分类号：_____ 密　级 _____

U D C：_____ 单位代码 10151

大连海事大学

全日制应用型硕士研究生学位论文

基于 MATLAB 的科研型潮流算法研究与设计

曹井川

指 导 教 师	姚玉斌　教 授
申请学位类别	工 程 硕 士
工 程 领 域	电气工程
学位授予单位	大连海事大学

2017 年 6 月

分 类 号 _____ 密 级 _____

U D C _____ 单位代码 10151

大 连 海 事 大 学

工程硕士学位论文

基于 Matlab 的科研型潮流算法研究与设计

(应用研究)

曹 井 川

指 导 教 师	姚玉斌	职 称	教授
学位授予单位	大连海事大学		
申请学位级别	工程硕士	工程领域	电气工程
论文完成日期	2017 年 5 月	答辩日期	2017 年 6 月

答辩委员会主席 朱景伟

Study and Design of the Research-oriented Power Flow Algorithm Based on Matlab

A Thesis Submitted to

Dalian Maritime University

In partial fulfillment of the requirements for the degree of

Master of Engineering

by

Cao Jingchuan

(Electrical Engineering)

Thesis Supervisor: Professor **Yao Yubin**

May 2017

大连海事大学学位论文原创性声明和使用授权说明

原创性声明

本人郑重声明：本论文是在导师的指导下,独立进行研究工作所取得的成果，撰写成硕士学位论文 “基于 Matlab 的科研型潮流算法研究与设计”。除论文中已经注明引用的内容外，对论文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本论文中不包含任何未加明确注明的其他个人或集体已经公开发表或未公开发表的成果。本声明的法律责任由本人承担。

学位论文作者签名：黄井川

学位论文版权使用授权书

本学位论文作者及指导教师完全了解大连海事大学有关保留、使用研究生学位论文的规定，即：大连海事大学有权保留并向国家有关部门或机构送交学位论文的复印件和电子版，允许论文被查阅和借阅。本人授权大连海事大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，也可采用影印、缩印或扫描等复制手段保存和汇编学位论文。同意将本学位论文收录到《中国优秀博硕士学位论文全文数据库》（中国学术期刊（光盘版）电子杂志社）、《中国学位论文全文数据库》（中国科学技术信息研究所）等数据库中，并以电子出版物形式出版发行和提供信息服务。保密的论文在解密后遵守此规定。

本学位论文属于：　　保　密□ 在____年解密后适用本授权书。

不保密□√（请在以上方框内打“√”）

论文作者签名：黄井川　导师签名：姚玉斌

日期：2017 年 6 月 19 日

摘　要

潮流计算是电力系统分析中一种基本运算，按照使用目的的不同又可分为科研型潮流计算和工程型潮流计算。科研型潮流算法主要是作为一种在研究中使用的运算工具，此算法的突出特点是矩阵元素修改方便、可维护性好，常被用作算法的进一步开发。Matlab 凭借其强大的功能，逐渐被越来越多的研究人员所使用，在研究工作中也迫切需要一种运用 Matlab 实现的潮流程序来进行相关的运算分析。

为此，从循环和矩阵运算两个角度设计了算法，并使用 Matlab 进行算法实现。

首先，对基于循环结构的科研型潮流算法进行了研究，设计了一种循环形成雅可比矩阵的方法，并通过判断节点类型来跳过不必要运算，节省了运算时间。从修正方程系数矩阵的元素排列特点和 Matlab 矩阵存储规律出发，对线性方程的求解过程进行了算法设计。

然后，对运用矩阵运算的科研型潮流算法进行了研究，主要是对雅可比矩阵的形成过程进行算法设计。根据形成雅可比矩阵的思路不同提出了矩阵相乘、点乘以及重复矩阵函数 3 种矩阵处理方式，具体又分为了 5 种不同的方法。由于功率不平衡量的计算以及雅可比矩阵的对角元素修正均需要用到节点计算功率，对节点计算功率的形成也设计了两种算法，分别为直接计算法和矩阵元素求和法。由于潮流计算中涉及的矩阵多为稀疏矩阵，采用稀疏矩阵技术对相关矩阵进行了处理，提高了运算速度。

所设计的方法对雅可比矩阵均按照$(2n\times 2n)$进行存储，方便科研工作中的元素寻找、参数修改，同时便于算法的二次开发。通过对东北电网 445 节点电力系统算例分析，说明了设计的算法在速度上符合研究使用，具备较强的实用价值。

关键词：科研型潮流算法；矩阵运算；雅可比矩阵；线性方程求解

ABSTRACT

Power flow is a basic calculation in power system analysis, it can be divided into research-oriented power flow and engineering-oriented power flow according to the different using purpose. Research-oriented power flow algorithm is mainly used as a basic computing tool in scientific research, the most outstanding feature of this algorithm is that the matrix will be easily modified and maintained, which makes this algorithm often be used for algorithm development. Matlab is gradually adopted by many researchers due to its powerful functions. In the research work, researchers need a fast-speed power flow program urgently to do necessary computing analysis of power system.

In this thesis, the algorithms are designed from the two aspects which are circulation and matrix operations.

Firstly this thesis studies the research-oriented power flow algorithm based on circulation computing, designs a looping method to form Jacobian matrix and skips the unnecessary operation to save time by confirming the bus type. Solving the linear equation algorithm is designed after the study on elements arrayed rule of the coefficient matrix and Matlab matrix storage rule.

Secondly, this thesis studies the research-oriented power flow algorithm based on matrix operation, mainly designs an algorithm for the formation of Jacobian matrix. On the basis of the different thoughts of forming Jacobian matrix, matrix multiplication method, dot product method and repetitive matrix are proposed, which can also be divided into five different methods. Two algorithms are designed for the formation of the injection power, which are the direct computing method and the matrix element summation method. All kinds of matrix involved in power flow mainly are sparse matrix, so that this thesis adopted the sparse technology to manipulated these matrix and improved the computing speed.

In this thesis, the Jacobian matrix is stored by the way of $(2n \times 2n)$ to facilitate element search and parameter modification in research work. At the same time, it also makes the algorithm has the possibility of second development. In addition, the calculation of 445-bus Northeast power system indicated that the algorithm can satisfy the use of scientific research and have a strong practical value.

Key Words: Research-oriented Power flow algorithm; Matrix operation; Jacobian matrix; Linear equations solving

目　录

第 1 章 绪论

1.1 选题背景

改革开放以来，我国的电力工业发展迅速，电能在能源终端消费的使用比重持续增加，提高了资源利用效率，使资源分配更加合理。电力系统是由生产、输送、变换、分配、使用电能的一次设备及其相应的控制、测量、保护等装置所组成的庞大而统一的整体[1]。现代电力系统正朝着远距离、大规模、特高压的输电方式发展，区域性联网乃至全国联网已成为大势所趋，大规模电网在带来诸多便利的同时，也对电力系统稳定运行提出了更高的要求[2]。电力系统分析通常是研究解决电力系统运营规划问题的基本方法，其主要功能就是依据系统给定的运行、接线方式对系统的运行情况进行计算分析[3]。在电力系统分析中，潮流计算通常是最基本、最常用的计算，根据系统所给定的各种运行条件、以及系统的网络结构和参数，求解得到各母线电压的相角、幅值等相应状态变量，进而得到系统内各部分功率的分布及损耗[4]。

潮流计算在电力系统分析中有着不可替代的作用，在进行电力系统分析、规划、运行方式选择等相关研究时，往往要对潮流计算的结果进行定量分析，从而对规划方案或者运行方式的可靠性、实用性进行评估[5-7]。其意义主要体现在以下几个方面[8]：

(1) 为电气主接线方式的确定和电气设备的选择提供数据。

(2) 潮流计算结果是进行电力系统继电保护相关整定计算时的重要参考。

(3) 为检修工作的安排及运行方式的选择提供参考。

(4) 在负荷容量增加或电力网络扩建时，需要进行基本情况和各种预想事故下的潮流计算，以此来确定需增加的装机容量以及供电是否可靠。

(5) 在进行静态稳定和暂态稳定分析时，需要对系统进行潮流计算并确定系统正常运行状态的值，并以此为参考，来判定系统受到干扰时的稳定性。

上述功能依靠离线潮流计算实现，而在线潮流计算的用途也十分广泛。随着电网不断升级改造，电网采集与监控系统（SCADA）和能量管理系统（EMS）在调度中心已经普及[9-10]。为达到系统实时监控这一目的，需要及时分析实时数据库所储存的有效数据，并依此来确定系统实时运行状态，在线潮流计算也就随之产

生。

潮流计算是在系统稳定状态下进行的计算，不包括负载、发电机的动态属性和过渡过程。因此，从数学角度来看，潮流计算实际上就是对高阶非线性的代数方程进行求解，不包括微分方程，对于这种非线性的方程通常使用迭代法进行求解。近年来，我国电力工业仍保持较快地发展速度，潮流算法的研究也逐渐成为一个热门的课题，相关的研究不仅体现在速度的提高、收敛性能的优化而且要使算法具备更好的人机交互性、灵活性[11]。

1.2 潮流算法的研究历程与现状

20 世纪 50 年代中期，电子计算机作为当时一种新兴计算工具逐渐开始应用于潮流计算，也促使了更多的潮流算法被使用。判断潮流算法性能优劣的依据主要为算法的收敛性，速度及内存占有量大小，是否满足灵活使用等方面。

在数字计算机应用于潮流计算的初始阶段，高斯-赛德尔法成为当时一种通用的解法。由于其原理简单，对内存需求较少，与当时计算机的计算水平以及和电力系统研究的层次相符合，但对于大规模的系统，其节点数更多，迭代的次数也随之增加，迭代过程中不收敛的现象开始频繁出现。

到了 20 世纪 60 年代，第二代数字计算机已经发展成熟，阻抗法开始得到广泛应用。阻抗法虽然没有导纳法迭代时所存在的收敛问题，但是由于其不具有导纳矩阵的稀疏性，导致阻抗法每次迭代的运算量比较大，牛顿法潮流计算的出现很好地解决了这一问题。牛顿法对于非线性方程的求解问题而言是一种非常经典的方法，在进行非线性方程求解时具备较好的收敛性。牛顿法在应用于电力系统潮流问题时，使用的是导纳矩阵。在利用导纳矩阵本身所具有的对称性的同时，使用了稀疏技术和节点优化编号等相关实用方法使得牛顿法速度更快、实用性更强[12]。同一时期，一些学者还对直流法、非线性规划法等几种潮流算法做了深入的探讨研究[13-15]，并取得一定的研究成果。

20 世纪 70 年代以后 Stott B 和 Alsao 改进了纯数学的牛顿法，提出了快速分解法[16]，该算法是 PQ 分解法所包括相关算法中最为优秀的算法之一。这种算法从参数实际的物理意义出发利用了电力网络的特点，对修正方程系数矩阵中的相关性较弱的元素进行了简化，使其常数化，并降低了矩阵阶数，显著提高了算法的计

算速度，减少了运算时对计算机内存的占用，使得一个内存为 32K 的数字计算机便可进行含 1000 个网络节点的系统的潮流计算[17-18]，适合在线分析和对计算速度要求较高的安全分析，对潮流算法的研究还包括保留非线性潮流算法[19]，概率潮流计算[20-21]等算法。

进入 21 世纪以后，以牛顿法和 PQ 分解法为代表的各类潮流算法的研究依然比较活跃 [22-24]。近年来，人工智能理论逐渐发展、相关数学理论更加丰富，模糊算法[25-26]、人工神经网络[27]、遗传算法[28-30]、同伦算法[31]等相关算法逐渐被引入潮流计算。但在众多潮流算法中，牛顿法与快速分解法这两种方法仍然是无法被其他方法替代的，学者们也不断尝试从不同方面对这两种方法进行改进，主要体现在计算速度的加快和收敛性的改善。

计算速度一直是潮流算法优劣的重要评判标准。以牛顿法为例，对牛顿法计算速度影响较大的主要有两个方面：第一个是线性方程组的迭代求解过程，第二个是修正方程系数矩阵即雅可比矩阵的形成过程。

稀疏技术能压缩矩阵的存储空间，加快相关矩阵的形成，对于提高潮流计算的速度意义重大。针对潮流算法的设计主要涉及以下几个方面：① 节点编号的优化，包括动态编号法、压缩带宽法等[32-33]。② 存储技术的改进，包括使用单链表法、十字链表法等方法进行存储[34-35]。③ 改进消元技术，对 LU 分解法和高斯消元法等方法进行改进[36-37]。文献[38]提出了一种综合潮流稀疏技术，使用十字链表和二叉链层表分别存储雅可比矩阵和节点导纳矩阵，两个矩阵之间通过指针直接关联，这一改进提高了导纳矩阵的提取速度，同时也避免了消元过程中对最原始的雅可比矩阵的破坏。文献[39]由矩阵的存储方式出发提出了二维链表的存储方法，与传统的单一方向存储相比，这种方法可以分别从行、列两个方向对矩阵进行快速地检索，大大提高了元素删除、插入、读取等相关操作的速度。

除了利用稀疏技术，学者们也在不断尝试从一些新的角度来进行潮流问题速度的提高。文献[40]运用了并行算法进行潮流计算，并行算法可定义为一种将多任务映射到多处理器的计算分析方法，与传统的串行算法相比，在计算大规模、多节点的复杂电力系统时有着较大的优势。该文献分析了国内外并行算法的研究现状，并比较了四种国内外认可度较高的并行算法：分块法、多重因子化法、稀疏矢量法和逆矩阵法，这四种算法的结合在未来会有很大的潜力和研究意义。

收敛性的改善同样是潮流算法改进的重要目的。文献[41]提出了一种变雅可比牛顿法的方法，提高了潮流算法在进行含小阻抗支路的系统计算时的收敛性。文献[42]提出了一种主要适用于配电网的新算法—交替迭代算法，此算法将线损计算和电压计算交替迭代进行，最后计算每条线路的功率。

Matlab 具备十分强大的矩阵处理能力，在进行潮流程序编写时有独特的优势。文献[43]运用矢量化编程提高了编程效率，并获得简洁直观的程序代码。文献[44]针对潮流计算的特点，利用 Matlab 在矩阵运算时的特点，对潮流程序进行了节点编号优化、稀疏技术处理以及矩阵除法运算等特殊处理，使程序更加简洁、快速。

在电力系统分析中，潮流计算是最基本的运算，根据研究重点的差异可以进行不同的分类，例如：按照潮流计算能否进行实时量测数据分析分为离线潮流计算和在线潮流计算[45]；按照使用范围的不同可以分为配电网潮流计算和输电网潮流计算；按照使用目的的不同又可分为科研型潮流计算和工程型潮流计算。本文将从算法设计出发，对科研型潮流算法进行探讨研究，并设计一种符合科研人员使用的潮流算法。

1.3 科研型潮流算法的研究

1.3.1 科研型潮流算法

潮流算法作为电力系统分析计算的主要工具用途十分广泛，根据其应用目的差异大体可分为两类：工程型的潮流计算和科研型的潮流计算。前者强调的是潮流计算在工程领域的应用，主要体现在对速度要求较高，为达到这一要求，稀疏技术与节点编号优化等方法也一并引入。这些方法的引入大大提高了计算速度，但编程较为复杂且不易修改、维护。后者更注重的是潮流计算在科研领域的应用，这些科研工作中所用的潮流算法其特点可概括如下：

(1) 潮流算法应具备较强的灵活性、易于修改。例如，在静态安全分析时，常进行一些设备的开断模拟以此来校验系统承受突发事故的能力，而进行这些开断模拟会引起网络参数和电力系统局部结构发生一些变化，因而要对导纳矩阵做出修正[46-47]。类似地在进行最优潮流计算时也会在优化过程中使控制变量发生变化进而使原始矩阵发生变化[48]。另外，在处理含小阻抗支路的病态潮流问题时需要对常规潮流算法做出改进，而一些常用的改进方法如文献[49]提出的修正系数的方

法、文献[50]提出的修正雅可比牛顿法都要对潮流算法中的雅可比矩阵进行修改。换言之，科研型潮流算法具备进一步开发的可能性。从这个角度来讲，科研人员使用的潮流程序应当具有较好的灵活性、各矩阵易于修改且便于维护[51]。稀疏技术和节点编号优化等技术虽然对速度有显著提高，但相关矩阵的修改、矩阵元素的寻找不方便。

(2) 潮流计算时应具备一定的速度。在科研工作中，虽然对速度要求不像工程计算中那样高，但仍希望所使用的算法能够快速地计算出结果，这样能够节省时间，便于其他工作的展开。

1.3.2 科研型潮流算法的编程工具

本文选择了用 Matlab 进行算法实现。具体原因如下：

(1) Matlab 是目前国际上认可度最高、使用范围最广的计算软件。最初 Matlab 的应用领域主要是解决矩阵运算问题，但随着近几十年的发展，Matlab 已经逐渐成为一种能够实现符号运算、数值分析计算、可视化、实验仿真、图形处理、编程等多种实用功能的集成软件[52]。Matlab 拥有数十个功能齐全、方便实用的工具箱，尤其是领域性工具箱可为学科研发提供非常大的帮助。在研究工作中，由于其功能强大、方便实用，Matlab 也逐渐成为许多研究人员的首选。在研究工作中也迫切需要一种运用 Matlab 实现的快速潮流程序来进行相关的运算分析[53]，这也正是本论文选择在 Matlab 平台实现的重要原因。

(2) Matlab 代码短小、编程容易、易于操作[54-56]。在 Matlab 的命令窗口中进行输入指令输入后，立即可以看到输出结果，并且相关函数的查询十分方便，使用者能够通过帮助来更好地学习 Matlab，体现了良好的交互性。

(3) 强大的矩阵运算能力。Matlab 有很多的矩阵函数，在进行矩阵分析运算时，可直接利用这些函数，矩阵的求逆、点乘等基本代数运算，求解特征向量、特征值，以及矩阵的生成、提取等。利用这些函数可将一般高级语言中复杂的循环结构用一个简洁的 Matlab 函数实现。在潮流程序编写时，会涉及到很多的矩阵形成、运算。

1.4 本文的主要研究内容

本文主要对适合科研使用的极坐标牛顿法潮流计算方法进行研究，根据

Matlab 的编程特点从循环和矩阵运算两个不同角度分别设计了不同的算法，使其在满足科研目的的同时，速度同样得到提高。具体工作如下：

第 1 章：首先简要介绍了本文的选题背景，根据相关文献分析了潮流计算的研究现状，然后又介绍了科研型潮流算法这一概念，以及此算法在 Matlab 软件平台实现的可行性及优越性。

第 2 章：对电力网络的数学模型进行了简要介绍，推导了牛顿法潮流算法的基本原理，对线性方程的解法进行了相关说明，完整地描述了潮流计算的数学过程，为后面两章算法的研究设计提供了理论支持。

第 3 章：主要设计了一种通过循环形成雅可比矩阵的方法及 3 种求解线性方程的方法，分别是：按行运算的高斯法、按列运算的高斯法、Matlab 的矩阵除法，并将这 3 种方法分别与循环形成雅可比矩阵的方法进行组合组成 3 种不同的潮流计算方法，通过算例的计算时间说明了算法的有效性、实用性。

第 4 章：主要研究了通过矩阵运算形成雅可比矩阵的方法，根据形成雅可比初始计算矩阵的思路不同，共提出了矩阵对角化相乘法、点乘法以及重复矩阵 3 种处理方式，具体又可分为 5 种不同的方法。对节点计算功率的形成设计了两种算法，分别是直接计算法和矩阵元素求和法，并将它们分别与线性方程解法组合进行算例计算，比较计算时间。

第 2 章 电力系统潮流计算数学模型

2.1 引言

电力系统是一个复杂的系统，包含了生产、变换、输送、消费电能的四个主要部分，这四个部分主要包含了发电机、变压器、线路、负荷、以及补偿设备等电力系统元件。为了能对系统各部分进行准确计算，需要结合电力系统各元件的物理模型和电磁过程给出相对简化而又符合实际的数学模型。

得到了系统各元件的数学模型之后，整个网络的电路模型也就随之确定，接下来，要结合所确定的电路模型，利用电路中的 KCL、KVL 定理，列出电力网络的方程。考虑到本文使用的潮流算法的特点，本章只讨论节点电压方程。

建立相关的网络方程组之后，便可进行潮流分布的计算。但在实际的工程实践中，一般给定的是各节点功率以及相关网络参数，得到的是一组非线性方程。求解非线性方程通常使用迭代的方法，在潮流计算中，牛顿法和快速分解法是当前普遍使用的两种算法。牛顿法收敛性能好，得到研究者的青睐。

非线性方程通过牛顿法线性化后要进行线性方程的求解，高斯法是求解线性方程的常用方法，运用高斯法时可对方程按行消元或按列消元。

2.2 潮流计算数学模型

2.2.1 电力系统网络元件数学模型

在电力系统潮流计算中，主要考虑输电线路、变压器、发电机、负荷的等值数学模型。

(1) 输电线路数学模型

在电力系统稳态分析过程中，通常假设电力系统三相对称并且线路已经完成了整循环换位。在这一假设下，三相电路模型可以用单相等值电路模型表示。输电线路的数学模型就是以电阻、电纳、电抗、电导等四种网络参数表示的等值电路，四种参数分别具有不同的物理意义：电阻、电抗、电纳分别反映的是线路的热效应、磁场效应、电场效应，电导表征的是线路在潮湿环境下出现的电晕和泄漏电流等现象。需要说明的是，由于泄漏电流一般非常小，并且在晴朗天气下电晕一般不会发生，因此，一般可设电导为 0。

当架空线路长度小于 300km，电缆线路长度小于 100km 时可忽略线路的波过程，其数学模型用集中参数表示，而超过这个范围之后需考虑分布参数特性即需要用相应的系数来修正，但这两种情况都可以用 π 型等值电路表示，如图 2.1 所示，图中 Z 指的是线路阻抗，B 表示线路电纳。

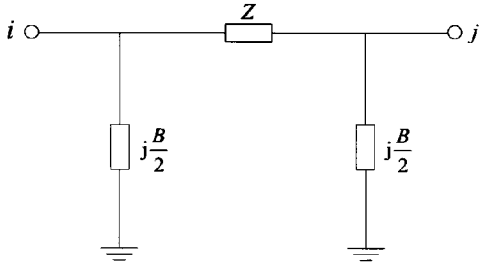

图 2.1 输电线路 π 型等值电路

Fig. 2.1 The π-equivalent circuit of transmission line

(2) 变压器数学模型

建立变压器模型，以双绕组变压器为例。首先要根据铭牌数据得到变压器的各类参数：R_T、X_T 分别表示变压器等值绕组的电阻、电抗；G_T、B_T 分别表示变压器的电导、电纳；k 表示非标准变比。将励磁支路向前移动得到变压器Γ型等值电路，如图 2.2 所示。

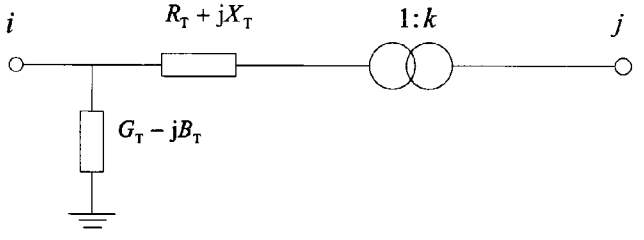

图 2.2 双绕组变压器Γ型等值电路

Fig 2.2 The Γ equivalent circuit of two-winding transformer

图 2.2 所示模型节点数少，且图中所示参数也有各自的物理意义，但由于存在非标准变比，在潮流计算时需将二次侧的线路和其他元件参数折算到一次侧，无法对多电压等级环形网络中的相关参数和变量进行严格准确地归算，因此不适合大型电力系统的计算。因此，为了适应多电压等级系统进行计算机编程，一般选择变压器 π 型等值电路进行计算，如图 2.3。

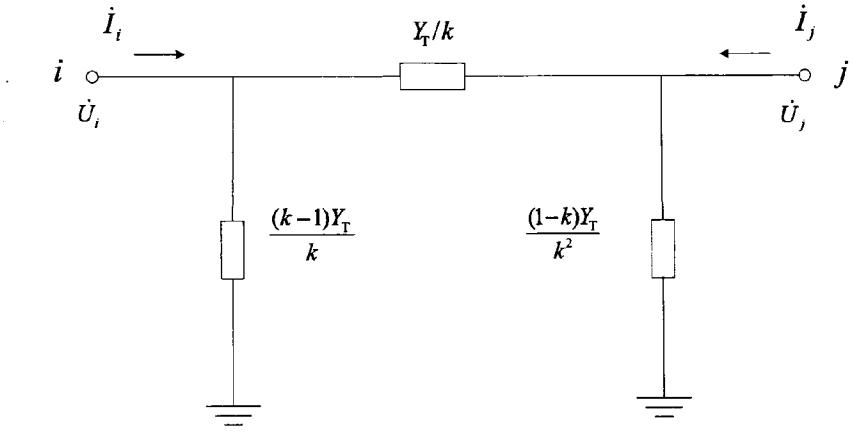

图 2.3 双绕组变压器 π 型等值电路

Fig 2.3 The π equivalent circuit of the two-winding transformer

图 2.3 所示变压器模型体现了变压器所具备的电压变换功能，在已知变压器的实际变比 k 和变压器导纳 Y_T 情况下可以得到相关节点的自导纳以及互导纳：$Y_{ii}=Y_T$、$Y_{ij}=Y_{ji}=Y_T/k$、$Y_{jj}=Y_T/k^2$。

2.2.2 电力网络节点电压方程

电力网络方程可定义为一组用来描述电力系统参数和变量两者关系的数学方程组，电路分析方法一般包括节点电压法、回路电流法以及割集电压法三种常用方法。其中，节点电压方程在潮流计算中被采用的频率最高。

在列写电力网络方程时，通常选取大地作为整个网络的参考节点即通常所说的零电位点。假设系统中除参考节点外有 n 个节点，以这 n 个节点为研究对象，可列节点电压方程

$$\boldsymbol{Y}_B\boldsymbol{U}_B=\boldsymbol{I}_B \tag{2.1}$$

式中，$\boldsymbol{I}_B$ 表示节点注入电流的列向量，$\boldsymbol{U}_B$ 表示节点电压的列向量。

将式(2.1)展开为方程

$$\left.\begin{aligned}
&Y_{11}\dot{U}_1+Y_{12}\dot{U}_2+\cdots+Y_{1n}\dot{U}_n=\dot{I}_1\\
&Y_{21}\dot{U}_1+Y_{22}\dot{U}_2+\cdots+Y_{2n}\dot{U}_n=\dot{I}_2\\
&\quad\vdots\\
&Y_{n1}\dot{U}_1+Y_{n2}\dot{U}_2+\cdots+Y_{nn}\dot{U}_n=\dot{I}_n
\end{aligned}\right\} \tag{2.2}$$

以节点矩阵形式来表示

$$\begin{bmatrix} Y_{11} & Y_{12} & \cdots & Y_{1n} \\ Y_{21} & Y_{22} & \cdots & Y_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ Y_{n1} & Y_{n2} & \cdots & Y_{nn} \end{bmatrix} \begin{bmatrix} \dot{U}_1 \\ \dot{U}_2 \\ \vdots \\ \dot{U}_n \end{bmatrix} = \begin{bmatrix} \dot{I}_1 \\ \dot{I}_2 \\ \vdots \\ \dot{I}_n \end{bmatrix} \tag{2.3}$$

式(2.3)中注入电流的实际意义可理解为各发电机电流与各负荷电流之和，并以发电机注入网络的电流为正。所以，若仅有负荷节点无发电机节点，则其注入电流为负值。另外，还存在一些联络节点在网络中仅起联络作用，其注入电流为零。而式(2.3)的节点电压通常是指该节点与参考节点的电压差，本文选取大地作为参考节点，编号为零。

节点导纳矩阵 $\boldsymbol{Y}_B$ 表示为

$$Y_B = \begin{bmatrix} Y_{11} & Y_{12} & \cdots & Y_{1i} & \cdots & Y_{1n} \\ Y_{21} & Y_{22} & \cdots & Y_{2i} & \cdots & Y_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ Y_{i1} & Y_{i2} & \cdots & Y_{ii} & \cdots & Y_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ Y_{n1} & Y_{n2} & \cdots & Y_{ni} & \cdots & Y_{nn} \end{bmatrix} \tag{2.4}$$

下面结合式(2.4)对节点导纳矩阵进行一下说明：

(1) 式(2.4)中的对角元素 Y_{ii} 称为自导纳，表达式为 $Y_{ii}=\sum_{j\in i} y_{ij}$，即与节点 i 直接相连的各支路互导纳之和。节点导纳矩阵中的非对角元素 Y_{ij} 被称为互导纳，值为 $Y_{ij}=Y_{ji}=-y_{ij}$，y_{ij} 为两节点所在支路的导纳值。

(2) 从矩阵结构上看，矩阵一个是阶数为 n 的对称矩阵，并且稀疏度很高。由于网络中节点间的联系并不是非常紧密，很多节点相互之间并未连接，若不考虑支路间互感，可认为对应的互导纳值为零。通常来说，网络规模越大，节点数目相对更多，稀疏程度越高。

(3) 从元素数值上看，Y_{ii} 的幅值总是大于 Y_{ij} 的幅值。

2.2.3 电力系统功率方程及节点分类

(1) 电力系统功率方程

如果已知节点的注入电流 I_B 和节点电压 U_B，就可以得到相对简洁并可直接求解的线性方程组。但在工程中，往往给定的是各节点功率 S_B。因此，在实际应用

计算时，对方程需做出如下修改：

$$\boldsymbol{Y}_B\boldsymbol{U}_B=\left[\frac{\tilde{\boldsymbol{S}}_B}{\boldsymbol{U}_B}\right]^* \tag{2.5}$$

把式(2.5)展开，得到潮流计算方程：

$$\Delta P_i+\mathrm{j}\Delta Q_i=P_{is}+\mathrm{j}Q_{is}-\dot{U}_i\sum_{j=1}^{n}\dot{Y}_{ij}^{*}\dot{U}_j^{*}\quad(i=1,2,\cdots,n) \tag{2.6}$$

式(2.6)即为潮流计算的功率方程，其中 P_i 为节点有功功率，Q_i 为节点无功功率，$\dot{U}_i$、$\dot{U}_j$ 为节点 i、j 的电压相量。此方程组有如下特点：

1) 它是一组代数方程；

2) 它是一组复数表示的非线性方程，因此适合使用迭代法求解；

3) 方程中的复数电压、导纳都存在两种常用的表示方式：直角坐标和极坐标。因而潮流计算的功率方程也可以用这两种形式分别表示。

将式(2.6)这一复数方程按实部和虚部分别展开得到 $2n$ 个实数方程即 n 个有功功率方程和 n 个无功功率方程。每个节点对应 6 个变量，分别是负荷所消耗的有功功率 P_{Li} 和无功功率 Q_{Li}；发电机节点注入的有功功率 P_{Gi} 和无功功率 Q_{Gi}；各节点电压幅值 U_i 和相位 δ_i（以极坐标表示时）。

各功率之间的关系为

$$\left.\begin{aligned}P_{is}&=P_{Gi}-P_{Li}\\Q_{is}&=Q_{Gi}-Q_{Li}\end{aligned}\right\} \tag{2.7}$$

负荷即用户所消耗的功率通常是不可控的，当这些变量出现预计外的变化时，会导致系统运行状态的偏离，因此称之为不可控变量。发电机发出的功率是可以人为控制的一种自变量，因此被称作控制变量。母线或节点电压的幅值及相角的大小由控制变量所决定，这两个变量同时也是系统最终状态的反映，所以称之为状态变量。

对于 n 个节点的复杂系统，变量数目为 $6n$，其中不可控变量、状态变量、控制变量分别为 $2n$ 个。在给定相应的 $4n$ 个变量后，看似可利用 $2n$ 个方程进行求解 $2n$ 个变量，但其实不然。在式(2.6)所示的功率方程中，电压相位角用相对值来表示的，如果变化量相同时，就无法求解绝对相位角及其功率损耗。因此，要对已知变量做出如下调整：

1) 对于 n 节点系统，通常已知的是 $2(n-1)$个控制变量 P_{Gi}、Q_{Gi}，用剩下的一对控制变量来平衡各节点功率以及各线路损耗。

2) 系统运算时，通常选取一对状态变量电压幅值 U_S、电压相角 δ_S作为参考变量，δ_S设为 0，U_S取标么值为 1.0。这样可对剩余$(n-1)$对状态变量进行求解，并可得到绝对相位角。

(2) 电力系统节点分类

系统中的节点按照给定变量的不同可分为 3 类：

1) PQ 节点

PQ 节点是系统中数目较多的一类节点。此类节点的已知量是发电机向节点输出的功率 P_{Gi}、Q_{Gi}，又由于负荷消耗的功率已知，可直接得到节点的注入功率，再根据功率方程求解状态变量电压的幅值和相位角$(U_i，\delta_i)$。这类节点一般包括发电功率固定的发电厂以及无其他电源的变电所母线。

2) PV 节点

PV 节点在系统中只是少量存在，某些情况下可能没有。这类节点已知的是发电机向节点输出出的有功功率 P_{Gi} 以及负荷消耗的有功功率 P_{Li}（相当于已知了该节点的注入有功功率 P_i），此外，还给定了该节点的电压幅值 U_i和负荷消耗的无功功率 Q_{Li}。功率方程的待求量则变为各节点注入的无功功率 Q_i以及电压的相角 δ_i。PV 节点的注入有功功率一般是已知的且具备较为充足的无功储备，可根据系统实际需求对系统的无功功率进行调节以此来维持节点电压值的恒定。PV 节点通常选择无功储备充足的发电厂或者装设无功补偿设备的变电所母线。

3) 平衡节点

在最终潮流分布完成之前，网损无法给出，需要设一个节点来平衡系统的有功功率，此节点被称作平衡节点。平衡节点在潮流计算中非常重要且只设一个。该节点的已知量包括负荷的等值功率 P_{Li}、Q_{Li}，节点电压的幅值及相角 U_i，δ_i，待求量则是节点注入功率 P_i、Q_i。主调频发电厂母线通常被选作平衡节点。

通过方程求解得到计算结果只是满足了数学要求的一组解，若要具有工程上的实际意义，各变量还需满足一定的约束条件：

① U_i需满足的约束条件，$U_{imin}<U_i<U_{imax}$，这一约束条件能够使各节点尤其是 PQ 节点的电压幅值保持在额定电压附近，从而保证整个系统具有较好的电压质

量。

② 一些节点的电压相位差的约束条件为：$|\delta_i-\delta_j| < |\delta_i-\delta_j|_{\max}$，该约束条件要求线路首末两节点之间的相位差应小于某一数值，从而保证电力系统运行稳定。

③ 控制变量的约束条件为：

$$\left.\begin{aligned} P_{Gi\min} \le P_{Gi} \le P_{Gi\max} \\ Q_{Gi\min} \le Q_{Gi} \le Q_{Gi\max} \end{aligned}\right\} (i=1,2,3\cdots,n) \tag{2.8}$$

式(2.8)中的 $P_{Gi\min}$、$P_{Gi\max}$、$Q_{Gi\min}$、$Q_{Gi\max}$ 是由发电机的运行极限和原动机出力所受到的约束共同决定的。

2.3 牛顿法潮流计算

牛顿法是解决非线性方程的经典算法，该算法首先将非线性方程线性化，然后通过迭代的方式，不断逼近真实解，获得满足收敛条件的解。牛顿法目前在潮流计算中被广泛地采用，根据节点电压方程坐标形式不同可以分成极坐标牛顿法、直角坐标牛顿法，本文运用的是极坐标牛顿法。

2.3.1 牛顿法基本原理

设待求 n 维非线性方程组为

$$\left.\begin{aligned} f_1(x_1,x_2,\cdots,x_n) &= y_1 \\ f_2(x_1,x_2,\cdots,x_n) &= y_2 \\ &\vdots \\ f_n(x_1,x_2,\cdots,x_n) &= y_n \end{aligned}\right\} \tag{2.9}$$

假设此方程组的近似解为 $x_1^{(0)}$，$x_2^{(0)}$，$\cdots$，$x_n^{(0)}$，与真实解的误差设为 $\Delta x_1^{(0)}$，$\Delta x_2^{(0)}$，$\cdots$，$\Delta x_n^{(0)}$，将两种变量代入式(2.9)可得

$$\left.\begin{aligned} f_1(x_1^{(0)}+\Delta x_1^{(0)}, x_2^{(0)}+\Delta x_2^{(0)},\cdots, x_n^{(0)}+\Delta x_n^{(0)}) &= y_1 \\ f_2(x_1^{(0)}+\Delta x_1^{(0)}, x_2^{(0)}+\Delta x_2^{(0)},\cdots, x_n^{(0)}+\Delta x_n^{(0)}) &= y_2 \\ &\vdots \\ f_n(x_1^{(0)}+\Delta x_1^{(0)}, x_2^{(0)}+\Delta x_2^{(0)},\cdots, x_n^{(0)}+\Delta x_n^{(0)}) &= y_n \end{aligned}\right\} \tag{2.10}$$

将式(2.10)在 $x_1^{(0)}$，$x_2^{(0)}$，$\cdots$，$x_n^{(0)}$ 处进行泰勒级数展开，以第一个方程为例展开

$$f_1(x_1^{(0)}+\Delta x_1, x_2^{(0)}+\Delta x_2,\cdots, x_n^{(0)}+\Delta x_n) = f_1(x_1^{(0)}, x_2^{(0)},\cdots, x_n^{(0)}) + \left.\frac{\partial f_1}{\partial x_1}\right|_0 \Delta x_1 + \left.\frac{\partial f_1}{\partial x_2}\right|_0 \Delta x_2 + \cdots + \left.\frac{\partial f_1}{\partial x_n}\right|_0 \Delta x_n + \phi$$

式中各项均可由 $x_1^{(0)}$, $x_2^{(0)}$, …, $x_n^{(0)}$ 代入求得，ϕ 表示的是 Δx_1， Δx_2, …, Δx_n 的高次方与函数 f_1 的高阶偏导的乘积。考虑到方程近似解 $x_i^{(0)}$ 与精确解之间相差很小，故可省去相应的高次方项仅保留常数项和一次项，可得

$$\left.\begin{aligned}
&f_1(x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)}) + \left[\left.\frac{\partial f_1}{\partial x_1}\right|_0 \Delta x_1^{(0)} + \left.\frac{\partial f_1}{\partial x_2}\right|_0 \Delta x_2^{(0)} + \cdots + \left.\frac{\partial f_1}{\partial x_n}\right|_0 \Delta x_n^{(0)}\right] = y_1 \\
&f_2(x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)}) + \left[\left.\frac{\partial f_2}{\partial x_1}\right|_0 \Delta x_1^{(0)} + \left.\frac{\partial f_2}{\partial x_2}\right|_0 \Delta x_2^{(0)} + \cdots + \left.\frac{\partial f_2}{\partial x_n}\right|_0 \Delta x_n^{(0)}\right] = y_2 \\
&\qquad\qquad \vdots \\
&f_n(x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)}) + \left[\left.\frac{\partial f_n}{\partial x_1}\right|_0 \Delta x_1^{(0)} + \left.\frac{\partial f_n}{\partial x_2}\right|_0 \Delta x_2^{(0)} + \cdots + \left.\frac{\partial f_n}{\partial x_n}\right|_0 \Delta x_n^{(0)}\right] = y_n
\end{aligned}\right\} \quad (2.11)$$

式(2.11)称作修正方程，对原来的方程进行了线性化处理，式(2.11)所示方程组可用矩阵形式表示如下

$$\begin{bmatrix} y_1 - f_1(x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)}) \\ y_2 - f_2(x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)}) \\ \vdots \\ y_n - f_n(x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)}) \end{bmatrix} = \begin{bmatrix} \left.\frac{\partial f_1}{\partial x_1}\right|_0 & \left.\frac{\partial f_1}{\partial x_2}\right|_0 & \cdots & \left.\frac{\partial f_1}{\partial x_n}\right|_0 \\ \left.\frac{\partial f_2}{\partial x_1}\right|_0 & \left.\frac{\partial f_2}{\partial x_2}\right|_0 & \cdots & \left.\frac{\partial f_2}{\partial x_n}\right|_0 \\ \vdots & \vdots & & \vdots \\ \left.\frac{\partial f_n}{\partial x_1}\right|_0 & \left.\frac{\partial f_n}{\partial x_2}\right|_0 & \cdots & \left.\frac{\partial f_n}{\partial x_n}\right|_0 \end{bmatrix} \begin{bmatrix} \Delta x_1^{(0)} \\ \Delta x_2^{(0)} \\ \vdots \\ \Delta x_n^{(0)} \end{bmatrix} \quad (2.12)$$

式(2.12)可缩写为

$$\Delta \boldsymbol{f}(x^{(0)}) = \boldsymbol{J}_0^{(0)} \Delta \boldsymbol{x}^{(0)} \quad (2.13)$$

式中 $\boldsymbol{J}_0$ 是雅可比矩阵，$\Delta \boldsymbol{f}$ 是由不平衡量组成的列向量，$\Delta \boldsymbol{x}$ 是由修正量组成的列向量。

通过式(2.12)的线性方程组可求出首次迭代计算的修正量 $\Delta x_1^{(0)}$, $\Delta x_2^{(0)}$, … $\Delta x_n^{(0)}$，求出以后便可对初值进行修正

$$\left.\begin{aligned}
x_1^{(1)} &= x_1^{(0)} + \Delta x_1^{(0)} \\
x_2^{(1)} &= x_2^{(0)} + \Delta x_2^{(0)} \\
&\vdots \\
x_n^{(1)} &= x_n^{(0)} + \Delta x_n^{(0)}
\end{aligned}\right\} \quad (2.14)$$

通过式(2.14)对初值修正完成以后便可得到一个新的近似解，将这个解再次代入到修正方程中并求得 Δf、Δx 各元素的新值，并根据式(2.14)修正，如此循环，便可得到足够精确的解

$$\left.\begin{array}{l}\max\left\{\left|\Delta f(x^{(k)})\right|\right\}<\varepsilon_1\\ \text{或}\ \max\left\{\left|\Delta x_i^{(k)}\right|\right\}<\varepsilon_2\end{array}\right\} \tag{2.15}$$

式(2.15)中的 ε_1、ε_2 为要求的精度，满足式(2.15)后，说明 $x^{(k)}$是符合计算精度要求的解，便可终止迭代计算。但仍需说明的一点是，运用牛顿法进行潮流计算时，对初值的要求较高，只有在所选初值比较接近精确值的情况下，收敛效果才能理想。

2.3.2 极坐标牛顿法潮流计算

运用牛顿法进行潮流计算时，对于节点电压，根据坐标形式不同，有两种表示方法：极坐标法和直角坐标法。下面将对极坐标牛顿法进行详细介绍。

先将节点电压以极坐标的形式进行表示

$$\left.\begin{array}{l}\dot{U}_i=U_i e^{\mathrm{j}\delta_i}=U_i(\cos\delta_i+\mathrm{j}\sin\delta_i)\\ \dot{U}_j=U_j e^{\mathrm{j}\delta_j}=U_j(\cos\delta_j+\mathrm{j}\sin\delta_j)\end{array}\right\}\quad i=(1,2,\cdots n) \tag{2.16}$$

式中 U_i、U_j 分别为节点 i、j 的电压幅值，δ_i、δ_j 分别为节点 i、j 的电压相角。

各节点导纳可表示为 $Y_{ij}=G_{ij}+\mathrm{j}B_{ij}$。将式(2.4)、(2.16)代入到式(2.6)表示的功率方程中，可得如下方程：

$$\Delta P_i+\mathrm{j}\Delta Q_i=P_{is}+\mathrm{j}Q_{is}-U_i e^{\mathrm{j}\delta_i}\sum_{j=1}^{n}(G_{ij}-\mathrm{j}B_{ij})U_j e^{-\mathrm{j}\delta_j} \tag{2.17}$$

将式(2.17)按照有功功率和无功功率分别表示，可得

$$\left.\begin{array}{l}\Delta P_i=P_{is}-U_i\sum_{j=1}^{n}U_j(G_{ij}\cos\delta_{ij}+B_{ij}\sin\delta_{ij})=0\\ \Delta Q_i=Q_{is}-U_i\sum_{j=1}^{n}U_j(G_{ij}\sin\delta_{ij}-B_{ij}\cos\delta_{ij})=0\end{array}\right\}\quad i=(1,2\cdots n) \tag{2.18}$$

式中 $\delta_{ij}=\delta_i-\delta_j$。

式(2.18)即为以极坐标表示的潮流方程，各节点的电压幅值 U_i、相角 δ_i 成为潮流方程需要求解的量。

为方便说明潮流计算的过程，进行如下约定：

(1) 设系统节点数目为 n 个；

(2) 设 PQ 节点共有 m 个，编号为 1，2，3，…，m；

(3) PV 节点数目 $n-1-m$ 个，编号为 $m+1$，$m+2$，…，$n-1$；

(4) 设平衡节点为第 n 节点。

式(2.18)中共包含了 n–1 个有功功率方程：

$$\left.\begin{aligned}
&\Delta P_1 = P_{1s} - U_1\sum_{j=1}^{j=n} U_j(G_{1j}\cos\delta_{1j} + B_{1j}\sin\delta_{1j}) = 0\\
&\Delta P_2 = P_{2s} - U_2\sum_{j=1}^{j=n} U_j(G_{2j}\cos\delta_{2j} + B_{2j}\sin\delta_{2j}) = 0\\
&\vdots\\
&\Delta P_{n-1} = P_{(n-1)s} - U_{n-1}\sum_{j=1}^{j=n} U_j(G_{n-1,j}\cos\delta_{n-1,j} + B_{n-1,j}\sin\delta_{n-1,j}) = 0
\end{aligned}\right\} i = (1,2\cdots n-1) \quad (2.19)$$

以及 m 个无功功率方程：

$$\left.\begin{aligned}
&\Delta Q_1 = Q_{1s} - U_1\sum_{j=1}^{j=n} U_j(G_{1j}\sin\delta_{1j} - B_{1j}\cos\delta_{1j}) = 0\\
&\Delta Q_2 = Q_{2s} - U_2\sum_{j=1}^{j=n} U_j(G_{2j}\sin\delta_{2j} - B_{2j}\cos\delta_{2j}) = 0\\
&\vdots\\
&\Delta Q_m = Q_{ms} - U_m\sum_{j=1}^{j=n} U_j(G_{mj}\sin\delta_{mj} - B_{mj}\cos\delta_{mj}) = 0
\end{aligned}\right\} i = (1,2\cdots m) \quad (2.20)$$

将牛顿法引入潮流计算，形成修正方程如下：

$$\begin{pmatrix}\Delta P_1\\ \Delta P_2\\ \vdots\\ \Delta P_{n-1}\\ \Delta Q_1\\ \Delta Q_2\\ \vdots\\ \Delta Q_m\end{pmatrix} = \left(\begin{array}{cccc:cccc}
H_{11} & H_{12} & \cdots & H_{1,n-1} & N_{11} & N_{12} & \cdots & N_{1m}\\
H_{21} & H_{22} & \cdots & H_{2,n-1} & N_{21} & N_{22} & \cdots & N_{2m}\\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots\\
H_{n-1,1} & H_{n-1,2} & \cdots & H_{n-1,n-1} & N_{n-1,1} & N_{n-1,2} & \cdots & N_{n-1,m}\\
\hdashline
J_{11} & J_{12} & \cdots & J_{1,n-1} & L_{11} & L_{12} & \cdots & L_{1m}\\
J_{21} & J_{22} & \cdots & J_{2,n-1} & L_{21} & L_{22} & \cdots & L_{2m}\\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots\\
J_{m1} & J_{m2} & \cdots & J_{m,n-1} & L_{m1} & L_{m2} & \cdots & L_{mm}
\end{array}\right)\begin{pmatrix}\Delta\delta_1\\ \Delta\delta_1\\ \vdots\\ \Delta\delta_{n-1}\\ \Delta U_1/U_1\\ \Delta U_2/U_2\\ \vdots\\ \Delta U_m/U_m\end{pmatrix} \quad (2.21)$$

式(2.21)中的系数矩阵就是雅可比矩阵，矩阵的各元素表示如下：

$$\left.\begin{aligned} H_{ij} &= \frac{\partial \Delta P_i}{\partial \delta_j} = -U_i U_j (G_{ij}\sin\delta_{ij} - B_{ij}\cos\delta_{ij}) \\ N_{ij} &= \frac{\partial \Delta P_i}{\partial U_j} U_j = -U_i U_j (G_{ij}\cos\delta_{ij} + B_{ij}\sin\delta_{ij}) \\ J_{ij} &= \frac{\partial \Delta Q_i}{\partial \delta_j} = U_i U_j (G_{ij}\cos\delta_{ij} + B_{ij}\sin\delta_{ij}) \\ L_{ij} &= \frac{\partial \Delta Q_i}{\partial U_j} U_j = -U_i U_j (G_{ij}\sin\delta_{ij} - B_{ij}\cos\delta_{ij}) \end{aligned}\right\} \quad (i \neq j) \tag{2.22}$$

$$\left.\begin{aligned} H_{ii} &= U_i^2 B_{ii} + Q_i \\ N_{ii} &= -U_i^2 G_{ii} - P_i \\ J_{ii} &= U_i^2 G_{ii} - P_i \\ L_{ii} &= U_i^2 B_{ii} - Q_i \end{aligned}\right\} \quad (i = j) \tag{2.23}$$

式(2.21)可用分块矩阵的形式表示为

$$\begin{bmatrix} \Delta \boldsymbol{P} \\ \Delta \boldsymbol{Q} \end{bmatrix} = \begin{bmatrix} \boldsymbol{H} & \boldsymbol{N} \\ \boldsymbol{J} & \boldsymbol{L} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{\delta} \\ \Delta \boldsymbol{U} / \boldsymbol{U} \end{bmatrix} \tag{2.24}$$

通过式(2.24)可求出修正量并对电压幅值和相角进行如下修正

$$\left.\begin{aligned} U_i^{(k+1)} &= U_i^{(k)} + \Delta U_i^{(k)} (i = 1,2,\cdots,m) \\ \delta_i^{(k+1)} &= \delta_i^{(k)} + \Delta \delta_i^{(k)} (i = 1,2,\cdots,n-1) \end{aligned}\right\} \tag{2.25}$$

在建立修正方程时，雅可比矩阵的形成是非常重要的一步，通过式(2.22)和(2.23)可以看出，矩阵各元素是通过相关节点的电压相量来表示的。随着电压变量的不断修正，各元素也会发生相应变化，所以每进行一次迭代，都要对雅可比矩阵各元素更新一次。

牛顿法对初值要求较高，若取值不当，收敛性就会变差，为使其保持良好的收敛性，常采用 0、1 启动法（也称作平启动法）即设各节点电压幅值为额定值，各节点相位角为 0。潮流计算多是在电网稳态下进行，其电压幅值一般在额定值附近波动，以标么值表示，则各节点电压的幅值在 1 附近，相位角接近为 0。

2.3.2 极坐标牛顿法潮流计算的基本步骤

修正方程的建立是潮流计算的核心问题。采用极坐标牛顿法进行潮流计算时，步骤如下：

(1) 输入原始数据：① 各节点基本数据：节点数目、收敛精度；② 支路导纳

值；③ PV 节点的电压幅值；④ 平衡节点电压幅值、相角(U,δ)；⑤ PV、PQ 节点注入的有功功率及 PQ 节点注入的无功功率。

(2) 形成节点导纳矩阵 Y_B。

(3) 设各节点电压初值 $U^{(0)}$、$\delta^{(0)}$。

(4) 迭代次数 k 置 0。

(5) 将当前电压值代入式(2.19)、(2.20)计算功率的不平衡量 $\Delta P^{(k)}$、$\Delta Q^{(k)}$。

(6) 根据式(2.22)、(2.23)求雅可比矩阵各元素的值。

(7) 解修正方程(2.21)得到修正量 $\Delta U^{(k)}$、$\delta^{(k)}$的值。

(8) 节点电压进行修正 $U^{(k+1)}=U^{(k)}+\Delta U^{(k)}$、$\delta^{(k+1)}=\delta^{(k)}+\Delta\delta^{(k)}$。

(9) 判断是否符合收敛要求 $\max\left\{\left|\Delta \boldsymbol{P}^{(k)}、\Delta \boldsymbol{Q}^{(k)}\right|\right\}<\varepsilon$，如果不能满足收敛要求则需返回至步骤(5)进行新一轮迭代。

(10) 根据求得的电压幅值、相角计算平衡节点的功率、PV 节点的无功功率和各线路上的损耗，求出完整的潮流分布。

根据上述步骤可得流程图如图 2.4 所示。

2.4 电力网络方程的解法

运用牛顿法形成了线性方程，对此方程的求解需要利用线性方程组的相关求解方法。直接计算法是应用较为普遍的一种方法，其速度快、收敛性好。因此，在潮流计算时采用更多直接计算法，直接计算法主要通过代入消去的方式对线性方程化简。

高斯消去法作为一种经典方法在求解线性方程时效果较好，它的优点是求解速度快，收敛性好，以它为基础改进的三角分解法、选主元素消去法至今仍在广泛使用。高斯消去法按照计算格式的不同可分为按行消去和按列消去，两种格式之间并无实质性差别。

2.4.1 按列消元按列回代的算法

设有 n 阶线性方程组 $\boldsymbol{AX}=\boldsymbol{F}$ 展开如下

$$\left.\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = f_1 \\ &a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = f_2 \\ &\cdots \\ &a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = f_n \end{aligned}\right\} \tag{2.26}$$

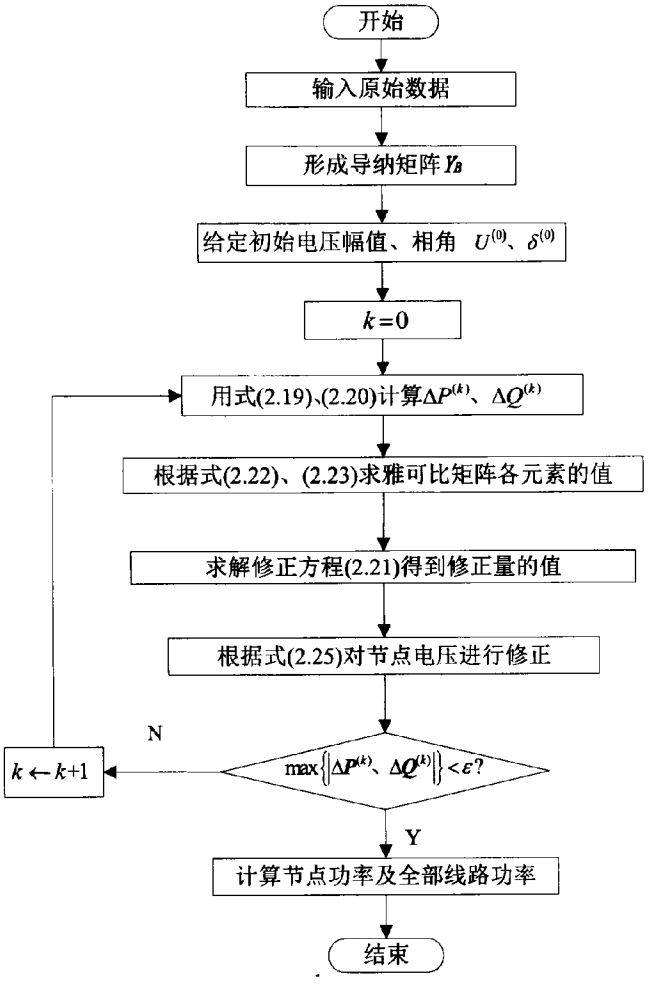

图 2.4 极坐标系牛顿法潮流计算流程图

Fig.2.4The flow chart of Newton power flow in polar coordinate system

消去法求解线性方程的基本思想是通过行初等变换将原系数矩阵化为上三角矩阵，从而将原来的复杂方程变成简单方程进行求解，并通过回代来求出各变量的值。整个消去运算主要针对 $\boldsymbol{F}$ 和系数矩阵 $\boldsymbol{A}$ 进行，因此为描述方便，将列向量 $\boldsymbol{F}$ 记作 $f_i = a_{i,n+1}$ 并与系数矩阵合并为一个增广矩阵 $\boldsymbol{B}$

$$\boldsymbol{B}=[\boldsymbol{A},\boldsymbol{F}]=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & a_{1,n+1} \\ a_{21} & a_{22} & \cdots & a_{2n} & a_{2,n+1} \\ \cdots & \cdots & \ddots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & a_{n,n+1} \end{bmatrix} \tag{2.27}$$

方程组求解的步骤具体如下

(1) 若 $a_{11}\neq 0$，首先计算系数比

$$a_{1j}^{(1)}=a_{1j}/a_{11} \qquad j=1,2,3,\cdots,n+1 \tag{2.28}$$

按式(2.28)对第一行的元素处理后，第一个元素变为 1，对增广矩阵第一列其他元素依次消去

$$a_{ij}^{(1)}=a_{ij}-a_{ij}a_{11} \qquad j=1,2,\cdots,n+1;i=2,3,\cdots,n \tag{2.29}$$

式(2.29)相当于消去了方程组(2.26)中第 i 行的未知量 x_1，第一步完成后可得到如下的增广矩阵

$$\boldsymbol{B}_1=[\boldsymbol{A}_1,\ \boldsymbol{F}_1]=\begin{bmatrix} 1 & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & a_{1,n+1}^{(1)} \\ & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & a_{2,n+1}^{(1)} \\ & \cdots & \cdots & \cdots & \cdots \\ & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & a_{n,n+1}^{(1)} \end{bmatrix} \tag{2.30}$$

(2) 依次类推，进行第 $k-1$ 次消元后可得到增广矩阵为

$$\boldsymbol{B}_{k-1}=[\boldsymbol{A}_{k-1},\ \boldsymbol{F}_{k-1}]=\begin{bmatrix} 1 & a_{12}^{(1)} & \cdots & a_{1k}^{(1)} & \cdots & a_{1n}^{(1)} & a_{1,n+1}^{(1)} \\ & 1 & \cdots & a_{2k}^{(2)} & \cdots & a_{2n}^{(2)} & a_{2,n+1}^{(2)} \\ & & \ddots & \vdots & & \vdots & \vdots \\ & & & a_{kk}^{(k-1)} & \cdots & a_{kn}^{(k-1)} & a_{k,n+1}^{(k-1)} \\ & & & \vdots & & \vdots & \vdots \\ & & & a_{nk}^{(k-1)} & \cdots & a_{nn}^{(k-1)} & a_{n,n+1}^{(k-1)} \end{bmatrix} \tag{2.31}$$

按照如下递推公式继续消去，当进行第 k 次消元时

$$\left.\begin{aligned} a_{kj}^{(k)}&=a_{kj}^{(k-1)}/a_{kk}^{(k-1)} \\ a_{ij}^{(k)}&=a_{ij}^{(k-1)}-a_{ik}^{(k-1)}a_{kj}^{(k)} \quad i=k+1,\cdots,n \end{aligned}\right\} j=k,k+1,\cdots n+1 \tag{2.32}$$

运用计算机编程时，在每次循环时，需要使用新值计算，而原值没有存在意义，可将其覆盖。因此，可将递推公式调整如下

$$\left.\begin{aligned} a_{kj}&=a_{kj}/a_{kk} \\ a_{ij}&=a_{ij}-a_{ik}a_{kj} \quad i=k+1,\cdots,n \end{aligned}\right\} j=k+1,\cdots n+1 \tag{2.33}$$

(3) 直至消去完成后可得到矩阵如下

$$\boldsymbol{B}_n=[\boldsymbol{A}_n,\ \boldsymbol{F}_n]=\begin{bmatrix} 1 & a_{1,2}^{(1)} & a_{1,3}^{(1)} & \cdots & a_{1,n}^{(1)} & a_{1,n+1}^{(1)} \\ & 1 & a_{2,3}^{(2)} & \cdots & a_{2,n}^{(2)} & a_{2,n+1}^{(2)} \\ & & 1 & \cdots & a_{3,n}^{(3)} & a_{3,n+1}^{(3)} \\ & & & \ddots & \vdots & \vdots \\ & & & & 1 & a_{n,n+1}^{(n)} \end{bmatrix} \tag{2.34}$$

上述过程称为消元过程，如要得到最终方程解，还需对方程组进行回代，设 m 为回代次数，公式如下

$$a_{i,n+1}^{(n+i-j+1)}=a_{i,n+1}^{(n+i-j)}-a_{ij}^{(i)}a_{j,n+1}^{(n)}\ i=j-1,\cdots,1;m=n-j+1;j=n,n-1,\cdots,2 \tag{2.35}$$

在使用计算机编程时，递推公式可表示为

$$a_{i,n+1}=a_{i,n+1}-a_{ij}a_{j,n+1}\qquad i=j-1,\cdots,1;j=n,n-1,\cdots,2 \tag{2.36}$$

2.4.2 按行消元按行回代的算法

在电力网络方程求解时，另一种比较常见的计算形式是逐行进行消元，其具体步骤如下：

(1) 若 $a_{11}\neq 0$，首先对其按照式(2.28)对矩阵 $\boldsymbol{B}$ 第一行进行规格化，然后按照下式对第二行进行消元并规格化

$$a_{2j}^{(1)}=a_{2j}-a_{21}a_{1j}^{(1)}\qquad j=1,2,\cdots,n+1 \tag{2.37}$$

$$a_{2j}^{(2)}=a_{2j}^{(1)}/a_{22}^{(1)}\qquad j=2,3,\cdots,n+1 \tag{2.38}$$

按照式(2.37)、式(2.38)进行处理后，增广矩阵变化如下

$$\boldsymbol{B}_2=[\boldsymbol{A}_2,\boldsymbol{F}_2]=\begin{bmatrix} 1 & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & a_{1,n+1}^{(1)} \\ & 1 & \cdots & a_{2n}^{(2)} & a_{2,n+1}^{(2)} \\ a_{31} & a_{32} & \cdots & a_{3n} & a_{3,n+1} \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & a_{n,n+1} \end{bmatrix} \tag{2.39}$$

(2) 将上述消元过程依次进行，对增广矩阵第 i 行的处理主要包含两步：首先作 $i-1$ 次消元，也就是利用前面已规格化处理的 $i-1$ 行矩阵进行依次消元，然后在对第 i 行作一次规格化。以 k 为消元次数，可得计算通式如下

$$a_{ij}^{(k)}=a_{ij}^{(k-1)}-a_{ik}^{(k-1)}a_{kj}^{(k)}\qquad j=k,\cdots,n+1;k=1,\cdots,i-1 \tag{2.40}$$

$$a_{ij}^{(i)} = a_{ij}^{(i-1)} / a_{ii}^{(i-1)} \qquad j = i, \cdots, n+1 \tag{2.41}$$

在进行算法编写时，其递推公式为

$$a_{ij} = a_{ij} - a_{ik} a_{kj} \qquad j = k+1, \cdots, n+1; k = 1, \cdots, i-1 \tag{2.42}$$

$$a_{ij} = a_{ij} / a_{ii} \qquad j = i+1, \cdots, n+1 \tag{2.43}$$

对第 i 行处理完成后，可得矩阵如下

$$\boldsymbol{B}_i = [\boldsymbol{A}_i, \boldsymbol{F}_i] = \begin{bmatrix} 1 & a_{12}^{(1)} & \cdots & \cdots & \cdots & a_{1n}^{(1)} & a_{1,n+1}^{(1)} \\ & 1 & \cdots & \cdots & \cdots & a_{2n}^{(2)} & a_{2,n+1}^{(2)} \\ & & \ddots & \cdots & \cdots & \cdots & \cdots \\ & & & 1 & \cdots & a_{in}^{(i)} & a_{i,n+1}^{(i)} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & & a_{ni} & \cdots & a_{nn} & a_{n,n+1} \end{bmatrix} \tag{2.44}$$

(3) 逐行消去以后便可得到式(2.34)所示的矩阵。

消元完成以后需按照下式进行回代，以求得最终结果

$$x_i = a_{i,n+1}^{(i)} - \sum_{j=i+1}^{n} a_{ij}^{(i)} a_{j,n+1}^{(n)} \qquad i = n-1, \cdots, 1 \tag{2.45}$$

$$a_{i,n+1} = a_{i,n+1} - a_{ij} a_{j,n+1} \qquad j = i+1, \cdots, n; i = n-1, \cdots, 1 \tag{2.46}$$

2.5 小结

本章首先对电力系统潮流计算时需要用到的数学模型进行了介绍，并推导了电力系统功率方程，然后详细介绍了极坐标牛顿法的原理及雅可比矩阵的形成过程，建立了一组线性化的修正方程，最后为求解这组方程又引入了线性方程的解法，完整地描述了潮流计算的数学过程。这是本文后续研究设计极坐标牛顿法的理论基础。

第 3 章 基于循环结构的科研型潮流算法研究

3.1 引言

第 2 章对极坐标牛顿法的计算原理和算法步骤进行了详细介绍，并分析了电力网络方程的解法。本章的主要任务根据第 2 章介绍的数学模型进行算法设计。

程序的设计从本质上说也是算法的设计，一个描述比较详细的算法几乎就是程序的具体体现。而对于好的算法不仅能做到步骤简化、简单明确，更重要的是能利用最小的时间复杂度和空间复杂度来完成所需计算，巧妙的程序流程设计，灵活的循环控制，以及好的搜索方法都可以作为算法设计的方向[57-59]。

影响牛顿法潮流计算速度的过程主要有两个：雅可比矩阵的形成过程和线性方程组的求解过程。本章也是从这两个过程出发分别介绍了一种雅可比矩阵形成方法和 3 种线性方程求解方法，最后运用算例对方法的有效性和实用性进行验证。

3.2 雅可比矩阵的形成

3.2.1 雅可比矩阵的设计思路

雅可比矩阵与导纳矩阵有着相同的矩阵结构，由于导纳矩阵的稀疏度极高，雅可比矩阵也必将高度稀疏。本节对雅可比矩阵中的元素是否为零元素进行判断，如果是零则直接跳过，从而节省了运算时间。

设网络中含 n 个节点，其中 m 个 PQ 节点，一个平衡节点，则导纳矩阵为 $n-1$ 阶，为了便于元素的寻找和修改，本文对雅可比矩阵均按照 $(2n)\times(2n)$ 进行存储。在使用极坐标表示时，对于 PV 节点，电压幅值已经给出，方程和未知量个数均减少 $n-m-1$ 个，雅可比矩阵实际变为 $(n-1+m)$ 阶，若按照 $(2n)\times(2n)$ 进行存储会出现许多空行、空列，如果在循环中能够对其进行跳过，便可节约计算时间。因此，需要根据节点的类型选择跳过剩余零元素，对式(3.1)中的雅可比矩阵元素分析后，可将判断的准则概括如下：

首先对两节点之间节点导纳值进行判断，如果两节点之间的导纳值为 0，则可将节点对应四个元素直接置零，不再进行下述判断；对于元素 H_{ij}，如果满足节点 i、j 都不是平衡节点，则 $H_{ij}\neq 0$；对于元素 N_{ij}，如果满足节点 i 不是平衡节点且 j 是 PQ 节点，则 $N_{ij}\neq 0$；对于元素 J_{ij}，如果满足节点 i 是 PQ 节点且 j 不是平衡节

点，则 $J_{ij} \neq 0$；对于元素 L_{ij}，只有节点 i、j 均为 PQ 节点时才会有 $L_{ij} \neq 0$。

3.2.2 雅可比矩阵的形成步骤

对各元素的判断依据分析后，便可进行程序步骤的设计，具体如下：

(1) 设置 $i=1$；

(2) 判断节点 i 是否为平衡节点，如果是则直接跳过节点 i 所对应的两行元素即跳至步骤(18)；

(3) 设置 $j=1$；

(4) 判断节点 j 是否为平衡节点，如果是，跳至步骤(13)；

(5) 判断节点 i、j 之间的导纳值是否为 0，如果为 0 则跳至步骤(13)；

(6) 按照式(2.22)计算雅可比矩阵元素 H_{ij} 的值；

(7) 判断节点 j 是否为 PQ 节点，如果不是则跳至步骤(9)；

(8) 按照式(2.22)计算雅可比矩阵元素 N_{ij} 的值；

(9) 判断节点 i 是否为 PQ 节点，如果不是则跳至步骤(13)；

(10) 按照式(2.22)计算雅可比矩阵元素 J_{ij} 的值；

(11) 判断节点 j 是否为 PQ 节点，如果不是则跳至步骤(13)；

(12) 按照式(2.22)计算雅可比矩阵元素 L_{ij} 的值；

(13) 令 $j=j+1$；

(14) 判断 j 是否为大于 n，若不是则返回至步骤(4)进行新一轮运算；

(15) 按照式(2.23)对元素 H_{ii} 进行计算；

(16) 判断节点 i 是否为 PQ 节点，如果不是跳至步骤(18)；

(17) 按照式(2.23)对元素 N_{ii}、J_{ii}、L_{ii}，依次进行计算；

(18) 令 $i=i+1$；

(19) 判断 i 是否大于 n，若 i 不大于 n 则返回至步骤(2)进行新一轮运算，否则结束。

依据上述步骤形成的流程图为图 3.1。对于以上过程需要说明的一点是，步骤(6)、(8)、(10)、(12)中对雅可比矩阵中的元素包括分块矩阵对角元素均是按照公式组(2.22)进行初步计算，但是各分块矩阵对角元素 H_{ii}、N_{ii}、J_{ii}、L_{ii} 与非对角元素的计算公式并不相同，在步骤(15)、(17)按照式(2.23)对对角元素进行了计算。

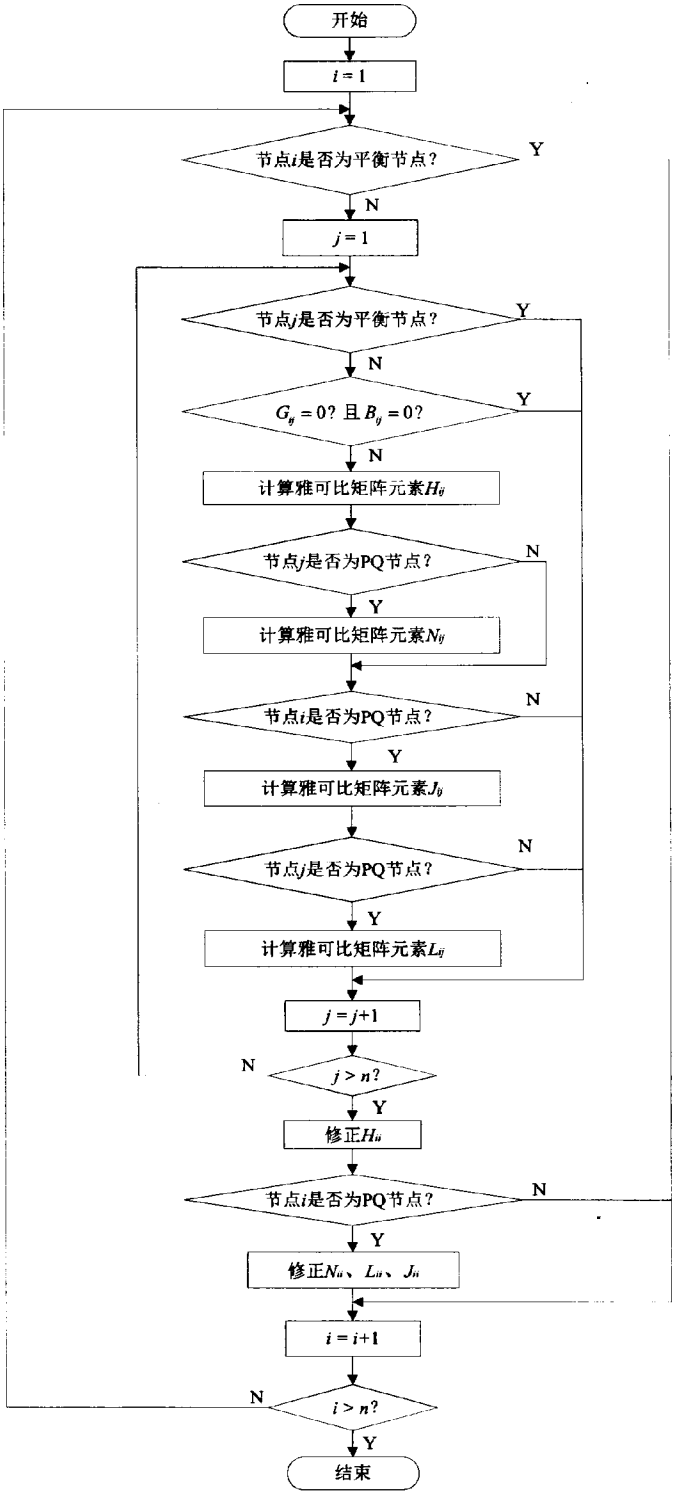

图 3.1 雅可比矩阵形成流程图

Fig.3.1 flow chart of the Jacobian matrix formation

修正过程的选择实际有两种，一种是按照普通的编程思路在按列循环的时候判断 $i=j$ 是否成立，若成立，则进行修正，不成立，则需要继续循环；另一种是上述过程中在步骤(14)之后进行修正，即这种修正是嵌套在按行循环中。显然，第一种需要在每一个小循环即列循环进行判断并修正，耗费的时间比较多，而第二种是在大的循环中进行的，需要的次数少，耗费的时间也少。

通过上述算法形成的雅可比矩阵按照($2n\times 2n$)进行存储，这样会导致内存需求有所增加，但编程难度降低，可维护性好，符合科研使用这一目的。

3.3 线性方程组求解的算法设计

在进行完雅可比矩阵形成过程的设计后要继续进行线性方程求解过程的设计，对于这一过程设计了以下 3 种方案：

(1) 消元过程中，对雅可比矩阵相关元素是否为 0 进行判断，并根据判断结果来设计算法，减少不必要的计算；

(2) 对(1)算法中的雅可比矩阵进行按列存储，其余过程仍然不变，观察改进后的效果；

(3) 利用 Matlab 的矩阵除法进行线性方程求解。

3.3.1 按行运算的高斯法的设计

修正方程的系数矩阵即雅可比矩阵是一个稀疏矩阵，利用其稀疏性减少不必要运算是提高方程求解速度的有效方法。本节从研究修正方程的系数矩阵的元素排列特点出发，增加一些相关元素的判断，减少不必要运算。

在式(3.1)中的分析已经得到用极坐标表示电压相量形成的雅可比矩阵实际变为$(n+m-1)$阶，同样，功率的不平衡量与电压修正量分别变为$(n+m-1)$维的列向量。雅可比矩阵按照$(2n)\times(2n)$存储，这样在存储空间内就有$(n-m+1)$个空行、空列，且空行与空列成对出现，可通过对角元素的判断来确定空行、空列的位置。

设 i 为系统中的一个节点，在雅可比矩阵中对应的对角元素为 H_{ii}、L_{ii}。若 i 为 PQ 节点，$H_{ii}\neq 0$、$L_{ii}\neq 0$，所在行不全为 0；若 i 是 PV 节点则 $H_{ii}\neq 0$、$L_{ii}=0$，L_{ii}所在行、列全为 0；若 i 是平衡节点则$L_{ii}=0$、$H_{ii}=0$，H_{ii}、L_{ii}两元素所在行、列均全为 0。因此可得结论如下：若对角元素为 0，对角元素所在的行、列所有元素皆为 0。

通过以上分析可以知道若一个对角元素为 0，则矩阵减少一行一列，也就是方程个数和未知数的个数分别减少一个。因此，可通过判断对角元素是否为 0 来选择跳过相应的方程的求解和未知数的消去，进而减少不必要运算。此外，对于系数为 0 的未知数也进行跳过。设 i 表示第 i 个方程，k 表示第 k 次消元，j 表示第 j 列，a_{ij} 表示系数矩阵的元素，b_j 表示不平衡量列向量的元素，可设计步骤如下：

(1) 设置 $i=1$。

(2) 判断 a_{ii} 是否为 0，若为 0 直接跳至步骤(19)；

(3) 对消元次数 k 赋初值：$k=1$；

(4) 判断 k 是否小于 i，若 k 不小于 i，则转至步骤(14)；

(5) 判断 a_{kk} 是否为 0，若为 0 跳至步骤(13)；

(6) 判断 a_{ik} 是否为 0，若为 0 跳至步骤(13)；

(7) 设置当前列号 $j=k+1$；

(8) 判断 j 是否大于 n，若 j 大于 n，则跳至步骤(12)；

(9) 判断 a_{kj} 是否为 0，若为 0 直接跳至步骤(11)

(10) 根据式(2.42)进行消去计算；

(11) 令 $j=j+1$，返回到步骤(8)；

(12) 对左端的不平衡量进行消去计算；

(13) 令 $k=k+1$，返回至步骤(4)；

(14) 设置当前的列号 $j=i+1$；

(15) 判断 j 是否大于 n，若 $j>n$，则跳至步骤(18)；

(16) 根据式(2.43)对系数矩阵的进行规格化运算；

(17) 令 $j=j+1$ 返回至步骤(15)；

(18) 根据式(2.43)对左端不平衡量进行规格化；

(19) 令 $i=i+1$；

(20) 判断 $i>n$ 是否成立，若不成立，返回至步骤(2)，否则结束。

通过以上步骤形成的流程图如图 3.2 所示。

3.3.2 按列运算的高斯法的设计

高级语言在进行元素的存储时一般都会规定一定的存储顺序，比如 C 语言是

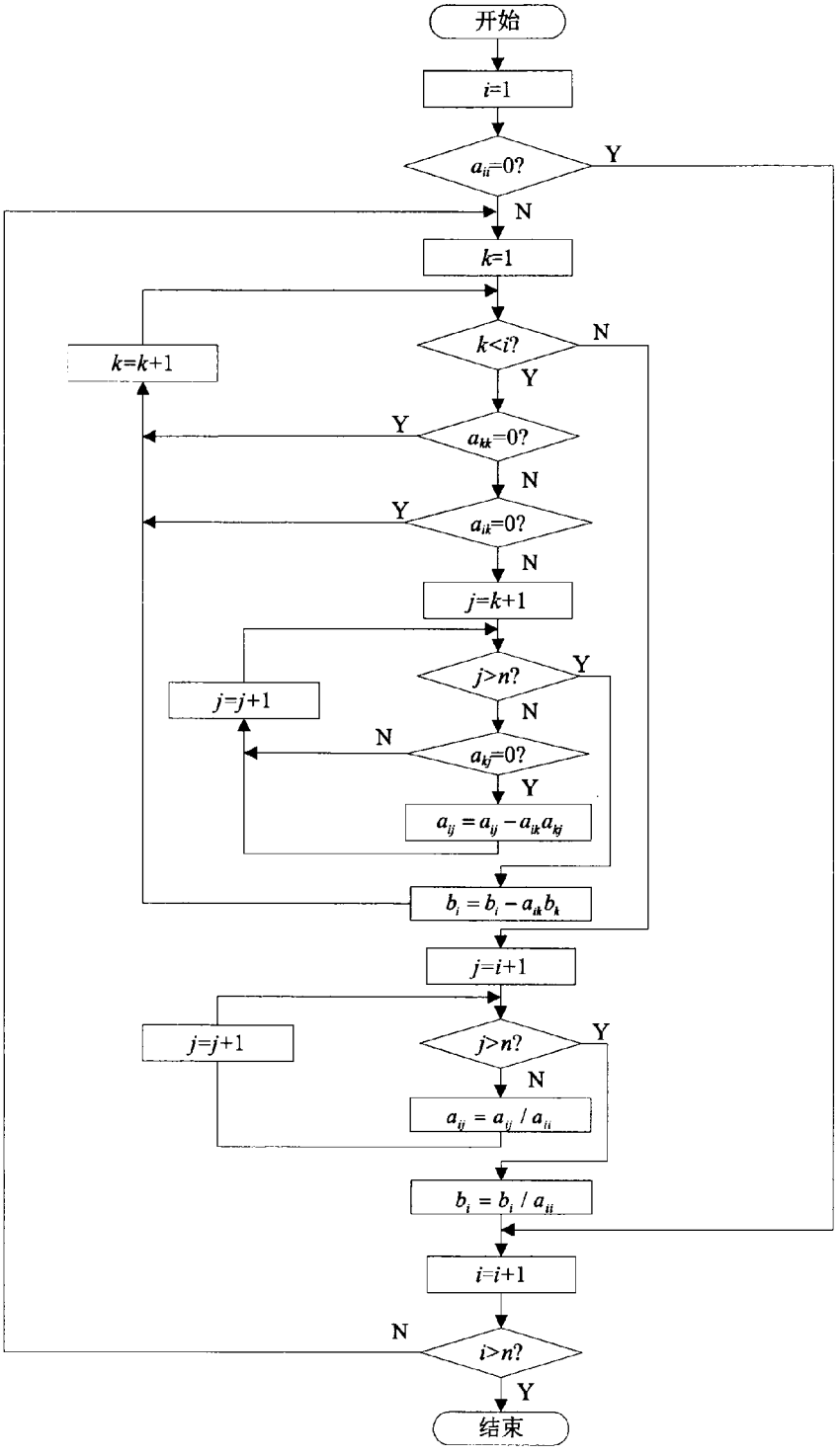

图 3.2 高斯消去法流程图

Fig. 3.2 Flow chart of Gauss elimination

按行的先后顺序进行矩阵或数组元素的存放，即从第一行的第一个元素进行存储，第一行存储完成后再进行下一行存储，依此类推。与 C 语言的存储不同，Matlab 的矩阵元素是按列进行存储的，先存第一列，再存第二列，依此类推。

理清了 Matlab 的矩阵的排列顺序，就可以由此来进行高斯法的改进。在 3.3.1 节设计的高斯法中，所有的循环均是按照每行进行，这样的循环顺序与 Matlab 的存储顺序不一致，在进行数据读取时，减缓了程序计算的速度。为使其速度提高，本节将在 3.3.1 节介绍的高斯法的基础上进行一些改变：

(1) 增加一个转置语句，将系数矩阵转置；

(2) 进行转置后，行列表示符号互换，由原来的行循环嵌套列循环变为列循环嵌套行循环；

(3) 消元公式的下标行列互换。

这样设计的程序更符合 Matlab 的存储方式，会提高高斯法解方程的速度，具体效果会在后面算例计算时体现。

3.3.3 Matlab 的矩阵除法

前面介绍的高斯法的思路都是围绕线性代数中方程解法展开的，在 Matlab 中，矩阵的代数运算非常成熟，对于矩阵的加、减、乘、除以及求矩阵的秩、特征值以及解线性方程都存在相应的函数进行运算，而且编程简单，易于实现。

本小节就是利用 Matlab 的矩阵除法运算函数求解线性方程。除法运算包含了左除和右除两个概念，左除即“A\B=inv(A)*B”，右除即“A/B=A*inv(B)”。这种除法运算是通过对方阵求逆来解方程的，这就要求系数矩阵即雅可比矩阵是一个可逆矩阵，但是本文在存储时采用了$(2n\times 2n)$的存储方式，存在许多空行、空列，无法直接进行求逆运算。为此，首先要对矩阵进行一下预处理，使其变为可求逆的矩阵。

通过算法设计，提取雅可比矩阵中非零元素所在行的标号并组成一个数组。具体的步骤如下：

(1) 定义两个变量 PQ_1、n，其中 PQ_1 是一个数组，对非零行的编号进行暂时存储，n 是节点的个数；

(2) 对数组预定义维数，将数组 PQ_1 定义成一个 $2n\times 1$ 的数组；

(3) 令 $j_1=0$；

(4) 设定 $k=1$；

(5) 判断 k 小于等于 n 是否成立，若不成立直接跳至步骤(10)；

(6) 判断 k 是否为平衡节点，如果是则跳至步骤(9)；

(7) 令 $j_1=j_1+1$；

(8) 令 $PQ_1(j_1)=k$；

(9) 令 $k=k+1$，并转至步骤(5)；

(10) 设定 $k=1$，从雅可比矩阵第一行开始，重新进行新的一轮判断；

(11) 判断 k 小于等于 n 是否成立，若不成立直接跳至步骤(16)；

(12) 判断节点 k 是否为 PQ 节点，如果不是跳至步骤(15)；

(13) 令 $j_1=j_1+1$；

(14) 令 $PQ_1(j_1)=k+n$；

(15) 令 $k=k+1$，并转至步骤(11)；

(16) 保留数组 PQ_1 中前 j_1 个元素

通过上述步骤得到的流程图如图 3.3 所示，在完成了以上步骤以后可以得到一个数组 PQ_1，通过数组所存储的非 0 行标号就可以完成矩阵中非 0 元素的提取，进而就可以利用 Matlab 中的除法运算进行方程的求解。经过上述处理后就可以将线性方程求解过程用一条编程语句“ $dx(PQ_1,1)=J_0(PQ_1,PQ_1)\backslash fx(PQ_1)$ ”来代替，其中“dx”是电压修正量，“fx”是功率的不平衡量，“J_0”为雅可比矩阵。

这样处理实际上也是将高斯消元的循环运算用矩阵运算代替，由于 Matlab 在矩阵运算上非常成熟，所以在速度上会有提高，具体效果会在算例分析时进行说明。

3.4 稀疏技术

在电力网络中，单个节点直接相连的支路数目一般小于 5 条，因此，由这样的网络所确定的导纳矩阵必然是一个高度稀疏的矩阵，在一个庞大的导纳矩阵中，一般来说，每一行非零元素平均不足 6 个[60]。同样，导纳矩阵与雅可比矩阵分块矩阵结构相同，也同样是一个稀疏度很高的矩阵，利用雅可比矩阵的稀疏性一直是提高潮流计算速度的有效方法。根据这一目的，经过研究，逐渐地形成了一套

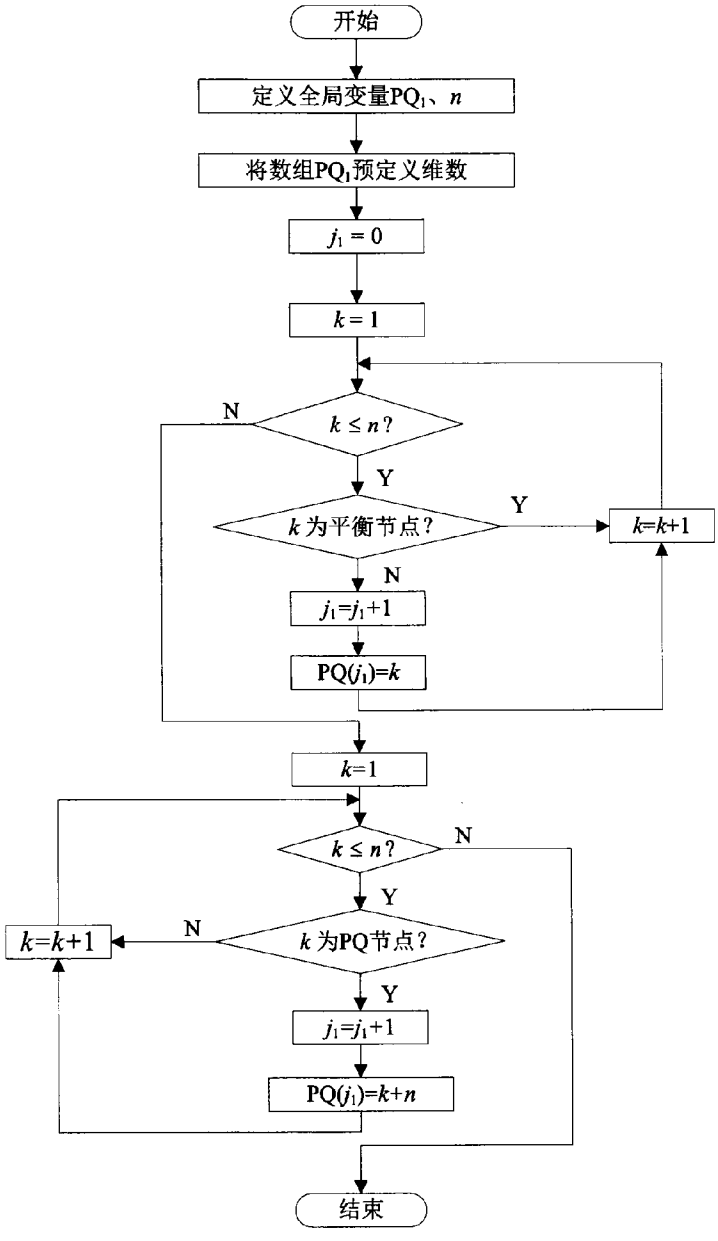

图 3.3 雅可比矩阵非零行标号的数组形成流程图

Fig. 3.3 Flow chart of the array formation of nonzero-line sequence number in Jacobian matrix

适应电力系统计算的稀疏技术。稀疏技术的原理就是要在程序编写和算法设计时尽可能地避免对零元素的存储和计算，包括稀疏矩阵的存储、因子表的形成以及为保持相关矩阵的稀疏性而使用的对节点编号进行优化等技术。

3.4.1 稀疏矩阵存储方式

稀疏矩阵存储方式利用了矩阵稀疏特性对其进行压缩存储。存储时，只对非零元素的数值及其有效的位置信息进行存储，相关计算也只针对非零元素进行。这样可以省略对零元素的存储及运算，节省空间，提高速度。

稀疏技术对 $m\times n$ 阶矩阵 $\boldsymbol{A}$ 有 3 种常用的存储方式，分别是：按坐标存储、按顺序存储以及链表存储。设矩阵 $\boldsymbol{A}$ 中有 b 个非零元素，a_{ij} 为稀疏矩阵中第 i 行第 j 列的非零元素。因此可通过定义相关数组来存储矩阵中非零元素的信息，格式如下：

(1) 按坐标存储

VA[ ]—a_{ij} 的值，总共 b 个；

IA[ ]—a_{ij} 的行号，总共 b 个；

JA[ ]—a_{ij} 的列号，总共 b 个；

按坐标存储是根据非零元素的坐标位置进行存储，方法简单直观、易于搜索，但参与运算不够方便。

(2) 按顺序存储

这种方式以行为顺序存储矩阵中非零元素，同行元素按照顺序排在一块。其格式如下：

VA[ ]—a_{ij} 的值，共 b 个；

IA[ ]—a_{ij} 的列号，共 b 个；

JA[ ]—a_{ij} 记录矩阵 $\boldsymbol{A}$ 中每行第一个非零元素在 $\boldsymbol{A}$ 中的位置，共 $n+1$ 个。

这种存储方式检索也相对容易，但不如坐标存储更为简单直观，但如果进一步分析会发现，顺序存储的方法便于参与运算，因此这种方法目前使用较为普遍。

(3) 链表存储

这种存储格式比按顺序存储多一个数组，即

LINK[ ]—下一个非零元素的位置，共有 b 个，每行的最后一个非零元素，在该位置上置零。

与坐标存储和顺序存储相比，链表存储更有利于所存储矩阵参与运算。

以上介绍的按顺序存储、链表存储皆以按行检索为前提，若使用按列检索，需将“行”看作“列”，将“列”看作“行”。

3.4.2 注入元对算法的影响

本文 2.4 节中介绍了按列、按行消去的线性方程解法，其中按行消去的方法在实用的潮流计算程序中使用更为普遍，但按列消元对应于网络变换时消去节点，物理意义更加明确。因此，为方便以下讨论，本文以按列消元为例进行说明。

在对方程组的系数矩阵进行消去时会有新的非零元素，称之为注入元。以高斯消元法进行逐列消元，在物理意义对应于以消去节点的方法逐一对节点进行消去，而注入元对应消去时出现得新的支路导纳。

首先，注入元的出现影响了系数矩阵的稀疏性。但通过观察可以发现，注入元出现的数量与网络节点编号的顺序，即消元的顺序相关。也就是说，为了不破坏系数矩阵的稀疏性，减少存储开销，需要对节点编号顺序进行优化。

节点编号优化有以下 3 种方法：

(1) 静态优化法—按照节点静态联结支路数的多少编号

这种方法最简单。统计好节点连接支路的数目后，按照由少到多的顺序，依次编号

(2) 半动态优化法—按照节点动态联结支路数的多少编号

这种方法使用较为广泛。运用这种方法时，先找一个联结支路数最少的一个节点编号，并将其消去；随后再次寻找联结支路数最小的节点进行编号，并将其消去；并依次类推。

(3) 动态优化法—按照节点动态增加支路数的多少编号

这种方法不常使用。这种方法使用时不先进行节点编号，而是寻找消元后支路增加最少的节点并编号，然后消去，并依次进行。

其次，注入元的出现会影响稀疏矩阵的存储。由于稀疏矩阵存储是针对非零元素进行的，新的非零元素的注入势必会改变原本设置的稀疏矩阵的存储，同时增加了元素检索和修改的难度。而节点编号优化固然可以大大减少注入元的数量从而尽可能保持矩阵的稀疏性，但会打乱矩阵最初始的排列顺序，影响元素的检索和修改。显然，对于科研型潮流算法而言，这样是非常不利的。

3.4.3 Matlab 的稀疏矩阵函数

通过稀疏技术存储方式的介绍可以看出，直接使用稀疏技术时，需要设计特

定的矩阵存储方法和检索方法，增加了算法的实现难度。在消元过程中，注入元的出现会造成系数矩阵中元素检索和存储的困难。而 Matlab 的稀疏矩阵函数是一个可以直接使用的函数，只是在函数内部进行了稀疏存储，但从使用者的角度来看，并不影响矩阵的修改和矩阵元素的检索，仍然可以像完全矩阵一样用行、列使用矩阵元素。

对于稀疏矩阵，sparse 的参数主要包括以下 3 个：元素值大小、元素的行、列下标，这样的稀疏存储是矩阵处理变得更为简单高效。

下面对稀疏矩阵函数进行说明：

(1) 稀疏矩阵的存储方式

Matlab 通过三个矩阵存储稀疏矩阵的信息：

1) 第一个矩阵存储稀疏矩阵中非 0 元素的值，是浮点型矩阵；

2) 第二个矩阵存储非零元素的行索引，是整型矩阵；

3) 第三个矩阵存储原矩阵中每一列的第一个非零元素在前两个矩阵中的位置，以及最后一列的最后一个非零元素在前两个矩阵中的位置，它也是整型矩阵。

(2) 稀疏矩阵的创建

最常用的语句是“M=sparse(N)”，其含义是将矩阵 N 变为稀疏矩阵进行存储，即通过矩阵 M 存储非零元素及其下标，并返回。

对于对角矩阵的稀疏处理，会使用 spdiags 函数，其用法主要有以下三种：

1) “[M,D] = spdiags(N)”，含义是从矩阵 N 中取出所有非零对角元素，并保存在矩阵 M 中，向量 D 表示非零对角元素的位置；

2) “M = spdiags(N,D)”，含义是从矩阵 N 中取出由 D 指定的对角线元素，并保存在矩阵 M 中；

3) “M = spdiags(N,D,m,n)”，表示形成 $m \times n$ 的稀疏矩阵 M，其元素是由向量 N 中的元素放在由 D 指定的相应位置上得到的。

Matlab 允许完全矩阵与稀疏矩阵的混合运算，并且它们运算产生的结果可以是完全矩阵的形式也可以是稀疏矩阵的形式。

由于本文所研究的科研型潮流算法需要较好的可维护性，因此，对于程序中出现的稀疏矩阵都采用 Matlab 的稀疏矩阵函数进行处理。

3.5 算例分析

为比较各种设计方案，根据本章中介绍的雅可比矩阵形成和线性方程组求解两个过程的算法设计并结合图 2.4 所示流程图进行程序编写，对东北电网 445 节点的实际电力系统进行计算，并根据计算所用时间对算法的效果进行评估。本文中所有算例计算均是在 2.66GHz 的 Intel Core2 的 PC 机上进行的。

东北电网共含有 445 个节点、544 条支路，其中存在很多小阻抗支路，为了对几种算法进行比较，把这些小阻抗支路转换为正常支路以满足各种算法的要求。另外，由于本章中设计了一种循环形成雅可比矩阵的方法，同时介绍了三种线性方程组的解法。根据线性方程求解方法的不同确定了三种方法，为方便说明，编号如下：

方法 1：使用按行运算的高斯法求解线性方程；

方法 2：使用按列运算的高斯法求解线性方程；

方法 3：使用 Matlab 的矩阵除法求解线性方程；

方法 3(s)：对方法 3 使用 Matlab 的稀疏矩阵函数进行优化。

通过以上 4 种方法对算例进行计算得到表 3.1 所示数据，需要说明的是本次计算只对本章设计的雅可比矩阵形成和线性方程求解两个过程进行计时，并不包括数据输入输出等环节所用的时间。

表 3.1 潮流算法计算时间比较

Tab.3.1 The comparison of computing time required by different power flow algorithm

潮流计算方法	计算时间(s)
方法 1	12.303
方法 2	7.991
方法 3	0.916
方法 3(s)	0.540

分析表 3.1，可以得到结论如下：

与方法 1 相比，方法 2 中按列运算的高斯解法更符合 Matlab 的矩阵存储规律，计算速度也更快。当然，与前两种方法相比，Matlab 自带的矩阵除法进行方程求解是一种更为成熟的算法，更适合 Matlab 的运行环境，使得方法 3 计算速度有很

大提高，同时，Matlab 的矩阵除法的调用也非常简单，使程序更加清晰。在使用了 Matlab 的稀疏矩阵函数后，方法 3 的速度也有了较大提高。

3.6 小结

本章设计了一种通过循环形成雅可比矩阵的方法及 3 种求解线性方程的方法，分别是：按行运算的高斯法、按列运算的高斯法、Matlab 的矩阵除法，并将这 3 种方法分别与循环形成雅可比矩阵的方法进行组合组成 3 种不同的潮流计算方法，并对方法 3 采用了稀疏技术，通过计算时间的比较验证了算法设计的可行性与有效性。

第 4 章 基于矩阵运算的科研型潮流算法研究

4.1 引言

第 3 章对雅可比矩阵的形成过程进行了算法设计，并设计了一种基于循环结构的雅可比矩阵形成方法。但在 Matlab 的运行环境下，执行循环的效率较低，因此尽量要借助矩阵和内部函数来实现相关操作，其实就是注重矩阵的整体运算，避免或尽量减少对矩阵中元素的操作。

矩阵运算是 Matlab 特有的一种运算，其他编程语言往往只定义标量的各种运算，Matlab 用矩阵替代了标量定义了一种更广义的运算。在矩阵运算中，以矩阵作为基本元素，而普通标量则被视做矩阵元素或特殊的矩阵。这样，原本复杂的矩阵运算问题就以一种简单的方法解决。

本章主要是针对雅可比矩阵形成过程进行了设计，即用矩阵运算的方式来替代循环。算法设计的思路是，首先将雅可比矩阵分成四个子块，然后通过矩阵运算得到各子块元素的值，最后将四个子块重新组合并去掉无关行从而得到雅可比矩阵。

4.2 矩阵运算形成雅可比矩阵的公式推导

在运用循环的方法形成雅可比矩阵时，更注重的是矩阵单个元素的运算，每进行一次循环，将确定四个下标相同的元素即 H_{ij}、N_{ij}、J_{ij}、L_{ij}。而对于矩阵运算形成雅可比矩阵的过程，更注重矩阵的整体运算，所谓整体运算就是以雅可比矩阵或其矩阵子块作为运算对象，不再局限于矩阵元素的单独处理。因此，将雅可比矩阵划分为如式(2.24)所示的四个子块，对每一个子块分别进行计算，然后再进行组合，这样处理使得编程更为简单。

本节根据 Matlab 的矩阵运算设计了不同的方法来形成雅可比矩阵，依次引入了矩阵相乘运算、点乘运算、重复矩阵函数 3 种方式对矩阵进行处理，形成了 5 种不同的方法。

4.2.1 矩阵相乘运算

使用矩阵运算形成雅可比矩阵时，为方便公式推导，首先需要理清 H_{ij}、N_{ij}、J_{ij}、L_{ij} 四种元素的关系，在依据四种元素划分的四个子块中，非对角元素与对角元

素不同，需要分别表示。其中非对角元素表示如式(2.22)所示，经过对比可以发现规律如下：

$$\left.\begin{aligned} H_{ij} &= L_{ij} \\ N_{ij} &= -J_{ij} \end{aligned}\right\} \tag{4.1}$$

对角元素可先按式(2.22)进行计算，得到下式：

$$\left.\begin{aligned} H'_{ii} &= -U_i^2(G_{ii}\sin\delta_{ii} - B_{ii}\cos\delta_{ii}) = U_i^2 B_{ii} \\ N'_{ii} &= -U_i^2(G_{ii}\cos\delta_{ii} - B_{ii}\sin\delta_{ii}) = -U_i^2 G_{ii} \\ J'_{ii} &= U_i^2(G_{ii}\cos\delta_{ii} - B_{ii}\sin\delta_{ii}) = U_i^2 G_{ii} \\ L'_{ii} &= -U_i^2(G_{ii}\sin\delta_{ii} - B_{ii}\sin\delta_{ii}) = U_i^2 B_{ii} \end{aligned}\right\} \tag{4.2}$$

然后按照式(2.23)修正：

$$\left.\begin{aligned} H_{ii} &= H'_{ii} + Q_i \\ N_{ii} &= N'_{ii} - P_i \\ J_{ii} &= J'_{ii} - P_i \\ L_{ii} &= L'_{ii} - Q_i \end{aligned}\right\} \tag{4.3}$$

通过以上规律可以看出，在进行矩阵运算时，只需对其中两种元素 H_{ij}、N_{ij} 或 J_{ij}、L_{ij} 进行计算，然后根据其相互之间规律直接得出另两种元素。

通过式(2.22)可以看出，元素 H_{ij}、N_{ij} 的定义式都是由 U_i、U_j、$\cos\delta_{ij}$、$\sin\delta_{ij}$ G_{ij}、B_{ij} 进行基本运算得到的，根据这一特点可设定元素 A_{ij} 并定义如下：

$$A_{ij} = U_i U_j \angle\delta_{ij}(\mathrm{G}_{ij} - \mathrm{jB}_{ij}) \tag{4.4}$$

式中 δ_{ij} 指节点 i、j 之间的相角差。

将式(4.4)展开如下：

$$A_{ij} = U_i U_j[(G_{ij}\cos\delta_{ij} + B_{ij}\sin\delta_{ij}) + j(G_{ij}\sin\delta_{ij} - B_{ij}\cos\delta_{ij})] \tag{4.5}$$

通过比较式(4.5)和式(2.22)可以得到：

$$-A_{ij} = H_{ij} + \mathrm{j}N_{ij} \tag{4.6}$$

分析式(4.1)、(4.6)可得出 A_{ij} 的实部、虚部与雅可比矩阵各元素的关系，对于分块矩阵的非对角元素可得关系如下：

$$\left.\begin{aligned} H_{ij} &= -\mathrm{Im}(A_{ij}) \\ N_{ij} &= -\mathrm{Re}(A_{ij}) \\ J_{ij} &= \mathrm{Re}(A_{ij}) \\ L_{ij} &= -\mathrm{Im}(A_{ij}) \end{aligned}\right\}(i \neq j) \tag{4.7}$$

另外，对于分块矩阵的对角元素可进行如下修正：

$$\left.\begin{aligned} H_{ii} &= -\mathrm{Im}(A_{ii}) + Q_i \\ N_{ii} &= -\mathrm{Re}(A_{ii}) - P_i \\ J_{ii} &= \mathrm{Re}(A_{ii}) - P_i \\ L_{ii} &= -\mathrm{Im}(A_{ii}) - Q_i \end{aligned}\right\} \tag{4.8}$$

这样就形成了雅可比矩阵的四个子块，从而利用了矩阵的相关运算代替了循环过程。下面以 A_{ij} 作为矩阵元素组成矩阵如下：

$$\boldsymbol{A} = \begin{pmatrix} U_1U_1\angle\delta_{11}\overset{*}{Y}_{11} & U_1U_2\angle\delta_{12}\overset{*}{Y}_{12} & \cdots & U_1U_n\angle\delta_{1n}\overset{*}{Y}_{1n} \\ U_2U_1\angle\delta_{21}\overset{*}{Y}_{21} & U_2U_2\angle\delta_{22}\overset{*}{Y}_{22} & \cdots & U_2U_n\angle\delta_{2n}\overset{*}{Y}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ U_nU_1\angle\delta_{n1}\overset{*}{Y}_{n,1} & U_nU_2\angle\delta_{n2}\overset{*}{Y}_{n2} & \cdots & U_nU_n\angle\delta_{nn}\overset{*}{Y}_{nn} \end{pmatrix} \tag{4.9}$$

进一步将 A_{ij} 写成相量形式可得：

$$\boldsymbol{A} = \begin{pmatrix} \dot{U}_1\overset{*}{Y}_{11}\overset{*}{U}_1 & \dot{U}_1\overset{*}{Y}_{12}\overset{*}{U}_2 & \cdots & \dot{U}_1\overset{*}{Y}_{1n}\overset{*}{U}_n \\ \dot{U}_2\overset{*}{Y}_{21}\overset{*}{U}_1 & \dot{U}_2\overset{*}{Y}_{22}\overset{*}{U}_2 & \cdots & \dot{U}_2\overset{*}{Y}_{2n}\overset{*}{U}_n \\ \vdots & \vdots & \ddots & \vdots \\ \dot{U}_n\overset{*}{Y}_{n1}\overset{*}{U}_1 & \dot{U}_n\overset{*}{Y}_{n2}\overset{*}{U}_2 & \cdots & \dot{U}_n\overset{*}{Y}_{nn}\overset{*}{U}_n \end{pmatrix} \tag{4.10}$$

将式(4.10)所表示的矩阵 $\boldsymbol{A}$ 称为雅可比初始计算矩阵，这个矩阵与导纳矩阵同为 n 阶。通过引入雅可比初始计算矩阵矩阵 $\boldsymbol{A}$ 可直接得到雅可比矩阵中四个分块矩阵，然后进行组合形成 $2(n-1)$阶的雅可比矩阵。因此，在运用矩阵运算形成雅可比矩阵的过程中，矩阵 $\boldsymbol{A}$ 的形成是最为关键的步骤。

矩阵相乘是一种基本的矩阵运算，式(4.10)中所示的矩阵可以拆分成三个矩阵相乘的形式，具体过程如下：

$$\boldsymbol{A}=\begin{pmatrix}\dot{U}_1 & & & \\ & \dot{U}_2 & & \\ & & \ddots & \\ & & & \dot{U}_n\end{pmatrix}\begin{pmatrix}\overset{*}{Y}_{11} & \overset{*}{Y}_{12} & \cdots & \overset{*}{Y}_{1n} \\ \overset{*}{Y}_{21} & \overset{*}{Y}_{22} & \cdots & \overset{*}{Y}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \overset{*}{Y}_{n1} & \overset{*}{Y}_{n2} & \cdots & \overset{*}{Y}_{nn}\end{pmatrix}\begin{pmatrix}\overset{*}{U}_1 & & & \\ & \overset{*}{U}_2 & & \\ & & \ddots & \\ & & & \overset{*}{U}_n\end{pmatrix} \tag{4.11}$$

式(4.11)中表述的形成雅可比初始矩阵的方法即通过对电压列向量对角化后相乘的形式来形成。

在 Matlab 中，其编程语句为：“A=diag(U)*conj(Y)*diag(U')”，语句中使用了 diag 函数和 conj 函数。“conj(M)”表示对矩阵 $\boldsymbol{M}$ 中的每个元素取共轭；diag(M)”表示取矩阵 $\boldsymbol{M}$ 的对角元素，如果矩阵 $\boldsymbol{M}$ 是一个行向量或者列向量，则“diag(M)”则表示以 $\boldsymbol{M}$ 作为对角元素形成矩阵，其他元素为 0。

4.2.2 点乘运算

上述的电压相量相乘的方法体现了矩阵运算的思想，利用了 Matlab 矩阵运算的优势，编程简洁。计算雅可比初始计算矩阵 $\boldsymbol{A}$ 时采用的是三个同阶方阵相乘的形式，对于大型电力系统的潮流计算，由于节点较多，对应方阵的阶数也较高，因此，计算量较大。

进行矩阵运算的目的就是要得到式(4.10)中的矩阵元素，下面说明另一种形成上述矩阵的方法即在利用矩阵相乘的基础上引入点乘运算，为方便叙述，将式(4.10)调整如下：

$$\boldsymbol{A}=\begin{pmatrix}\dot{U}_1\overset{*}{U}_1\overset{*}{Y}_{11} & \dot{U}_1\overset{*}{U}_2\overset{*}{Y}_{12} & \cdots & \dot{U}_1\overset{*}{U}_n\overset{*}{Y}_{1n} \\ \dot{U}_2\overset{*}{U}_1\overset{*}{Y}_{21} & \dot{U}_2\overset{*}{U}_2\overset{*}{Y}_{22} & \cdots & \dot{U}_2\overset{*}{U}_n\overset{*}{Y}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \dot{U}_n\overset{*}{U}_1\overset{*}{Y}_{n1} & \dot{U}_n\overset{*}{U}_2\overset{*}{Y}_{n2} & \cdots & \dot{U}_n\overset{*}{U}_n\overset{*}{Y}_{nn}\end{pmatrix} \tag{4.12}$$

这样变化后，可将 $\dot{U}_i\overset{*}{U}_j$ 看作一个元素与 $\overset{*}{Y}_{ij}$ 相乘，通过两个矩阵的元素相乘构成新矩阵的元素，使用的运算方法是矩阵运算中的点乘运算。点乘运算属于矩阵点运算的一种，与矩阵的相乘相比，计算量更小，点运算符一般有.*、./、.\和.^等几种。矩阵的点运算与矩阵的基本算术运算有所不同，点运算是将两矩阵的对应元素进行相关运算，因此，要求两个矩阵维数必须相等。

矩阵的点乘除了用“.*”符号进行表示也可运用相关矩阵函数表示，Matlab 提供 dot 函数计算矩阵的点乘，其具体用法表示为：***Y***=dot(***M***,***N***)，计算同维矩阵 ***M*** 和 ***N*** 的点乘积，当于把 ***M***、***N*** 两个矩阵对应元素相乘后赋值到 ***L*** 矩阵对应元素，编程语句为为“L(i,j)=M(i,j)*N(i,j)”。

在 Matlab 的代数运算中，转置可分为共轭转置和非共轭转置两大类，对于实数矩阵 ***M*** 而言，共轭转置与非共轭转置的效果没有区别，单纯的转置运算可用 transpose(M)实现，不论是实矩阵还是复矩阵，都只实现转置而不作共轭变换。若想在实现矩阵 ***Z*** 转置的同时又实现共轭，则需使用 ***Z'*** 来实现。本节中，若想利用点乘法形成雅可比初始计算矩阵，需要构造一个以 $\dot{U}_i\overset{*}{\dot{U}}_j$ 为矩阵元素且与导纳矩阵同阶的矩阵 ***B***，可通过矩阵运算形成

$$\boldsymbol{B}=(\boldsymbol{U}\boldsymbol{U}') \tag{4.13}$$

U' 就表示对矩阵 ***U*** 转置的同时对矩阵元素取共轭，计算后可得矩阵 ***B*** 为

$$\boldsymbol{B}=\begin{pmatrix} \dot{U}_1\overset{*}{\dot{U}}_1 & \dot{U}_1\overset{*}{\dot{U}}_2 & \cdots & \dot{U}_1\overset{*}{\dot{U}}_n \\ \dot{U}_2\overset{*}{\dot{U}}_1 & \dot{U}_2\overset{*}{\dot{U}}_2 & \cdots & \dot{U}_2\overset{*}{\dot{U}}_n \\ \vdots & \vdots & \ddots & \vdots \\ \dot{U}_n\overset{*}{\dot{U}}_1 & \dot{U}_n\overset{*}{\dot{U}}_2 & \cdots & \dot{U}_n\overset{*}{\dot{U}}_n \end{pmatrix} \tag{4.14}$$

完成了上述推导过程后，便得到了另一种矩阵运算形成雅可比初始计算矩阵的方法即使用矩阵相乘构造相关矩阵后运用点乘运算来求取矩阵。矩阵形式表达如下：

$$\boldsymbol{A}=\left(\begin{pmatrix}\dot{U}_1\\ \dot{U}_2\\ \vdots\\ \dot{U}_n\end{pmatrix}\begin{pmatrix}\overset{*}{\dot{U}}_1 & \overset{*}{\dot{U}}_2 & \cdots & \overset{*}{\dot{U}}_n\end{pmatrix}\right).*\begin{pmatrix} \overset{*}{Y}_{11} & \overset{*}{Y}_{12} & \cdots & \overset{*}{Y}_{1n} \\ \overset{*}{Y}_{21} & \overset{*}{Y}_{22} & \cdots & \overset{*}{Y}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \overset{*}{Y}_{n1} & \overset{*}{Y}_{n2} & \cdots & \overset{*}{Y}_{nn} \end{pmatrix} \tag{4.15}$$

式(4.15)可用语句“U*U'.*conj(Y)”表示，可直接得到雅可比初始计算矩阵。

4.2.3 重复矩阵函数

式(4.11)中通过三个同阶矩阵相乘直接得到雅可比初始计算矩阵 ***A***，为方便进行算法设计可将其过程进行分解，首先通过前两个矩阵相乘可以得到结果如下：

$$\boldsymbol{A}_1=\begin{pmatrix}\dot{U}_1\overset{*}{Y}_{11} & \dot{U}_1\overset{*}{Y}_{12} & \cdots & \dot{U}_1\overset{*}{Y}_{1n}\\ \dot{U}_2\overset{*}{Y}_{21} & \dot{U}_2\overset{*}{Y}_{22} & \cdots & \dot{U}_2\overset{*}{Y}_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ \dot{U}_n\overset{*}{Y}_{n1} & \dot{U}_n\overset{*}{Y}_{n2} & \cdots & \dot{U}_n\overset{*}{Y}_{nn}\end{pmatrix} \tag{4.16}$$

通过对 $\boldsymbol{A}_1$ 分析可以看出，$\boldsymbol{A}_1$ 的形成除了使用式(4.11)中所介绍的方法即通过对角矩阵与共轭导纳矩阵相乘外，还可以通过 repmat 函数与点乘运算结合来实现。

repmat 函数是 replicate matrix 的缩写，主要作用是复制和平铺矩阵，实现语句通常为“M=repmat(N,m,n)”，其含义是将矩阵 $\boldsymbol{N}$ 复制 $m\times n$ 个分块矩阵，以矩阵 $\boldsymbol{N}$ 作为矩阵 $\boldsymbol{M}$ 的矩阵元素，$\boldsymbol{M}$ 是由 $m\times n$ 个 $\boldsymbol{N}$ 平铺而成。首先设要复制的矩阵为 $\boldsymbol{U}$

$$\boldsymbol{U}=\begin{pmatrix}\dot{U}_1\\ \dot{U}_2\\ \vdots\\ \dot{U}_n\end{pmatrix} \tag{4.17}$$

根据上文中介绍的 repmat 函数，可使用语句“U_R=repmat(U,1,n–1)”对其进行操作，表示将矩阵 $\boldsymbol{U}$ 为元素进行复制形成一个新矩阵，可得结果如下：

$$\boldsymbol{U}_R=\begin{pmatrix}\dot{U}_1 & \dot{U}_1 & \cdots & \dot{U}_1\\ \dot{U}_2 & \dot{U}_2 & \cdots & \dot{U}_2\\ \vdots & \vdots & \ddots & \vdots\\ \dot{U}_n & \dot{U}_n & \cdots & \dot{U}_n\end{pmatrix} \tag{4.18}$$

通过上一小节中介绍的点乘运算并结合式(4.16)和式(4.18)，可以得到结果如下：

$$\boldsymbol{A}_1=\begin{pmatrix}\dot{U}_1 & \dot{U}_1 & \cdots & \dot{U}_1\\ \dot{U}_2 & \dot{U}_2 & \cdots & \dot{U}_2\\ \vdots & \vdots & \ddots & \vdots\\ \dot{U}_n & \dot{U}_n & \cdots & \dot{U}_n\end{pmatrix}.*\begin{pmatrix}\overset{*}{Y}_{11} & \overset{*}{Y}_{12} & \cdots & \overset{*}{Y}_{1n}\\ \overset{*}{Y}_{21} & \overset{*}{Y}_{22} & \cdots & \overset{*}{Y}_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ \overset{*}{Y}_{n1} & \overset{*}{Y}_{n2} & \cdots & \overset{*}{Y}_{mn}\end{pmatrix} \tag{4.19}$$

这样就可以得到另一种形成 $\boldsymbol{A}_1$ 的方法，即通过矩阵点乘运算和矩阵重复来形成矩阵 $\boldsymbol{A}_1$，进而得出一种形成雅可比初始计算矩阵的新方法：

$$\boldsymbol{A}=\begin{pmatrix}\dot{U}_1 & \dot{U}_1 & \cdots & \dot{U}_1\\ \dot{U}_2 & \dot{U}_2 & \cdots & \dot{U}_2\\ \vdots & \vdots & \ddots & \vdots\\ \dot{U}_n & \dot{U}_n & \cdots & \dot{U}_n\end{pmatrix}.*\begin{pmatrix}\overset{*}{Y}_{11} & \overset{*}{Y}_{12} & \cdots & \overset{*}{Y}_{1n}\\ \overset{*}{Y}_{21} & \overset{*}{Y}_{22} & \cdots & \overset{*}{Y}_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ \overset{*}{Y}_{n1} & \overset{*}{Y}_{n2} & \cdots & \overset{*}{Y}_{nn}\end{pmatrix}\begin{pmatrix}\overset{*}{U}_1 & & & \\ & \overset{*}{U}_2 & & \\ & & \ddots & \\ & & & \overset{*}{U}_n\end{pmatrix} \tag{4.20}$$

分析式(4.20)和式(4.11)可得，矩阵重复函数与点乘运算结合可以将对角矩阵相乘替代，首先可对矩阵 $\boldsymbol{U}$ 进行转置并取共轭得到矩阵 $\boldsymbol{U}'$，然后运用语句“U_{RT}=repmat(U',1,n-1)”对其进行复制可得矩阵如下：

$$\boldsymbol{U}_{RT}=\begin{pmatrix}\overset{*}{U}_1 & \overset{*}{U}_n & \cdots & \overset{*}{U}_n\\ \overset{*}{U}_1 & \overset{*}{U}_n & \cdots & \overset{*}{U}_n\\ \vdots & \vdots & \ddots & \vdots\\ \overset{*}{U}_1 & \overset{*}{U}_2 & \cdots & \overset{*}{U}_n\end{pmatrix} \tag{4.21}$$

根据式(4.21)可以得到另一种形成雅可比初始计算矩阵的方法，如下所示：

$$\boldsymbol{A}=\begin{pmatrix}\dot{U}_1 & & & \\ & \dot{U}_2 & & \\ & & \ddots & \\ & & & \dot{U}_n\end{pmatrix}\begin{pmatrix}\overset{*}{Y}_{11} & \overset{*}{Y}_{12} & \cdots & \overset{*}{Y}_{1n}\\ \overset{*}{Y}_{21} & \overset{*}{Y}_{22} & \cdots & \overset{*}{Y}_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ \overset{*}{Y}_{n1} & \overset{*}{Y}_{n2} & \cdots & \overset{*}{Y}_{nn}\end{pmatrix}.*\begin{pmatrix}\overset{*}{U}_1 & \overset{*}{U}_2 & \cdots & \overset{*}{U}_n\\ \overset{*}{U}_1 & \overset{*}{U}_2 & \cdots & \overset{*}{U}_n\\ \vdots & \vdots & \ddots & \vdots\\ \overset{*}{U}_1 & \overset{*}{U}_2 & \cdots & \overset{*}{U}_n\end{pmatrix} \tag{4.22}$$

通过 4.2.2 节的分析可得，点乘运算比矩阵相乘计算量更小，因此使用点乘运算与重复矩阵的方法完全替换矩阵相乘，可得下式：

$$\boldsymbol{A}=\begin{pmatrix}\dot{U}_1 & \dot{U}_1 & \cdots & \dot{U}_1\\ \dot{U}_2 & \dot{U}_2 & \cdots & \dot{U}_2\\ \vdots & \vdots & \ddots & \vdots\\ \dot{U}_n & \dot{U}_n & \cdots & \dot{U}_n\end{pmatrix}.*\begin{pmatrix}\overset{*}{Y}_{11} & \overset{*}{Y}_{12} & \cdots & \overset{*}{Y}_{1n}\\ \overset{*}{Y}_{21} & \overset{*}{Y}_{22} & \cdots & \overset{*}{Y}_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ \overset{*}{Y}_{n1} & \overset{*}{Y}_{n2} & \cdots & \overset{*}{Y}_{nn}\end{pmatrix}.*\begin{pmatrix}\overset{*}{U}_1 & \overset{*}{U}_2 & \cdots & \overset{*}{U}_n\\ \overset{*}{U}_1 & \overset{*}{U}_2 & \cdots & \overset{*}{U}_n\\ \vdots & \vdots & \ddots & \vdots\\ \overset{*}{U}_1 & \overset{*}{U}_2 & \cdots & \overset{*}{U}_n\end{pmatrix} \tag{4.23}$$

通过上述方法生成的雅可比初始计算矩阵是一个 $2n$ 阶的阵，在矩阵生成的过程中，一些元素本应为 0 的行、列也被赋值。因此，要去掉这些被赋值的行、列，具体方法与 3.3.3 节中介绍的方法一致。

4. 2. 4 计算功率的形成过程

通过式(4.8)可以看出，雅可比矩阵对角元素的修正需要利用节点功率，3.2.2

节中推导的计算功率形成需要在每次循环时进行计算，执行循环时效率比较低。本节将在矩阵运算的基础上设计两种方法，一种是通过复功率定义式直接矩阵运算得到，简称为直接计算法；另一种是通过雅可比初始计算矩阵 $\boldsymbol{A}$ 中的矩阵元素求和实现，简称为矩阵元素求和法。

(1) 直接计算法

复功率的直接定义式如式(2.6)所示，将其写成矩阵形式可得：

$$\begin{pmatrix}\tilde{S}_1\\ \tilde{S}_2\\ \vdots\\ \tilde{S}_n\end{pmatrix}=\begin{pmatrix}\dot{U}_1\sum_{j=1}^{j=n}\overset{*}{Y}_{1j}\overset{*}{U}_j\\ \dot{U}_2\sum_{j=1}^{j=n}\overset{*}{Y}_{2j}\overset{*}{U}_j\\ \vdots\\ \dot{U}_n\sum_{j=1}^{j=n}\overset{*}{Y}_{nj}\overset{*}{U}_j\end{pmatrix} \tag{4.24}$$

式(4.24)的形成过程如下：

$$\begin{pmatrix}\tilde{S}_1\\ \tilde{S}_2\\ \vdots\\ \tilde{S}_n\end{pmatrix}=\begin{pmatrix}\dot{U}_1\\ \dot{U}_2\\ \vdots\\ \dot{U}_n\end{pmatrix}.*\begin{pmatrix}\overset{*}{Y}_{11} & \overset{*}{Y}_{12} & \cdots & \overset{*}{Y}_{1n}\\ \overset{*}{Y}_{21} & \overset{*}{Y}_{22} & \cdots & \overset{*}{Y}_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ \overset{*}{Y}_{n1} & \overset{*}{Y}_{n2} & \cdots & \overset{*}{Y}_{nn}\end{pmatrix}\begin{pmatrix}\overset{*}{U}_1\\ \overset{*}{U}_2\\ \vdots\\ \overset{*}{U}_n\end{pmatrix} \tag{4.25}$$

(2) 矩阵元素求和法

首先对 Matlab 中的 sum 函数进行一些简单说明。设 $\boldsymbol{M}$、$\boldsymbol{N}$ 两个矩阵，主要应用形式如下：

“M=sum(N)”表示对矩阵 $\boldsymbol{N}$ 每列进行求和；

“M=sum(N,1)”表示对矩阵 $\boldsymbol{N}$ 每列进行求和；

“M=sum(N,2)”表示对矩阵 $\boldsymbol{N}$ 每行进行求和。

通过雅可比初始计算矩阵 $\boldsymbol{A}$ 中的元素得到各节点复功率的值，通过比较式(4.12)和式(4.24)可以发现，各节点复功率可通过对矩阵 $\boldsymbol{A}$ 中各行元素求和得到。在 Matlab 中可用语句“S=sum(A,2)”来实现。这个语句的具体意义是将矩阵 $\boldsymbol{A}$ 中各行的元素相加形成一个复数列向量，其中元素为式(3.4)所示，对元素取实部、虚部可分别得到节点有功功率 P_i、节点无功功率 Q_i，无需再单独计算。

4.3 预定义维数

Matlab 是以块(Block)的形式对矩阵进行存储的，计算机在运行过程中内存会被分割为许多不连续的存储区域，这样就导致了存储空间的不连续。在 3.1、3.2 节中设计的算法都是循环实现的矩阵，在循环过程中，矩阵的大小会随循环次数增加而变大，这样会造成矩阵需要的存储空间大于刚开始分配的存储空间。又由于存储空间的不连续，只能寻找更大的块来存储矩阵，这一过程叫做内存的动态分配。

动态分配过程会占有大量时间，也会使计算机内存不能充分利用。所以要在循环之前根据矩阵变量的大小进行预定义存储空间，这样 Matlab 能够在一直在一个合适的块中进行存储，减少了程序运行时间。数值矩阵通常利用 zeros 或 ones、函数，而字符矩阵一般用 cell 函数进行预定义存储空间。

本文所涉及的矩阵都为数值矩阵，故可将导纳矩阵、雅可比矩阵、不平衡量以及修正量组成的一维矩阵分别进行预定义存储空间，也就是根据矩阵的大小对矩阵的维数进行提前定义，从而提高程序的运行速度。

4.4 算法实现

4.2 节介绍了运用矩阵运算形成雅可比矩阵的方法，根据形成雅可比初始计算矩阵的思路不同，依次引入了矩阵相乘运算、点乘运算、重复矩阵函数 3 种方式对矩阵进行处理，形成了 5 种不同的方法。本节将根据 4.2 节的公式推导对 5 种方法进行算法的设计与实现。但 5 种方法只是在雅可比初始计算矩阵的形成上存在差别，在算法的其它步骤上并无差异。

对于矩阵运算形成雅可比矩阵的算法法设计步骤如下：

(1) 按照式(4.11)、(4.15)、(4.20)、(4.22)、(4.23)描述的方法分别计算雅可比初始计算矩阵；

(2) 运用直接计算法或矩阵元素求和法计算节点复功率，分别取其实部、虚部；

(3) 根据式(4.7)计算四个初始的雅可比矩阵子块矩阵 $\boldsymbol{H}^0$、$\boldsymbol{N}^0$、$\boldsymbol{J}^0$、$\boldsymbol{L}^0$；

(4) 设定 $k=1$；

(5) 根据式(4.8)和步各节点复功率的值，对雅可比矩阵分块子矩阵进行修正；

(6) 判断 $k>$ n 是否成立，若成立则 $k=k+1$ 并返回至步骤(5)；

(7) 由雅可比矩阵子块组合成雅可比矩阵；

(8) 根据图 3.3 中的流程图所示的步骤形成储存非零行标号的数组。

根据上述步骤可设计流程图如图 4.1 所示，通过流程图可以看出，在雅可比矩阵的形成过程中，通过矩阵运算对其子矩阵分别计算并重新组合。除对角元修正过程外，所有步骤均是通过矩阵运算完成，不再涉及循环过程。

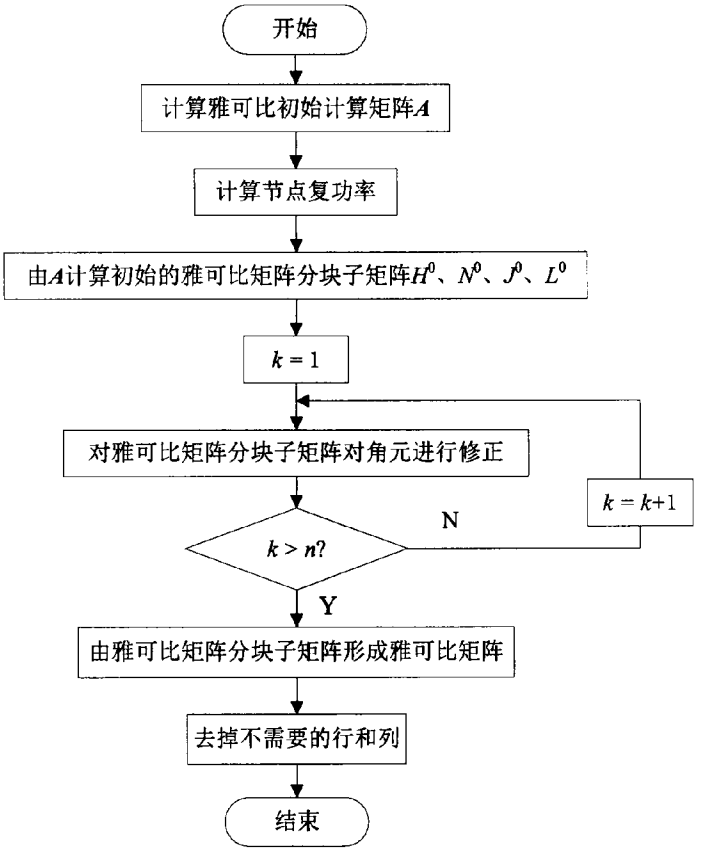

图 4.1 雅可比矩阵形成流程图

Fig.4.1 flow chart of the Jacobian matrix formation

4.5 算例分析

在完成上述算法设计后，对算例进行计算，并以量化的方式对算法的效果进行说明。在 3.5 节中也对算例进行了计算，为方便与前面算法效果进行对比，本节也采用东北电网 445 节点系统进行计算，并对计算所用的时间进行比较，评估算法的效果。为便于算法的计算时间比较，首先根据计算功率求取时采用的方法不同将潮流算法分为两大类：采用直接计算法求取计算功率和采用元素求和法求取

计算功率。

4.5.1 元素求和法求取计算功率

本章中只是针对雅可比矩阵的形成过程进行了设计，并没有涉及线性方程求解的过程，因此，需要借助第 3 章中的线性方程求解方法，本节将采用 Matlab 的矩阵除法分别与本章设计的形成雅可比矩阵的方法结合进行算例分析。为了与第 3 章方法进行区别，根据雅可比矩阵形成方法的不同依次编号如下：

方法 4：使用式(4.11)表示的形成雅可比矩阵的方法；

方法 5：使用式(4.15)表示的形成雅可比矩阵的方法；

方法 6：使用式(4.20)表示的形成雅可比矩阵的方法；

方法 7：使用式(4.22)表示的形成雅可比矩阵的方法；

方法 8：使用式(4.23)表示的形成雅可比矩阵的方法。

具体的计算时间如表 4.1 所示：

表 4.1 不同雅可比矩阵形成方法的计算时间

Tab.4.1 Computing time required by different Jacobian matrix formation method

雅可比矩阵计算方法	潮流计算时间(s)
方法 4	1.121
方法 5	0.559
方法 6	0.865
方法 7	0.856
方法 8	0.612

通过表 4.1 中数据可得结论如下：

(1) 通过表 3.1 与表 4.1 的比较可得，在不使用稀疏矩阵函数的情况下，除方法 4 略慢于方法 3 外，方法 5、6、7、8 均比方法 3 快，说明了在 Matlab 中，矩阵运算是解决潮流计算问题的更好选择。

(2) 通过表 4.1 可以看出方法 4 是以上算法中最慢的算法，原因是方法 4 进行了两次矩阵相乘，而方法 5、6、7 中矩阵相乘次数均为一次，方法 8 没有进行矩阵直接相乘，四种方法都是通过构造矩阵使用点乘运算来代替矩阵相乘，这说明了在不使用稀疏矩阵函数的情况下，点乘法比矩阵直接相乘的计算量小很多。

(3) 通过表 4.1 可以看出方法 5 是以上算法中最快的，方法 5 通过列向量 $\boldsymbol{U}$ 与

其转置矩阵取共轭的矩阵 $\boldsymbol{U}'$ 相乘构造了与导纳矩阵同阶的矩阵 $\boldsymbol{B}$，与方法 5、方法 6、方法 7 中 n 阶矩阵相乘的方法相比，计算量大大减少，因此速度得到提高。

由于潮流计算中涉及到很多稀疏矩阵，可利用 3.4 节中介绍的稀疏矩阵函数对表 4.1 中算法进行优化。通过稀疏矩阵函数对各算法中涉及的导纳矩阵、雅可比矩阵以及式(4.11)、式(4.20)、式(4.22)所使用的对角化后的电压矩阵进行存储，并对东北电网 445 节点系统进行计算。计算时间如表 4.2 所示。

表 4.2 不同潮流算法使用稀疏矩阵函数后的计算时间

Tab.4.2 Computing time required by different Jacobian matrix formation method using sparse function

雅可比矩阵计算方法	潮流计算时间(s)
方法 4	0.230
方法 5	0.265
方法 6	0.288
方法 7	0.252
方法 8	0.310

通过表 4.2 中数据可得结论如下：

(1) 通过表 4.2 与表 4.1 比较，可以看出对各矩阵使用稀疏矩阵函数存储后，表中方法计算速度均有了提高，因此，对于潮流算法优化而言，稀疏矩阵函数的使用很有必要。

(2) 方法 5 中使用的行向量和列向量不是稀疏矩阵，方法 8 中两个电压矩阵均采用了 repmat 函数对矩阵进行复制，新组成的矩阵也不再是稀疏矩阵，这两种算法只有导纳矩阵、雅可比矩阵可使用稀疏矩阵存储，因此速度提高量较小；方法 6、方法 7 均有一个矩阵采用了 repmat 函数对其进行复制，新组成的矩阵不再是稀疏矩阵，所涉及的稀疏矩阵除了导纳矩阵、雅可比矩阵外，还有一个电压对角矩阵，因此速度提高量适中；而方法 4 所涉及的导纳矩阵、雅可比矩、两个电压对角矩阵均具有稀疏性，因此速度提高量最大，有 0.9s 之多，成为了速度最快的算法。

4.5.2 直接计算法求取计算功率

4.5.1 节算法涉及到的复功率计算均采用的是雅可比初始计算矩阵元素按行求和的方法，为了对 4.2.4 节中介绍的两种计算功率计算方法进行比较，下面用直接

计算法来替代元素求和法对相同算例进行计算，并比较计算时间。

在不使用稀疏矩阵函数的情况下，计算时间如表 4.3 所示。

表 4.3 直接计算法计算复功率的潮流计算时间比较

Tab.4.3 Computing time required by different power flow algorithm using complex power direct calculation method

雅可比矩阵计算方法	潮流计算时间(s)
方法 4	1.157
方法 5	0.606
方法 6	0.901
方法 7	0.896
方法 8	0.657

通过表 4.3 中数据可得结论如下：

(1) 整体比较表 4.1 和表 4.3 可得，直接计算法求取计算功率比元素求和法求取计算功率速度上略慢；

(2) 功率计算是相对其他过程独立存在的，表 4.3 中各方法均比表 4.1 中各方法慢 0.04s 左右，并没有影响到各算法间相互之间速度快慢的排列顺序。

对表 4.3 中的各算法使用稀疏矩阵函数后，可以得到计算时间如表 4.4 所示。

表 4.4 不同潮流算法使用稀疏矩阵函数后的计算时间

Tab.4.4 Computing time required by different Jacobian matrix formation method using sparse function

雅可比矩阵计算方法	潮流计算时间(s)
方法 4	0.178
方法 5	0.217
方法 6	0.240
方法 7	0.203
方法 8	0.263

(1) 比较表 4.4 与表 4.3 可得，在使用稀疏矩阵函数后，五种方法在速度上均有了明显提高；

(2) 比较表 4.2 与表 4.4 可得，在使用稀疏矩阵函数后，使用直接计算法求取

功率时比元素求和法均有了提高。在式(4.25)中，直接计算法存在矩阵 $\dot{\boldsymbol{Y}}$，对此矩阵可用稀疏矩阵方法进行存储，因而在计算速度上有了提高。

4.6 小结

本章主要进行了雅可比矩阵形成过程的设计，通过矩阵运算的方式形成雅可比矩阵，并对东北电网 445 节点系统进行计算，通过计算时间的比较，验证了算法设计的有效性。雅可比初始计算矩阵的形成是最为关键的环节，根据形成雅可比初始计算矩阵的思路不同，依次引入了矩阵相乘运算、点乘运算、重复矩阵函数 3 种方式对矩阵进行处理，形成了 5 种不同的方法。对计算功率的形成设计了两种算法，分别是直接计算法和矩阵元素求和法。此外，还利用稀疏矩阵函数对潮流算法进行优化。为验证 5 种方法及其改进的效果，在本章的 4.5 节进行了算例计算。将这 5 种方法分别与 3.3.3 介绍的 Matlab 的矩阵除法相结合组成潮流算法进行算例计算，并根据计算时间得出相关结论。

结论与展望

潮流计算作为电力系统分析计算的主要工具用途十分广泛，根据其应用领域不同可分为工程型潮流算法和科研型潮流算法。本文主要对科研型潮流算法进行研究，运用循环方式和矩阵运算方式进行了算法的设计，并在 Matlab 的运行环境下进行算法实现。

通过本文的研究可总结如下：

(1) 运用循环方式的潮流算法是一种经典的方法，其思路清晰，便于编程者理解，算法的可移植性好。本文首先设计了一种循环形成雅可比矩阵的方法，然后从研究修正方程的系数矩阵的元素排列特点和 Matlab 矩阵存储规律出发，对线性方程求解过程进行了算法设计，并根据潮流计算中稀疏矩阵的特点使用稀疏矩阵函数进行算法优化。

(2) 矩阵运算形成雅可比矩阵的原理是通过雅可比初始计算矩阵来形成雅可比矩阵中的四个矩阵子块，进而组合成完整的雅可比矩阵。本文根据形成雅可比初始计算矩阵的思路不同，依次引入了矩阵相乘运算、点乘运算、重复矩阵函数 3 种方式对矩阵进行处理，形成了 5 种不同的方法。在进行对角元素修正时会用到计算功率，对计算功率的形成设计了两种算法，分别是直接计算法和矩阵元素求和法。矩阵运算则更注重于公式的推导，可通过公式的变换来设计不同的形成方法，而且在 Matlab 的运行环境下更具备算法改进的潜力，Matlab 中矩阵函数的应用使得算法实现更加方便，算法设计更加多样，也使得程序运行速度更快。

本文所设计的算法具备较好的实用价值，但由于时间和水平有限，本文的设计仅局限于极坐标牛顿法这一种方法，而潮流算法种类较多，其他算法的相关研究也应继续展开，从而使科研型潮流算法的内容更为丰富，可选算法更多。

参考文献

[1] 于永源, 杨绮雯. 电力系统分析(第 3 版) [M]. 北京: 中国电力出版社, 2007.

[2] Chiang H. System modeling and stability problems[M]. John Wiley & Sons, Inc. 2010:14-28.

[3] 陈珩, 万秋兰, 陈怡, 等. 电力系统稳态分析(第 3 版) [M]. 北京: 中国电力出版社, 2007.

[4] 王锡凡, 方万良, 杜正春. 现代电力系统分析[M]. 北京: 科学出版社, 2003.

[5] Gusev S, Oboskalov V. Recursion based contingency analysis of an electrical power system[C]. International Symposium on Industrial Electronics (INDEL), 2016, 10(12): 1-4

[6] Wang R, Zhang H, Li C, et al. Dynamic power flow calculation method of power system with wind power[C]. 5th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies (DRPT), Changsha, 2015, 1987-1991.

[7] 孔德明. 电力系统动态潮流算法与程序开发: (博士学位论文). 济南: 山东大学, 2012.

[8] Ren Z, Wang K, Li W Y, et al. Probabilistic power flow analysis of power systems incorporating tidal current generation[J]. IEEE Transactions on Sustainable Energy, 2016, 99(2): 21-23.

[9] 安然然, 张海波, 郭琦, 等. 基于 RTDS 仿真模型的能量管理系统潮流数据在线仿真系统开发[J]. 电网技术, 2012, 36 (3): 277-282.

[10] Kim S J, Giannakis G B, Lee K Y. Online optimal power flow with renewables[C]. Asilomar Conference on Signals, Systems and Computers. IEEE, 2015: 355-360.

[11] 孙秋野, 陈会敏, 杨家农, 等. 牛顿类潮流计算方法的收敛性分析[J]. 中国电机工程学报, 2014, 34(13): 2196-2200.

[12] 刘启蒙, 杨鉴, 戈文江. 电网节点编号优化算法的改进[J]. 河北电力技术, 2010, 29(1): 33-35.

[13] 李保卫, 李佩杰, 韦化. 用于改进潮流计算中 PV-PQ 节点类型转换逻辑的非线性规划模型[J]. 电网技术, 2009, 33(03): 29-32,3.

[14] 李佩杰, 韦化, 白晓清. 小干扰稳定约束最优潮流的非线性半定规划方法[J]. 中国电机工程学报, 2013, 33(7): 69-76.

[15] Yamashiro S. Optimization of power flow by DC method[J]. Electrical Engineering in Japan, 1977, 97(6): 45-51.

[16] Stott B, Alsac O. Fast decoupled load flow[J]. IEEE Transactions on Power Apparatus and System, 1974, 93(3): 859-869.

[17] 黄升. 含小阻抗支路系统的快速分解法潮流算法分析与改进: (硕士学位论文). 大连: 大连海事大学, 2014.

[18] 姚玉斌, 刘莉, 陈学允. 基于快速分解法的连续潮流法[J]. 哈尔滨工业大学学报, 2000, 32(2): 128-131.

[19] 岩本伸一, 田村康男. 保留非线性的快速潮流计算法[J]. 岩本伸一, 田村康男来华讲学论文汇编. 武汉: 武汉水电学院, 1982.

[20] Meliopoulos A P S, Cokkinides G J, Chao X Y. A new probabilistic power flow analysis method[J]. IEEE Transactions on Power Systems, 1990, 5(1): 182-190.

[21] Stefopoulos G K, Meliopoulos A P, Cokkinides G J. Probabilistic power flow with non-conforming electric loads[J]. International Journal of Electrical Power & Energy Systems, 2005, 27(9–10): 627-634.

[22] Tinney W F, Hart C E. Power flow solution by newton's method[J]. IEEE Transactions on Power Apparatus & Systems, 2007, 86(11): 1449-1460.

[23] Lin W M, Teng J H. Three-phase distribution network fast-decoupled power flow solutions[J]. International Journal of Electrical Power & Energy Systems, 2000, 22(5): 375-380.

[24] 夏沛, 汪芳宗. 大规模电力系统快速潮流计算方法研究[J]. 电力系统保护与控制, 2012, 40(9): 38-42.

[25] Matos M A, Gouveia E. The fuzzy power flow revisited[C]. 2005 IEEE Russia Power Tech. 2008, 1-7.

[26] Ippolito L, Loia V, Siano P. Extended fuzzy c-means and genetic algorithms to optimize power flow management in hybrid electric vehicles[J]. Fuzzy Optimization and Decision Making, 2003, 2(4): 359-374.

[27] 韩富春, 王英, 张丽, 等. 人工神经网络在电力系统网损计算中的应用[J]. 太原理工大学学报, 2004, 35(6): 664-666.

[28] 罗军, 于歆杰. 基于遗传算法的稀疏节点优化编号方法[J]. 电网技术, 2006, 30(22): 54-58.

[29] Kumari M S, Maheswarapu S. Enhanced genetic algorithm based computation technique for multi-objective Optimal Power Flow solution[J]. International Journal of Electrical Power & Energy Systems, 2010, 32(6):736-742.

[30] Solomonesc F, Teslovan R, Barbulescu C, et al. Genetic algorithm power flow computing approach[C]. 47th International Universities Power Engineering Conference (UPEC), London, 2012: 1-6.

[31] 王晓昆. 电力系统潮流的同伦算法(硕士学位论文). 保定: 华北电力大学(河北), 2008.

[32] 王守相, 王成山. 配电系统节点优化编号方案比较[J]. 电力系统自动化, 2003, 27(8): 54-58.

[33] 颜伟, 黄正波, 余娟. 潮流计算中的二层链表与有序节点关联信息生成法[J]. 电网技术, 2010, 34(11): 87-92.

[34] 王文举, 李光耀. 配电网的回路阻抗法潮流计算与仿真可视化[J]. 同济大学学报(自然科学版), 2012, 40(3): 459-467.

[35] 何志军. 改进十字链表的存储方法在短路电流计算中的应用: (硕士学位论文). 长沙: 湖南大学, 2016

[36] Zhou G, Bo R, Chien L. GPU-Based batch LU-factorization solver for concurrent analysis of massive power flows[J]. IEEE Transactions on Power Systems, 2017, 32(6):23-25.

[37] Jiang H, Chen D, Li Y. A fine-grained parallel power flow method for large scale grid based on lightweight gpu threads[C]. IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS), Wuhan, 2016, 785-790.

[38] 颜伟, 黄正波, 余娟, 等. 牛顿法潮流计算的高效综合稀疏技术[J]. 中国电力, 2010, 43(7): 19-23.

[39] 朱凌志, 安宁. 基于二维链表的稀疏矩阵在潮流计算中的应用[J]. 电网技术, 2005, 29(8): 51-55.

[40] 薛巍, 舒继武, 王心丰, 等. 电力系统潮流并行算法的研究进展[J]. 清华大学学报(自然科学版), 2002, 42(9): 1192-1195,1199.

[41] 姚玉斌, 刘东梅, 陈学允. 求解含有小阻抗支路系统潮流的一种新方法[J]. 哈尔滨工业大学学报, 2001, 33(4): 525-529.

[42] 谢开贵, 周家启. 树状网络潮流计算的新算法[J]. 中国电机工程学报,2001, 21(9): 117-121.

[43] 苏津, 阳育德, 覃智君. 基于矢量化运算模式的电力系统潮流计算[J]. 电网技术, 2008, 32(3): 88-92.

[44] 刘可真, 陈勇. 电力系统潮流计算的 Matlab 程序改进方法[J]. 科技创新导报, 2008(24): 45-46.

[45] Dolan M J, Davidson E M, Ault G W, et al. Distribution power flow management utilizing an online constraint programming method[J]. IEEE Transactions on Smart Grid, 2013, 4(2): 798-805.

[46] Agreira C I F, Ferreira C M M, Barbosa F P M. Electric power system's steady-state security analysis applying the rough set theory considering an incomplete information system[C]. 44th International Universities Power Engineering Conference (UPEC), Glasgow, 2009, 1-5.

[47] 李磊. 在线静态安全分析算法研究: (硕士学位论文). 天津: 天津大学, 2009.

[48] 杨伟, 滕百岸, 孙磊. 电力市场中最优潮流模型及算法研究[J]. 电网技术, 2012, 36(2): 126-130.

[49] 刘理想. BX 型快速分解法的小阻抗潮流算法研究: (硕士学位论文). 大连: 大连海事大学, 2015.

[50] 侯莉. 含小阻抗支路系统牛顿法潮流算法研究: (硕士学位论文). 大连: 大连海事大学, 2014.

[51] 顾本华. 电力系统潮流计算实用化研究: (硕士学位论文). 大连: 大连海事大学, 2012.

[52] 周建民, 岂兴明, 矫津毅. MATLAB 从入门到精通[M]. 北京: 人民邮电出版社, 2008.

[53] 郁忠梅, 李有安, 赵法起, 等. 基于 Matlab 的电力系统潮流计算[J]. 山东农业大学学报(自然科学版), 2010, 41(02): 291-294.

[54] 张德喜. MATLAB 实用教程[M]. 北京: 中国铁道出版社, 2016.

[55] 李维波. MATLAB 在电气工程中的应用[M]. 中国电力出版社, 2006.

[56] Granelli G, Montagna M. MATLAB implementation of the complex power flow[J]. COMPEL, 2013, 32(3): 923-935.

[57] Eickmann J, Kellermann J, Moser A. Efficient power flow approximation methodology for topology optimization algorithms in transmission system operation[C]. 2015 International Conference on Clean Electrical Power (ICCEP), Taormina, 2015, 42(1): 147-154.

[58] 李文博. 输配电网潮流与优化的理论研究: (博士学位论文). 山东大学, 2013.

[59] 顾洁, 陈章潮, 徐蓓. 一种新的配电网潮流算法—改进牛顿—拉夫逊法[J]. 华东电力, 2000, 28(5): 10-12.

[60] 杨跃光, 刘璇. 一种改进的 Newton 算法在电力系统潮流计算中的研究与应用[J]. 陕西电力, 2011, 21(10): 4-6,10.

攻读学位期间公开发表论文

姚玉斌, 曹井川, 王丹. 一种基于 Matlab 的极坐标牛顿法潮流计算方法. 中国, 发明专利, 201610863886.1,2016.9.29.

致 谢

时间如白驹过隙，转眼两年的研究生生活即将结束，回首两年求学之路，感慨万千，时值论文完成之际，表达我心中的感激之情。

首先，感谢的是我的导师姚玉斌教授，老师的严谨治学、严于律己的态度，老师的博学广知，都令我受益匪浅，老师不仅仅是教导我如何治学，如何研究问题，更教会了我该如何为人处世。在此，向姚玉斌老师表达深深的感激之情！

感谢实验室的赵伟同学、刘景旺同学、阳义青师兄，以及实验室的各位挚友，是你们创造了严肃而不失活力的实验室氛围，与你们在一起，总是能其乐融融。身处这种良好的氛围中，我才能专心科研，顺利完成本篇论文。

感谢一直以来支持我的亲朋好友，尤其是养育我多年的父母。漫漫求学路，一路有你们相随，是你们的鼓励和期盼让我一直不断成长，是你们的陪伴和温情让我能战胜困难，勇往直前！

感谢百忙之中评阅论文的各位老师！

鹰击长空，鱼翔浅底！我会带着老师的嘱托和各位的祝福继续前行！

作者简介

姓 名： 曹井川

性 别： 男

出生年月：1991 年 5 月

民 族： 汉族

籍 贯： 山东省济宁市

研究方向：电力系统分析

学习经历：2011.09—2015.07 青岛科技大学 本科 电气工程及其自动化

2015.09—2017.07 大连海事大学 硕士 电气工程