一的电压初值(也称为“平直电压”)，

✓“平直电压”法假定：

$$U_i^{(0)} = 1 \qquad \theta_i^{(0)} = 0^\circ$$

或

$$e_i^{(0)} = 1 \qquad f_i^{(0)} = 0 \qquad (i = 1,2,\ldots,n; i \neq s)$$

✓这样一般能得到满意的结果。但若系统因无功紧张或其它原因导致电压质量很差或有重载线路而节点间角差很大时，仍用上述初始电压就有可能出现问题。

- ✓解决这个问题的办法可以先用上一节的高斯—塞德尔法迭代$1\sim2$次；以此迭代结果作为牛顿法的初值。
- ✓也可以先用直流法潮流求解一次以求得一个较好的角度初值，然后转入牛顿法迭代。

第五节　快速解耦法

- ✓ 随着电力系统规模的日益扩大以及在线计算要求的提出，为了改进牛顿法在内存占用量及计算速度方面的不足，人们开始注意到电力系统有功及无功潮流间仅存在较弱联系的这一固有物理特性，于是产生了一类具有有功、无功解耦迭代计算特点的算法。
- ✓ 在1974年由Scott B.提出的快速解耦法(Fast Decoupled Load Flow，简写为FDLF)是在广泛的数值试验基础上挑选出来的最为成功的一个算法，它无论在内存占用量以及计算速度方面，都比牛顿法有了较大的改进，从而成为当前国内外最优先使用的算法。

(一)快速解耦法基本原理

- ✓ Scott B.提出的**快速解耦法**是脱胎于极坐标形式的牛顿潮流算法，经过演化而得到的。以下对演化过程作一个简短的复习。
- ✓ 由于交流高压电网中输电线路等元件的x 》 r，因此电力系统呈现了这样的**物理特性**：
  - ➢ 即有功功率的变化主要决定于电压相位角的变化，
  - ➢ 无功功率的变化则主要决定于电压模值的变化。
- ✓ 这个特性反映在**牛顿法修正方程式**雅可比矩阵的元素上，是N及M二个子块元素的数值相对于H、L二个子块的元素要小得多。

$$\begin{matrix} n-1 \\ n-m-1 \end{matrix}\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = -\begin{bmatrix} H & N \\ M & L \end{bmatrix}\begin{bmatrix} \Delta\theta \\ \Delta U/U \end{bmatrix}\begin{matrix} n-1 \\ n-m-1 \end{matrix}$$

✓作为简化的第一步，可以将它们略去不计，于是得到如下两个已经解耦的方程组：

$$\Delta P = -H\Delta\theta$$

$$\Delta Q = -L(\Delta U / U)$$

✓这一步简化将原来2n-m-2阶的方程组化为一个n-1及一个n-m-1阶的较小的方程组，显著地节省了内存需量和解题时间。但H及L的元素仍然是节点电压的函数且不对称.

✓算法的进一步简化是基于在实际的高压电力系统中，下列的假设一般都能成：

- 线路两端的相角差不大(小于10^{o}--20^{o}，而且 $|G_{ij}|<<|B_{ij}|$，于是可以认为：

$$\cos\theta_{ij}\approx 1;\ G_{ij}\theta_{ij}<<B_{ij}$$

- 与节点无功功率相对应的导纳 Q_i/U_i^2 通常远小于节点的自导纳B_{ii}，也即

$$Q_i<<U_i^2 B_{ii}$$

✓计及上两式后，H 及 L 各元素的表示式可以简化为：

$$H_{ij} = U_i U_j B_{ij}$$

$$L_{ij} = U_i U_j B_{ij}$$

✓于是 H 及 L 可表示成

$$H = UB'U$$

$$L = UB''U$$

$$H = UB'U$$

$$L = UB''U$$

- 式中：U 是由各节点电压模值组成的对角阵。由于PV节点的存在，B'及B''的阶数将不同，分别为n-1及n-m-1阶。

$$\Delta P = -H\Delta\theta$$

$$\Delta Q = -L(\Delta U / U)$$

- 将上式代入式并加以整理，可得：

$$\Delta P / U = -B'(U\Delta\theta)$$

$$\Delta Q / U = -B''\Delta U$$

- 通过这一步简化，上述二式中的系数矩阵B'及B''由节点导纳矩阵的虚部所组成，从而是一个常数且对称的矩阵。

✓ 为了加速收敛，目前通用的快速解耦法又对B'及B''的构成作了下列进一步修改。

(1) 在形成B'时，略去那些主要影响无功功率和电压模值，而对有功功率及电压角度关系很少的因素。这些因素包括输电线路的充电电容以及变压器非标准变比。

(2) 为了减少在迭代过程中无功功率及节点电压模值对有功迭代的影响，将有功方程式右端U的各元素均置为标么值1.0，也即令U作为单位阵。

(3) 在计算B'时，略去串联元件的电阻。

✓于是，目前通用的快速解耦潮流算法的修正方程式可写成：

$$\Delta P / U = B' \Delta \theta$$

$$\Delta Q / U = B'' \Delta U$$

✓这里的B’与B”不仅阶数不同，而且其相应元素的构成也不相同，具体计算公式为：

$$\left.\begin{aligned} B'_{ij} &= -\frac{1}{X_{ij}}, \qquad B'_{ii} = -\sum_{j\neq i} B'_{ij} = \sum_{j\neq i}\frac{1}{X_{ij}} \\ B''_{ij} &= -\frac{X_{ij}}{R_{ij}^2+X_{ij}^2} = -B_{ij}, \quad B''_{ii} = -B_{io} + \sum_{j\neq i}\frac{X_{ij}}{R_{ij}^2+X_{ij}^2} = -B_{ii} \end{aligned}\right\}$$

✓式中：B_{ij}及B_{ii}分别为节点导纳矩阵相应元素；B_{i0}为节点i的总并联对地电纳，R_{ij}及X_{ij}为相应网络元件的电阻及电抗。

(二)快速解耦法的特点和性能

- ✓ 快速解耦法和牛顿法的不同，主要体现在修正方程式上面。比较两种算法的修正方程式，可见快速解耦法具有以下特点：

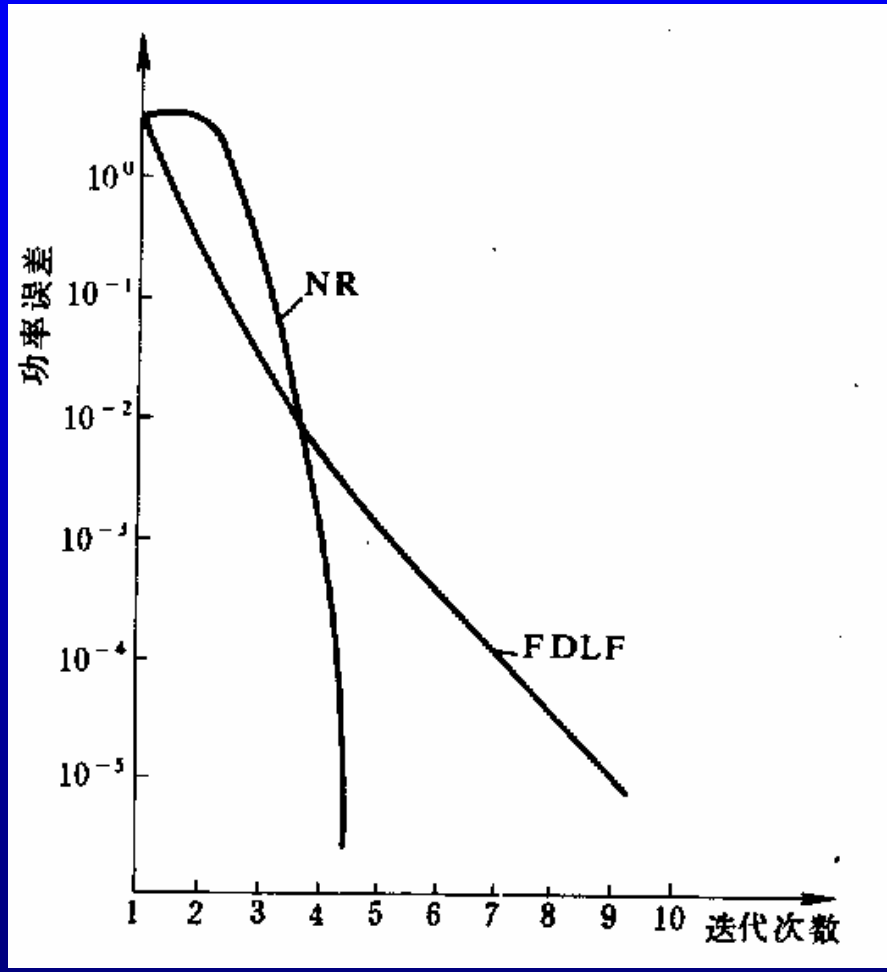
(1)用解两个**阶数几乎减半**的方程组(n-1阶及n-m-1阶)代替牛顿法的解一个2n-m-2阶方程组，显著地减少了内存需量及计算量；

(2)不同于牛顿法的每次迭代都要**重新形成雅可比矩阵**并进行三角分解，这里B'及B''是二个**常数阵**，因此大大缩短了每次迭代所需的时间，

(3)**雅可比矩阵J不对称**，而B'及B''**都是对称阵**，为此只要形成并贮存因子表的上三角或下三角部分，这样又减少了三角分解的计算量并节约了内存。

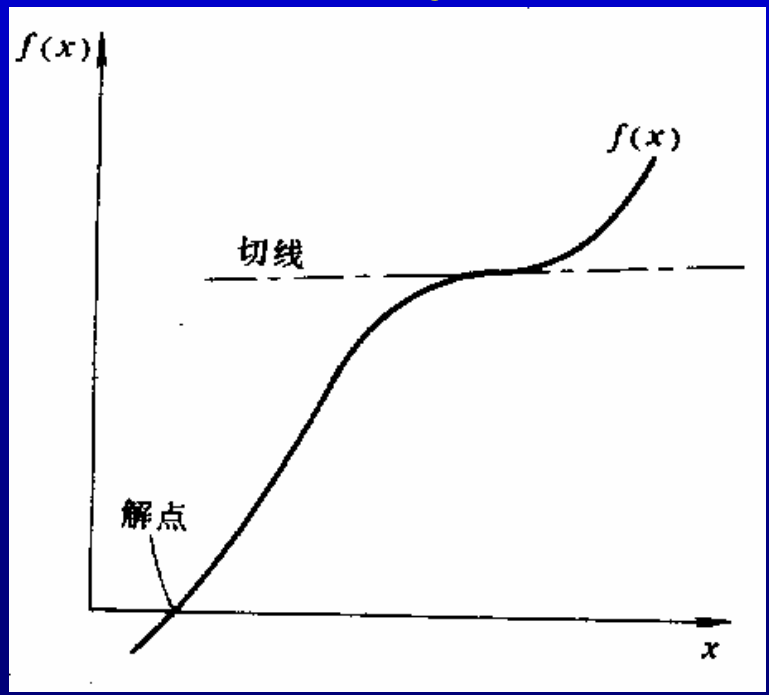
- ✓由于上述原因，快速解耦法所需的内存量约为牛顿法的60%，而每次迭代所需时间约为牛顿法的1／5。
- ✓就收敛特性而言，由于B'及B''在迭代过程中保持不变，在数学上属于“等斜率”法，因此本方法将从牛顿法的平方收敛特性退化为线性收敛特性。于是，快速解耦法达到收敛所需的迭代次数比牛顿法要多，但由于每次迭代所需的时间远比牛顿法少，所以总的计算速度仍有大幅度的提高。

✓下图表示了牛顿法和快速解耦法的典型收敛特性。



- ✓快速解耦法也具有良好的收敛可靠性。除了当网络中出现了在下面要进一步讨论的元件R／X比值过大的病态条件以及因线路特别重载以致两个节点间相角差特别大的情况之外，一般均能可靠地收敛。有些用牛顿法计算出现收敛困难的算题，转而采用本算法可能反而能够成功。

✓ 例如当求解的函数$f(x)$不是单调变化而具有“驼峰”，当初始值或迭代点正好落在这一范围内时，如图1-4所示，牛顿法的切线方程式将使修正量$\Delta x^{(k)}$出现异常的数值，其后果是减慢收敛速度或甚至导致发散，也有可能错误地收敛到一个不能运行的解点上去。而快速解耦法的“等斜率”求解过程却往往不会受到影响。



- ✓ 快速解耦法的程序设计较之牛顿法要来得简单。因此，简单、快速、内存节省以及较好的收敛可靠性形成了快速解耦法的突出优点，成为当前使用最为普遍的一个算法。它不仅大量地用在规划设计等离线计算的场合，而且由于其计算速度快，也已经广泛地在安全分析等在线计算中得到应用。

(三)元件大R/X比值病态问题

- ✓ FDLF是基于**两个基本假设**，R《X 以及线路两端相角差比较小。
- ✓ 当系统存在不符合这些假设的因素时，就会出现迭代次数大大增加或甚至不收敛的情况。而其中又以出现元件大R/X比值的机会最多，例如：
  - ➢ 低电压网络
  - ➢ 某些电缆线路
  - ➢ 三绕组变压器的等值电路
  - ➢ 通过某些等值方法所得到的等值网络等
- ✓ 大R/X比值病态问题已成为快速解耦法应用中的一个**最大障碍**。

✓解决这个问题的途径主要有以下两种：

1．对大R／X比值支路的参数加以补偿，可以分成串联补偿法及并联补偿法两种。

(1)串联补偿法

图 1-6　对大 R/X 比值支路的串联补偿

m为增加的虚构节点，－jXc为新增的补偿电容，X+Xc》R。

缺点：如果原来支路的R/X非常大，从而使Xc的值选的过大时，新增节点m的电压有可能偏离节点i及j的电压很多，从而这种不正常的电压本身导则潮流计算收敛缓慢，甚至不能收敛。

✓解决这个问题的途径主要有以下两种：

(2) 并联补偿法

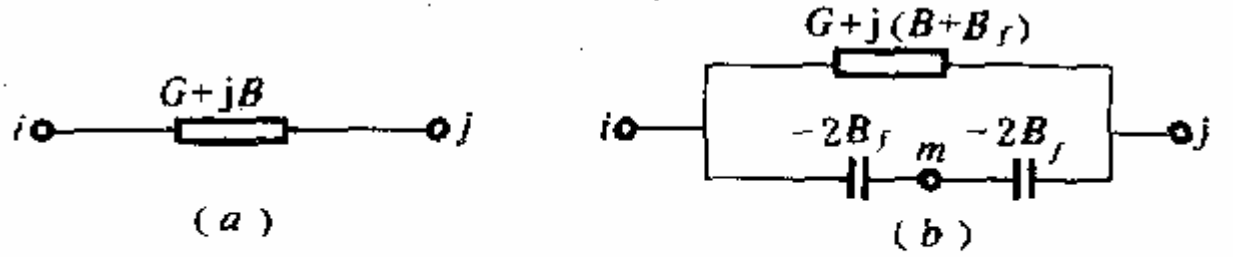


图 1-7　对大 R/X 比值支路的并联补偿

经过补偿的支路i－j的等值导纳为

$$Y_{ij} = G + j(B + B_f) + \frac{1}{\left(\frac{1}{-2jB_f} - \frac{1}{-2jB_f}\right)} = G + jB$$

即仍等于原来支路i－j的导纳值。

并联补偿新增节点m的电压不论B_f的取值大小都始终介于支路i－j两端点的电压之间，不会产生病态的电压现象，从而克服了串联补偿法的缺点。

2. 对算法加以改进

- ✓ 为了克服快速解耦算法在处理大R／x比值问题上的缺陷，许多研究工作立足于对原有算法加以改进，希望能找到一种方法，它既能保持快速解耦法的基本优点，又能克服由于大R／X比值问题所带来的收敛困难。在提出的这一类算法中，基本上还保留了原来解耦算法的框架，但对修正方程式及其系数矩阵B'及B''的构成作出各种不同的修改。

- ✓前已提到，在构成快速解耦法B'的元素时应不计串联元件的电阻(R)，仅用其电抗值(X)，而在形成B"的元素时则仍用精确的电纳值(B)，称为XB方案。
- ✓与此相对应，组成快速解耦法应该还有BX方案，BB方案和XX方案。
- ✓在不同的试验网络上所进行的大量计算实践表明，这些方案在处理大R／X比值问题上的能力以BB方案最差，XX方案稍好，但不如XB及BX方案。后两种方案在计算一些IEEE典型试验网络时所需的迭代次数，如表1-1所示。

表 1-1　　R/X 比值改变时两种算法对不同试验网络的收敛迭代次数[12]

（收敛精度 0.01MW/Mvar）

X 变值因子	14 节点系统		30 节点系统		57 节点系统		118 节点系统	
	XB	BX	XB	BX	XB	BX	XB	BX
1.000	4—4①	5—4	4—3	5—4	5—4	5—4	5—4	5—4
0.500	8—6	7—6	8—6	6—5	8—6	7—6	10—7	5—5
0.250	20—14	10—9	21—14	8—7	19—13	12—11	20—19	7—6
0.200	28—19	11—10	30—19	8—8	27—19	14—13	>60	8—7
0.166	>60	11—11	>60	9—9	>60	16—16		9—8
0.125		13—12		11—10		22—21		10—9
0.100		13—13		11—11		28—27		12—12

① 二个数字中前一个为有功迭代次数，后一个为无功迭代次数。

- BX方案占有明显的优势。

- 为了解决大R／X比值病态问题，以上所述对元件参数补偿以及对算法进行改进二种途径各有利弊。使用补偿法要增加一个节点，当网络中大R／X比值的元件数很多时将使计算网络的节点数增加很多。而采用改进算法就不存在这个问题。但目前已提出的一些改进算法并没有做到完全免除对元件R／X比值的敏感性。当某个元件的R／X比值特别高时，这些算法所需的迭代次数仍将急剧上升或甚至发散。这种情况下对这些元件采用补偿方法可能是一种好的选择。

第六节　保留非线性潮流算法

- ✓ 牛顿法求解非线性潮流方程时采用了逐次线性化的方法。20世纪70年代后期，人们开始研究这样的问题，即如果采用更加精确的数学模型，将泰勒级数的高阶项或非线性项也考虑进来，也许会进一步提高算法的收敛性能及计算速度，于是便产生了一类称之为保留非线性的潮流算法。又因为其中大部算法主要包括了泰勒级数的前三项即取到泰勒级数的二阶项，所以也称为二阶潮流算法，实现这种想法的第一个尝试是在极坐标形式的牛顿法修正方程式中增加了泰勒级数的二阶项，所得到的算法对收敛性能略有改善，但计算速度无显著提高。

- 后来，岩本伸一及田村康男根据直角坐标形式的潮流方程是一个二次代数方程组的这一特点，提出了采用直角坐标的保留非线性快速潮流算法，在速度上比牛顿法有较多的提高，引起了广泛的重视。在此以后，又出现了一些计入非线性的其它潮流算法。这些算法除了作为常规的潮流计算工具之外，也已经在状态估计、最优潮流等其它计算中得到应用。

一、保留非线性快速潮流算法

✓采用直角坐标形式的潮流方程为

$$P_i = \sum_{j \in i} [e_i (G_{ij} e_j - B_{ij} f_j) + f_i (G_{ij} f_j + B_{ij} e_j)]$$

$$Q_i = \sum_{j \in i} [f_i (G_{ij} e_j - B_{ij} f_j) - e_i (G_{ij} f_j + B_{ij} e_j)]$$

$$U_i^2 = e_i^2 + f_i^2$$

✓该潮流问题实际上就是求解一个不含变量一次项的二次代数方程组。

- 对这样的方程组用泰勒级数展开，则二阶项系数已是常数，没有二阶以上的高阶项，所以泰勒级数只要取三项就能够得到一个没有截断误差的精确展开式。因此从理论上，假若能够从这个展开式设法求得变量的修正量，并将它对估计初值加以修正，则只要一步就可求得方程组的解。而牛顿法由于线性近似，略去了高阶项，因此用每次迭代所求得的修正量对上一次的估计值加以改进后；仅是向真值接近了一步而已。

（一）数学模型

✓我们先定义

✓ y(x)为n 维函数相量

$$y(x)=[y_1(x), y_2(x), \cdots, \ y_n(x)]^T$$

y^s 为n 维函数给定值相量

$$y^s=[y_1{}^s, y_2{}^s, \cdots, \ y_n{}^s]^T$$

x为n维未知变量相量

$$x=[x_1, x_2, \cdots, \ x_n]^T$$

✓一个具有n个变量的齐次二次代数方程式的普遍形式为：

$$
\begin{aligned}
y_i(x) = &\left[(a_{11})_i x_1 x_1 + (a_{12})_i x_1 x_2 + \cdots + (a_{1n})_i x_1 x_n\right] \\
&+ \left[(a_{21})_i x_2 x_1 + (a_{22})_i x_2 x_2 + \cdots + (a_{2n})_i x_2 x_n\right] \\
&+ \cdots + \left[(a_{n1})_i x_n x_1 + (a_{n2})_i x_n x_2 + \cdots + (a_{nn})_i x_n x_n\right]
\end{aligned}
$$

✓于是潮流方程组就可以写成如下的矩阵形式

$$y^s = y(x) = A \begin{bmatrix} x_1 x \\ \cdots \\ x_2 x \\ \cdots \\ \vdots \\ \cdots \\ x_n x \end{bmatrix}$$

或

$$f(x) = y(x) - y^s = 0$$

式中系数矩阵为

$$A=\left[\begin{array}{cccc:cccc:c:cccc}
(a_{11})_1 & (a_{12})_1 & \cdots & (a_{1n})_1 & (a_{21})_1 & (a_{22})_1 & \cdots & (a_{2n})_1 & \cdots & (a_{n1})_1 & (a_{n2})_1 & \cdots & (a_{nn})_1 \\
(a_{11})_2 & (a_{12})_2 & \cdots & (a_{1n})_2 & (a_{21})_2 & (a_{22})_2 & \cdots & (a_{2n})_2 & \cdots & (a_{n1})_2 & (a_{n2})_2 & \cdots & (a_{nn})_2 \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\
(a_{11})_n & (a_{12})_n & \cdots & (a_{1n})_n & (a_{21})_n & (a_{22})_n & \cdots & (a_{2n})_n & \cdots & (a_{n1})_n & (a_{n2})_n & \cdots & (a_{nn})_n
\end{array}\right]$$

$$A \in R^{n\times n^2}$$

✓ 把$y_i(x)$在初值$x^{(0)}$附近展开，可得到如下没有截断误差的精确展开式

$$y_i(x) = y_i(x^{(0)}) + \sum_{j=1}^{n} \left.\frac{\partial y_i}{\partial x_j}\right|_{x=x^{(0)}} \Delta x_j + \frac{1}{2!}\sum_{j=1}^{n}\sum_{k=1}^{n} \left.\frac{\partial^2 y_i}{\partial x_j \partial x_k}\right|_{x=x^{(0)}} \Delta x_j \Delta x_k$$

✓ 于是与y^S对应的精确的泰勒展开式为

$$\boldsymbol{y}^s = \boldsymbol{y}(\boldsymbol{x}^{(0)}) + \boldsymbol{J}\Delta\boldsymbol{x} + \frac{1}{2}\boldsymbol{H}\begin{bmatrix} \Delta x_1 & \Delta x \\ \Delta x_2 & \Delta x \\ \vdots & \\ \Delta x_n & \Delta x \end{bmatrix}$$

式中 $\Delta x = [x - x^{(0)}] = [\Delta x_1, \Delta x_2, \cdots, \Delta x_n]^T$ 为修正相量

$$\boldsymbol{J} = \begin{bmatrix} \dfrac{\partial y_1}{\partial x_1} & \dfrac{\partial y_1}{\partial x_2} & \cdots & \dfrac{\partial y_1}{\partial x_n} \\ \dfrac{\partial y_2}{\partial x_1} & \dfrac{\partial y_2}{\partial x_2} & \cdots & \dfrac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \dfrac{\partial y_n}{\partial x_1} & \dfrac{\partial y_n}{\partial x_2} & \cdots & \dfrac{\partial y_n}{\partial x_n} \end{bmatrix}_{\boldsymbol{x}=\boldsymbol{x}^{(0)}} \qquad \boldsymbol{J} \in R^{n\times n}$$

$$\boldsymbol{H} = \left[\begin{array}{cccc:cccc:c:cccc} \dfrac{\partial^2 y_1}{\partial x_1 \partial x_1} & \dfrac{\partial^2 y_1}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 y_1}{\partial x_1 \partial x_n} & \dfrac{\partial^2 y_1}{\partial x_2 \partial x_1} & \dfrac{\partial^2 y_1}{\partial x_2 \partial x_2} & \cdots & \dfrac{\partial^2 y_1}{\partial x_2 \partial x_n} & \cdots & \dfrac{\partial^2 y_1}{\partial x_n \partial x_1} & \dfrac{\partial^2 y_1}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 y_1}{\partial x_n \partial x_n} \\ \dfrac{\partial^2 y_2}{\partial x_1 \partial x_1} & \dfrac{\partial^2 y_2}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 y_2}{\partial x_1 \partial x_n} & \dfrac{\partial^2 y_2}{\partial x_2 \partial x_1} & \dfrac{\partial^2 y_2}{\partial x_2 \partial x_2} & \cdots & \dfrac{\partial^2 y_2}{\partial x_2 \partial x_n} & \cdots & \dfrac{\partial^2 y_2}{\partial x_n \partial x_1} & \dfrac{\partial^2 y_2}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 y_2}{\partial x_n \partial x_n} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ \dfrac{\partial^2 y_n}{\partial x_1 \partial x_1} & \dfrac{\partial^2 y_n}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 y_n}{\partial x_1 \partial x_n} & \dfrac{\partial^2 y_n}{\partial x_2 \partial x_1} & \dfrac{\partial^2 y_n}{\partial x_2 \partial x_2} & \cdots & \dfrac{\partial^2 y_n}{\partial x_2 \partial x_n} & \cdots & \dfrac{\partial^2 y_n}{\partial x_n \partial x_1} & \dfrac{\partial^2 y_n}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 y_n}{\partial x_n \partial x_n} \end{array}\right]$$

$$\boldsymbol{H} \in R^{n\times n^2}$$

$$y^s = y(x^{(0)}) + J\Delta x + \frac{1}{2}H\begin{bmatrix} \Delta x_1 & \Delta x \\ \Delta x_2 & \Delta x \\ \vdots & \\ \Delta x_n & \Delta x \end{bmatrix}$$

✓H是一个常数矩阵，其阶数很高，但高度稀疏。注意若精确的泰勒展开式中略去第三项，就成为通常的牛顿法的展开式。

$$y^s = y(x^{(0)}) + J\Delta x + \frac{1}{2}H\begin{bmatrix} \Delta x_1 & \Delta x \\ \cdots\cdots & \\ \Delta x_2 & \Delta x \\ \cdots\cdots & \\ \vdots & \\ \Delta x_n & \Delta x \end{bmatrix}$$

- ✓ 研究表明可以进一步将上式改写为

$$y^s = y(x^{(0)}) + J\Delta x + y(\Delta x)$$

- ✓ 该式的推出促成了本算法的突破，因为可以非常方便地计算二阶项。值得顺便指出的是，该式是一个很重要的关系式，在研究其它算法时将多次引用。