Infeasible Primal-Dual Interior Point Algorithms for Solving Optimal Power Flow Problems

by

Xihui Yan

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical Engineering

Waterloo, Ontario, Canada, 1997

0-612-21399-4

Abstract

Many applications in power system operations and planning need efficient optimization methods to solve large-scale problems within a short period of time. This requirement is even more pronounced for real-time controls where fast solution speed is most important. As a major on-line application, the OPF problem is concerned with using mathematical programming methods to determine a secure and economic operating condition of power systems. The main objective of this research is, therefore, to develop and systematically evaluate advanced interior point methods for the efficient and reliable OPF solutions.

In this thesis, the OPF problem is formulated as a constrained nonlinear program in terms of all control/state variables, considering both power balance equality and security inequality constraints. Two particular OPF cases are studied in detail, namely, the real and reactive power dispatch problems. The minimization of production cost is considered as the objective in real power dispatch problems; while for reactive power dispatch problems, the objective function is the transmission active power losses to be minimized during the optimization process.

Successive linear programming is used to deal with the nonlinearity of the underlying problems. Consequently, the nonlinear OPF problem is linearized as a sequence of linear sub-problems, which are in turn solved by using interior point methods. To better suit the application of interior point methods, the sparse linear formulations are derived for both real and reactive power dispatch problems, based on decouple and couple load flow models, respectively.

The study of interior point methods is concentrated on infeasible primal-dual path-following methods. The derivations of two variants in this class of methods are presented in detail, namely, the infeasible primal-dual and the predictor-corrector primal-dual algorithms. Both algorithms are extended for a more general linear programming problem, considering lower and upper bounds for special needs in our applications. The search directions produced by these algorithms are analyzed to better understand the characteristics of interior point methods under research.

To explore the full potential of interior point methods for power engineering problems, intensive study has focused on all issues that influence the algorithm performance, such as the adjustment of barrier parameter, the determination of Newton step length and the initial point, and the use of multiple corrector steps. Practical issues related to successive linearization procedure are also investigated, including the choice of the linear step size and the tolerances for linear programming as well as for OPF procedure. Their effects on OPF performance are evaluated.

As the results of these investigations, several heuristics are proposed to reduce the number of iterations and to save computational work in every iteration. Extensive numerical experiments have demonstrated that the OPF solution speed can be significantly improved by customizing algorithm parameters to the specific applications under concern. Finally, the use of sparse techniques is investigated in developing fast and robust interior point codes. Test results on large-scale problems have confirmed the efficiency and reliability of the algorithms.

Acknowledgements

I wish to express my sincere gratitude to Professor Victor H. Quintana for his supervision, constant encouragement, and continuous support during the course of this research.

I also wish to thank Professor Anthony Vanelli for generously allowing me to use the CPLEX software as well as computer resources.

Finally, I would like to thank the National Science and Engineering Research Council (NSERC) of Canada for the financial support.

To my mother and father in eternity

To my wife Xiaohui and our great son Ao

Glossary

The following abbreviations are frequently used in the text of the thesis:

AC	Alternating Current
DC	Direct Current
IPM	Interior Point Method
KKT	Karush-Kuhn-Tucker necessary conditions
LP	Linear Programming
NLP	NonLinear Programming
OPF	Optimal Power Flow
PDIPA	Primal-Dual Interior Point Algorithm
PC-PDIPA	Predictor-Corrector Primal-Dual Interior Point Algorithm
QP	Quadratic Programming
RPD	Reactive Power Dispatch
SCED	Security-Constrained Economic Dispatch
SLP	Successive Linear Programming

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The fundamental mission of a power system is to provide consumers with sustained, reliable and cost-efficient electrical energy. In order to achieve this goal, system operators need to constantly adjust various controls such as generation outputs, transformer tap ratios, etc., to assure the continuous economic and secure system operations. This is a difficult task that relies highly on the *optimal power flow* (OPF) function at power system control centers [15]. The OPF procedure consists of using mathematical methodology to find the optimal operation of a power system under feasibility and security constraints. It has been considered as a basic tool for determining secure and economic operating conditions of power systems.

The optimal power flow problem can be traced back as early as 1920's when economic allocation of generation was the only concern [47]. The economic operation of power systems was achieved by dividing loads among available generator units such that their incremental generation costs are equal. This was a rather simple problem where only operating limits on real power generations were considered and the effect of system losses was either neglected or approximated by penalty factors calculated from the loss formula or load flow Jacobian matrix [96].

As power systems became increasingly large and complex, the security became an important issue, which requires more detailed system models. On the other hand, the evolution of digital computers made such detailed modeling become possible. In 1962, Carpentier for the first time established the OPF problem on a rigorous mathematical base [14]. He formulated it as a constrained nonlinear programming problem and derived its optimality conditions using the Kuhn-Tucker theorem. In his formulation, the OPF problem is expressed in terms of all control and state variables, with both network and security constraints. The objective function can be total generation cost or transmission losses, depending on a specific application.

In the past three decades, various optimization techniques have been proposed to solve the OPF problems. They range from improved mathematical techniques to more efficient problem formulations [16, 82, 22]. According to different models in use, the OPF methods can be classified as non-compact methods where network sparsity is retained, or compact ones in which the state variables are expressed in terms of control variables using various sensitivities. Based on the applied mathematical optimization, the OPF methods can be categorized as Nonlinear Programming (NLP), Successive Linear Programming (SLP), and Non-conventional techniques. A brief review on the OPF developments is provided next.

The gradient methods, using only first-order information, were initially used for the solution of OPF problems [14, 24]. These methods are characterized by slow and unreliable convergence. Soon after, the quadratic programming (QP) approaches were proposed, which use the second-order derivatives to improve the convergence of the gradient methods. Their distinct feature is that they use the Quasi-Newton process to iteratively approximate the Hessian matrix and, thus, avoid the difficulty in explicitly calculating the second derivatives of the load flow equations [49, 77].

However, the reduced Hessian so created is dense, which may make these methods too slow as the number of control variables becomes very large.

As the demand for faster and more stable techniques grew, a more accurate representation of the second-order information became essential. Lagrangian techniques with the exact Hessian matrix regained engineers' interest. Although these methods were proposed as earlier as 1960's, few were either reliable and fast until Sun *et al.* [84] introduced a Newton approach combined with Lagrangian techniques and penalty functions. With efficient data structure and sparse techniques, Sun's algorithm became very attractive and successful at the time. The major difficulty in this algorithm development turned out to be the efficient identification of binding inequality constraints.

Recently, the linear programming (LP) techniques have been proposed to solve the OPF problem [4, 82]. These methods are based on the linearization of OPF constraints and the objective function. An incremental model is created and a proper LP method is applied. As the linear model gives satisfactory results only in a small neighborhood around the base point, a successive refinement procedure is usually needed to improve the accuracy of the solution. Despite this, many applications have proved that linear programming methods are computationally very efficient and reliable with ease of handling inequality constraints. They appear to be a good compromise between solution speed and accuracy [80, 81, 98]. Linear programming will be employed in this thesis as the solution method for nonlinear OPF problems.

The popularity of linear programming approaches is also due to Karmarkar's paper on an interior point method [51]. His main idea is to solve a constrained problem as a sequence of unconstrained sub-problems based on three theoretical components [97]: Fiacco & McCormick's logarithmic barrier method for optimization with inequalities, Lagrange's method for optimization with equalities, and Newton's method for solving the nonlinear equations of Karush-Kuhn-Tucker (KKT) optimality conditions. With their nice polynominal complexity plus computational efficiency, interior point methods have proved much faster than the traditional simplex methods for large-scale problems, and have become a candidate for many applications [3, 61, 57]. Their promising results in recent OPF applications [94, 97, 45] have also motivated the current thesis research.

As OPF algorithms became faster and their on-line applications became realistic, techniques for OPF solution tracking, after system topology and/or load changes, were developed and applied. Parametric linear or quadratic programming and the continuation method are just a few examples of these techniques [15, 50]. In the meantime, non-conventional methods such as fuzzy modeling and control gave very interesting applications, introducing a new dimension for OPF research and developments [1, 66]. Current OPF algorithms and computer programs demonstrate speed and complexity never seen before. Yet, numerical stability, flexibility in applications, and real-time capabilities are still an issue.

1.1 Motivation

This thesis is concerned with the potential application of interior point methods in the successive linear solution of optimal power flow problems. More specifically, the main objective of this research is to develop and systematically evaluate the infeasible primal-dual path-following algorithms for the efficient solution of real and reactive power dispatch problems. Although interior point methods have received intensive study and achieved significant developments, there are still several questions that deserve more research to further improve the performance of the methods.

These issues include how to dynamically adjust the barrier parameter and Newton step length, how to effectively choose an initial point to reduce the number of iterations for the specific type of problems, and how to explore the problem-dependent data structure to solve linear system of equations more efficiently. In order to exploit the full computational potential of interior point methods for power system optimization problems, it is essential to investigate all issues that influence the performance of the algorithms. The following provides the motivation underlying the present thesis.

- The critical need for a fast and reliable solution of large-scale optimization problems in power system operations, especially in real-time controls.
- The current successful applications and experience on using linear programming to solve various nonlinear power engineering problems.
- The attractive properties of linear programming methods in terms of the solution efficiency and reliability.
- The large-scale problem solving capability of interior point methods due to their polynominal complexity and computational efficiency, as evidenced by the encouraging results in many applications.
- The reality that the linear programming method based on infeasible primal-dual path-following algorithms has not been systematically evaluated in power system applications.
- The lack of thorough investigation and analysis on various implementation issues of interior point methods for power engineering problems.

- The performance of the algorithms is closely related to several factors such as barrier parameter, initial point, Newton step length, etc. Therefore, customizing these factors to a specific application can possibly speed up convergence.
- To the author's knowledge, the influence of linear step size and convergence tolerances on interior point algorithms as well as SLP procedure has not been thoroughly investigated.
- In power engineering, the systematical evaluation of advanced simplex techniques and the comparison of their relative performance with interior point methods have not been done yet.

1.2 Outline

Chapter 1 starts by introducing background materials about the OPF problem. Recent developments in OPF techniques are briefly reviewed. Based on this information, the motivation is given to carry on the proposed research. Then, the outline of the thesis is described and the author's contributions are summarized.

In *Chapter 2*, the OPF problem is formulated as a constrained nonlinear program in terms of all control and state variables, considering both power balance equality and security inequality constraints. Two particular cases of OPF problems are studied in detail, including security-constrained economic dispatch (SCED) and minimum transmission active-power loss reactive power dispatch (RPD) problems. The sparse linear formulations for both SCED and RPD problems are derived based on the decouple and complete load flow models, respectively. An iterative strategy is described to refine the successive linear solutions of OPF problems.

The interior point methods are presented in *Chapter 3*, where a brief review of their recent progress is included. Then, two advanced interior point methods are studied in detail, i.e., infeasible primal-dual algorithm and predictor-corrector primal-dual algorithm. The complete derivations for both algorithms are provided, incorporating lower and upper bounds to meet our special requirements. The common features as well as individual characteristics of the algorithms are analyized. The important issues associated with their implementations are discussed, including the choices of Newton step size, barrier parameter, initial point, and so on.

Chapter 4 presents experimental results of the real and reactive power dispatch problems using the proposed interior point algorithms. Detailed investigation is conducted on those implementation issues to evaluate their impact on the performance of the algorithms. Also, practical issues related to successive linear programming are studied in detail, including the adjustments of linear step size and stopping criteria. In addition, the use of sparse matrix techniques is considered to improve the computational efficiency and reliability.

In *Chapter 5*, numerical experience on using advanced features of a state-of-the-art simplex code is presented. The recent developments in the simplex technology are thoroughly investigated, such as preprocessing, scaling, crashing, steepest-edge pricings and so on. Their influence on large-scale real and reactive power dispatch problems are evaluated. Then, the comparison of relative performance between this simplex code and a predictor-corrector primal-dual interior point algorithm is conducted, and the numerical results on 118 to 2124 bus systems are discussed.

For testing algorithms, *Chapter 6* describes an efficient technique to create large-scale realistic network data. Finally, *Chapter 7* summarizes the conclusions of this work and provides recommendations for future research.

1.3 Contributions

To the author's knowledge, the main contribution of this thesis has been the development and systematic evaluation of advanced interior point methods for the successive linear solution of optimal power flow problems. This contribution includes the following aspects:

1. The detailed derivation and numerical analysis of the *infeasible primal-dual* and the *predictor-corrector primal-dual algorithms*, where both lower and upper bounds are considered for special needs in OPF applications.

2. A thorough investigation of the primal-dual algorithm on such implementation issues as the choices of Newton step size, barrier parameter, and initial point. The proposed heuristic strategies of adaptively changing these parameters have proved very effective in speeding up convergence.

3. Intensive study on the predictor-corrector algorithm has been carried out to evaluate the influence of barrier parameter, initial point, multiple correctors. The ideas of customizing these parameters to OPF applications have improved the algorithm performance dramatically.

4. The practical issues associated with the successive linear programming (SLP) have been investigated to evaluate their impact on the interior point algorithm as well as SLP procedure. Extensive numerical experiments have shown that the proper adjustments of linear step size and tolerances are crucial for achieving fast solution speed while maintaining solution accuracy.

5. The bottleneck of interior point methods is to repeatedly solve the Newton equations for search directions. The use of sparse matrix techniques for such

equations has been investigated, which concludes that the *normal equation method* can produce fast and reliable solution.

6. The recent developments in simplex technology has been investigated to evaluate their impact on power engineering problems. Comparison between the state-of-the-art simplex code and advanced interior point algorithms has been conduced on large-scale OPF problems.

7. An efficient technique has been developed to create realistic network data of different size, topology, and sparsity for testing algorithms.

Chapter 2

Optimal Power Flow Problem

2.1 Introduction

The operator of a power system is constantly facing the problem of adjusting a set of its variables, such as generator power output and terminal voltage, in order to assure the continued economic and secure operation of the system. This is a very difficult task that is usually done through the *optimal power flow* function performed by computers at utility control centers [15]. An optimal power flow procedure determines the optimal steady-state operation of a power system so as to minimize a chosen objective function and satisfy certain physical and operating constraints. The effectiveness of such a control function is not only dependent on the appropriate problem formulation but also on the efficiency of mathematical programming techniques.

This chapter describes the optimal power flow problem and its general solution procedure. The optimal power flow problem is formulated as a constrained nonlinear program in terms of all power system control and state variables [14]. Its constraints include power balance equality and security inequality constraints. In order to reduce the problem size and complexity, the optimal power flow problem is decomposed into the *real* and *reactive* power dispatch problems based on the decoupling effects between the real power/phase angle and the reactive power/voltage magnitude [77]. In this thesis, two particular cases of optimal power flow problems are studied: (i) the security-constrained economic dispatch; and (ii) the minimum real-power transmission loss reactive power dispatch.

In a security-constrained economic dispatch, the total generation cost is minimized by rescheduling the generator real power outputs while keeping the real power balance and security constraints satisfied. Since the impact of voltage magnitudes on the real power scheduling is negligible, a simplified DC load flow model is used to improve the solution efficiency. In the case of a reactive power dispatch, the total real-power losses are minimized by adjusting the generator terminal voltage, transformer tap ratios and shunt susceptance. At the same time, the real/reactive power balance and security constraints are reinforced. Due to its highly nonlinear nature and strong coupling between voltage and phase angles, the reactive power scheduling is simulated by an AC load flow model to improve its solution accuracy.

Linear programming has been widely used in solving nonlinear power system optimization problems. It has been shown that the linear approach is reliable, fast and sufficiently accurate in most applications [82]. Hence, a linearization method is applied in this thesis to deal with the nonlinear optimal power flow problems. Consequently, each of the above real and reactive power dispatch problems are solved as a sequence of linear sub-problems which, in turn, are solved by linear programming methods. To reduce the computational burden, the sparse linear techniques are used for the solution of each linear sub-problem.

2.2 General Problem Formulation

The optimal power flow (OPF) is to maintain the optimal steady state operation of a power system by adjusting a set of control variables while satisfying certain operating and security constraints. It is a typical nonlinear programming problem that can be mathematically expressed as

$$\begin{aligned} & \min f(x) \\ \text{subject to} \quad & \\ & g(x) = 0 \\ & h_l \leq h(x) \leq h_u \\ & x_l \leq x \leq x_u \end{aligned} \tag{2.1}$$

This problem is formulated in terms of all power system control and state variables (x) which comprise real/reactive power generations, phase shifters, shunt susceptance, transformer taps, and voltage angles and magnitudes. The equality constraints $g(x) = 0$ stands for the power balance equations while the inequality constraints are physical and operational limits. The objective function $f(x)$ is usually the total power generation cost or the power system losses, depending on the application.

Due to the size and complexity of the problem, it is a common practice to decompose the optimal power flow problem into real and reactive power problems [77, 18]. This decomposition is based on decoupling the effects of real power/phase angle from reactive power/voltage magnitude. Each of the problems can be approximated by a linear or quadratic programming model and solved by an iterative scheme to a desired accuracy. In the next section, two particular cases of optimal power flow problems are discussed, i.e., the security-constrained economic dispatch, and the minimum real-power transmission losses reactive-power dispatch.

2.2.1 Security-Constrained Economic Dispatch

In the security-constrained economic dispatch the real power outputs of generators are to be determined by minimizing the total operating cost subject to the power balance equality constraints, the security inequality constraints, and the generator operating limits on real power output. The reactive power controls, such as generator terminal voltages, shunt susceptance and transformer taps, are assumed to be fixed. By eliminating the effect of voltage magnitudes, the real power dispatch problem is formulated as follows:

$$\begin{aligned} & \min \textstyle\sum_{k=1}^{n_G} C_k(P_{Gk}) \\ \text{subject to} \quad & \\ & \textstyle\sum_{k=1}^{n_G} P_{Gk} = P_D + P_L(\theta) \\ & F^{min} \leq F(\theta) \leq F^{max} \\ & P_G^{min} \leq P_G \leq P_G^{max} \end{aligned} \tag{2.2}$$

where

n_b — the total number of buses;

n_G — the number of generators;

n_l — the number of transmission lines;

P_G — an $n_G \times 1$ vector of real-power generations;

C_k — the production cost ($/hr) of the *k-th* generator;

P_D — the total real-power demand;

P_L — the total real-power network losses;

F — an $n_l \times 1$ vector of transmission line flows;

θ — an $n_b \times 1$ vector of voltage phase angles;

Superindices *min* and *max* stand for the lower and upper limits of relevant variables. Note that in formulation (2.2) the generation cost $C_k(P_{Gk})$ is expressed as either a piecewise linear or quadratic function of real power generations; the network losses $P_L(\theta)$ and the branch flow $F(\theta)$ are the nonlinear and nonconvex functions of voltage angles. Note also that the real power balance equation is in a compact form which can be replaced by a set of the real power load flow equations. Depending on which form the power balance equation takes, the resulting formulation is called either a compact modeling or a sparse modeling [15]. In this thesis the sparse formulation is used to solve the real power dispatch problem because it is more suitable to the application of the interior point methods, as shown in Chapter 3.

2.2.2 Reactive Power Dispatch

In the reactive power dispatch the real power generations (except on the slack bus) are assumed fixed. The reactive power controls, such as generator terminal voltages, shunt susceptance and transformer taps are to be determined by minimizing the total system losses, subject to the load flow equality constraints, the operating limits on voltages and reactive power generations, and the physical limits on shunt susceptance and transformer tap positions. The problem can be formulated as

$$
\begin{aligned}
& \min P_L(Q_G, t, b_s, V, \theta) \\
\text{subject to} & \\
& Q(Q_G, t, b_s, V, \theta) = 0 \\
& P(Q_G, t, b_s, V, \theta) = 0 \\
& F^{min} \leq F(V, \theta) \leq F^{max} \qquad\qquad (2.3)\\
& Q_G^{mim} \leq Q_G \leq Q_G^{max} \\
& V^{min} \leq V \leq V^{max} \\
& b_s^{min} \leq b_s \leq b_s^{max} \\
& t^{min} \leq t \leq t^{max}
\end{aligned}
$$

where

n_s — the number of shunt susceptance;

n_t — the number of tap-changing transformers;

P_L — the total real-power network losses;

Q_G — an n_G-vector of reactive-power generations;

V — an n_b-vector of bus voltage magnitudes;

b_s — an n_s-vector of shunt susceptance;

t — an n_t-vector of transformer tap ratios;

θ — an n_b-vector of voltage phase angles;

P, Q — $(n_b - 1)$-vectors of power flow equations;

The rest of the parameters are the same as those in the last section. It should be pointed out that the reactive power balance equations in (2.3) are different from the conventional load flow equations, because in the former case the control variables are to be determined, whereas in the latter case most of them are assigned a given value. Another notable point is that the system losses is a nonlinear and nonconvex function of bus voltages. Moreover, the loss function has a strong coupling between voltage and phase angles. As a result, the reactive power dispatch problem is relatively difficult to solve, comparing to its real power counterpart.

In order to remove the phase angle coupling, some suggestions have been made to transform the state variables in terms of the control variables, based on various sensitivity models [74]. However, the calculations of those sensitivities usually involve the computation of an inverse matrix which is not trivial task for a large-scale problem. Also, any changes in the system configuration will result in the complete recalculations of all parameters. In addition, due to some simplification made in the model, the results may not be accurate enough. To avoid these difficulties, the original formulation (2.3) is used in this thesis for the solution of the reactive power dispatch problem. Since this formulation is very sparse, it allows various sparse techniques to be used to exploit the problem data structure.

2.3 Linear Security-Constrained Economic Dispatch

In the security-constrained economic dispatch (SCED) the minimization of the total production cost is accomplished by regulating the real power outputs of generators. It is a nonlinear problem in nature that can be solved through successive linear programming. In this section, a linear model for the SCED problem is derived. To reduce the problem size without affecting the solution accuracy, the following assumptions are used during the linearization:

- All reactive power controls are kept constant and, therefore, are not considered in the linear model. These controls include generator terminal voltages, transformer taps, and switchable shunt susceptance.
- The changes in the voltage magnitude and reactive power due to the real power controls are considered negligible.

Consequently, the linear real power dispatch model involves only the variables of real power generations and voltage phase angles. Its constraints include the real power balance equalities, the line flow security inequalities, and the operating limits on real power generations. In addition, a step length limit is imposed on phase angle changes to assure the validity of the linear model. The detailed formulation is described in the following subsections.

2.3.1 Generation Cost Objective Function

The objective function of the security-constrained economic dispatch is the total generation cost which is the summation of the cost involved by all generators. The production cost of each generator can be expressed as a quadratic function of its real power generation[46]:

$$C_k(P_{Gk}) = a_k + b_k * P_{Gk} + c_k * P_{Gk}^2 \tag{2.4}$$

where P_{Gk} is the real power output of the k-th generator; and the scalar a_k, b_k, and c_k are the coefficients of the constant, first-order, and second-order terms, respectively. By differentiating equation (2.4) with respect to the real power output, the linear incremental cost function will be

$$\Delta C_k(\Delta P_{Gk}) = (b_k + 2 * c_k * P_{Gk})\Delta P_{Gk} \tag{2.5}$$

Finally, the total incremental cost of the problem becomes

$$\Delta C(\Delta P) = \sum_{k=1}^{n_G}(b_k + 2 * c_k * P_{Gk})\Delta P_{Gk} \tag{2.6}$$

where n_G is the number of generators in a power system.

2.3.2 Real-Power Balance Constraints

In the steady-state operation of a power system, the real and reactive power balance must be maintained, which means that at any time the power generations meet all load demands plus the network losses. This balance condition should also be satisfied in determining the optimal operating state of the system. The power balance condition is usually described by the following load flow equations [46]:

$$
\begin{aligned}
P_i &= V_i^2 G_{ii} + V_i \sum_{j \in i} V_j [G_{ij} \cos(\theta_i - \theta_j) + B_{ij} \sin(\theta_i - \theta_j)] \quad i = 1, \ldots, n_b \qquad (2.7) \\
Q_i &= -V_i^2 B_{ii} + V_i \sum_{j \in i} V_j [G_{ij} \sin(\theta_i - \theta_j) - B_{ij} \cos(\theta_i - \theta_j)] \quad i = 1, \ldots, n_b \qquad (2.8)
\end{aligned}
$$

where G_{ii}, G_{ij}, B_{ii} and B_{ij} are elements of the real and imaginary parts of the bus admittance matrix $Y = G + jB$; V_i, V_j, θ_i and θ_j are the magnitudes and phase angles of the voltages at bus i and j, respectively; P_i and Q_i are the real and reactive power injections at bus i. The symbol $j \in i$ under the summation sign refers to all the buses (j) that are connected to bus i.

In the security-constrained economic dispatch problem, since the changes in the voltage and reactive power are negligible, only the real power balance equations (2.7) are considered as the equality constraints. Notice that the power injection P_i is defined as the generation minus the load on the bus i; thus, equation (2.7) can be rewritten as

$$
V_i^2 G_{ii} + V_i \sum_{j \in i} V_j [G_{ij} \cos(\theta_i - \theta_j) + B_{ij} \sin(\theta_i - \theta_j)] - P_{Gi} + P_{Li} = 0 \qquad (2.9)
$$

where P_{Gi} and P_{Li} are the real power generation and the load demand at bus i, respectively ($P_{Gi} = 0$ for a nongeneration bus). Considering power generations and phase angles are unknown variables and all other variables are constant, the linear incremental model of the power balance equation (2.9) can be derived from its first-order Taylor expansion:

$$
H \Delta\theta - \Delta P_G = 0 \qquad (2.10)
$$

where H is the part of Jacobian matrix, whose elements are defined as following:

$$
\begin{aligned}
H_{ii} &= \frac{\partial P_i}{\partial \theta_i} = V_i \sum_{j \in i} V_j [-G_{ij} \sin(\theta_i - \theta_j) + B_{ij} \cos(\theta_i - \theta_j)] \qquad (2.11) \\
H_{ij} &= \frac{\partial P_i}{\partial \theta_j} = V_i V_j [G_{ij} \sin(\theta_i - \theta_j) - B_{ij} \cos(\theta_i - \theta_j)] \qquad (2.12)
\end{aligned}
$$

2.3.3 Real-Power Security Constraints

The security constraint refers to the thermal ratings of the transmission lines. These thermal limits should not be violated in order to keep the system operating safely. Therefore, the actual power flows on transmission lines should be restricted below the above limits:

$$|F_l| \leq F_l^{max} \quad \text{or} \quad -F_l^{max} \leq F_l \leq F_l^{max} \qquad l = 1, 2, \ldots, n_l \tag{2.13}$$

where F_l and F_l^{max} are the line flow and thermal rating of the l-th transmission line. The branch flow F_l is the function of the voltages and phase angles on its two connecting buses. It can be derived from the π equivalent circuit of the transmission line and expressed as follows:

$$F_l = \sqrt{P_l^2 + Q_l^2} \tag{2.14}$$

and

$$\begin{aligned} P_l &= V_i^2 g_{ik} - V_i V_k [g_{ik} \cos(\theta_i - \theta_k) + b_{ik} \sin(\theta_i - \theta_k)] & (2.15) \\ Q_l &= -V_i^2 (b_{ii} + b_{ik}) - V_i V_k [g_{ik} \sin(\theta_i - \theta_k) - b_{ik} \cos(\theta_i - \theta_k)] & (2.16) \end{aligned}$$

where subscript i and k are the two terminal buses of the l-th transmission line; g_{ik} and b_{ik} are the real and imaginary parts of the series admittance, and b_{ii} is the half shunt susceptance of the line. Neglecting the changes in voltages and reactive powers, the first-order Taylor expansion of equation (2.14) is

$$F_l = F_l^{(0)} + w_i \Delta\theta_i + w_k \Delta\theta_k \tag{2.17}$$

where

$$\begin{aligned} w_i &\approx \frac{1}{F_l^{(0)}} \frac{\partial P_l}{\partial \theta_i} = V_i V_k [g_{ik} \sin(\theta_i - \theta_k) - b_{ik} \cos(\theta_i - \theta_k)] / F_l^{(0)} & (2.18) \\ w_k &\approx \frac{1}{F_l^{(0)}} \frac{\partial P_l}{\partial \theta_k} = -V_i V_k [g_{ik} \sin(\theta_i - \theta_k) - b_{ik} \cos(\theta_i - \theta_k)] / F_l^{(0)} & (2.19) \end{aligned}$$

Finally, the linear incremental form of functional constraint (2.13) becomes

$$\Delta F_l^{min} \leq w_i \Delta\theta_i + w_k \Delta\theta_k \leq \Delta F_l^{max} \tag{2.20}$$

where

$$\Delta F_l^{min} = -F_l^{max} - F_l^{(0)}$$
$$\Delta F_l^{max} = F_l^{max} - F_l^{(0)};$$

and $F_l^{(0)}$ is the line flow at the current linearization point for the real power dispatch problem.

2.3.4 Summary of Linear SCED Formulation

The linear formulation of the security-constrained economic dispatch problem can now be explicitly stated as follows:

$$\min c^T \Delta P_G$$

subject to

$$\begin{aligned} &H\Delta\theta - \Delta P_G = 0 \\ &\Delta F^{min} \leq W\Delta\theta \leq \Delta F^{max} \\ &\Delta P_G^{min} \leq \Delta P_G \leq \Delta P_G^{max} \\ &\Delta\theta^{min} \leq \Delta\theta \leq \Delta\theta^{max} \end{aligned} \tag{2.21}$$

where the components of the vectors are defined as

$\Delta\theta_i$ is the variation of the voltage phase angle at bus i.

ΔP_{Gk} is the variation of the real power generation at bus k.

c_k is the incremental cost coefficient of generator k, which is determined by (2.5).

H_{ii}, H_{ij} are the coefficients of power balance constraints, given by (2.11 - 2.12).

w_{ii}, w_{ij} are the coefficients of security constraints, defined by (2.18 - 2.19).

The above formulation is obtained based on the use of the DC load flow model.

2.4 Linear Reactive Power Dispatch

In the reactive power dispatch the minimization of total network losses is accomplished by controlling generator terminal voltages, shunt susceptance and transformer tap ratios. It is the nonlinear and nonconvex problem which can be solved by using linear programming method. Unlike the real power scheduling, the reactive power scheduling is more difficult to solve due to its highly nonlinear nature and strong coupling between the voltage and phase angle. In order to simplify the problem without sacrificing its solution accuracy, the following assumptions are used in deriving the linear reactive power dispatch model:

- The real power controls, i.e., the generator real power outputs, are kept fixed except for the generation on the slack bus. Therefore, they are not considered as variables in the linear model.
- The changes in the real power flow directly caused by the reactive power controls can be ignored. These controls include shunt susceptance and transformer tap ratios.

The second assumption is made because of the fact that, after the real power scheduling has been done, the real power flow is determined. In this case, the transmission losses are mainly caused by the reactive power flow on the network. The reactive power dispatch reduces the transmission losses by better allocating reactive power sources, which improves the system voltage profiles and, thus, eliminates the unnecessary reactive circulation in the network. As a result, the reactive controls have little impact on the real power distributions. Based on the above assumptions, the detailed linear formulation is derived as follows.

2.4.1 Real-Power Transmission Loss Objective Function

The objective function of the reactive power dispatch is the total real-power transmission losses dissipated in a power network. Since the power losses are defined as the total generations minus the total loads of the system, it can be evaluated by adding all the bus real power injections [96]:

$$P_L = \sum_{i=1}^{n_b} P_i \tag{2.22}$$

where P_L is the total real power losses of the system; P_i is the real power injection at bus i, which is defined as the generation minus the load on that bus. By substituting P_i with equation (2.7), the real power losses can be expressed in terms of bus voltages and phase angles:

$$P_L = \sum_{i=1}^{n_b}\{V_i^2 G_{ii} + V_i \sum_{j\in i} V_j[G_{ij}\cos(\theta_i - \theta_j) + B_{ij}\sin(\theta_i - \theta_j)]\} \tag{2.23}$$

Note that for a given bus i, the items with subscript ij in its inner summation of equation (2.23) are related to a transmission component between bus i and bus j. Similarly, for bus j, there are also the same items with subscript ji, except that the phase angle order is reversed. Therefore, the *cos* items are retained but the *sin* items are canceled by each other. As a result, equation (2.23) is reduced to

$$P_L = \sum_{i=1}^{n_b}[V_i^2 G_{ii} + V_i \sum_{j\in i} V_j G_{ij}\cos(\theta_i - \theta_j)] \tag{2.24}$$

By differentiating equation (2.24) with respect to the voltage and phase angle, the linear incremental loss function can be obtained as

$$\Delta P_L = c_\theta^T \Delta\theta + c_v^T \Delta V \tag{2.25}$$

where

$$c_{\theta_i} = \frac{\partial P_L}{\partial \theta_i} = -2V_i \sum_{j \in i} V_j G_{ij} \sin(\theta_i - \theta_j) \tag{2.26}$$

$$c_{v_i} = \frac{\partial P_L}{\partial V_i} = 2V_i G_{ii} + 2 \sum_{j \in i} V_j G_{ij} \cos(\theta_i - \theta_j) \tag{2.27}$$

2.4.2 Real/Reactive-Power Balance Constraints

Although real power changes caused by reactive controls can be neglected, the real power balance equations still need to be satisfied due to the variation in voltages and phase angles. Therefore, both real and reactive power balance equations are considered in the reactive power dispatch problem.

Real Power Balance Constraints

The linear form of real power balance constraints can be derived in a similar way as the case in the security-constrained economic dispatch, except that voltage changes have to be considered. In addition, since the real power generations are kept constant, they are not included in the linear model. Consequently, the real power balance equation (2.9) can be linearized as

$$H\Delta\theta + N\Delta V = 0 \tag{2.28}$$

where H and N are the part of Jacobian matrix; the elements of H are defined by (2.11)-(2.12); and the elements of N are calculated as follows:

$$N_{ii} = \frac{\partial P_i}{\partial V_i} = 2V_i G_{ii} + \sum_{j \in i} V_j [G_{ij} \cos(\theta_i - \theta_j) + B_{ij} \sin(\theta_i - \theta_j)] \tag{2.29}$$

$$N_{ij} = \frac{\partial P_i}{\partial V_j} = V_i [G_{ij} \cos(\theta_i - \theta_j) + B_{ij} \sin(\theta_i - \theta_j)] \tag{2.30}$$

Reactive Power Balance Constraints

The reactive power balance constraint is more complex than its real power counterpart, because it contains not only voltages and phase angles but also the reactive controls, such as reactive power generations, switchable shunt susceptance and tap-changing transformers. Its complete formulation can be obtained by extending the reactive power load flow equation (2.8) to include all the above variables:

$$\begin{aligned} -V_i^2 B_{ii} + V_i \sum_{j\in i} V_j[G_{ij}\sin(\theta_i - \theta_j) - B_{ij}\cos(\theta_i - \theta_j)] + \\ -Q_{Gi} + Q_{Li} + Q_{Si}(b_{si}) + Q_{Ti}(t_i) = 0 \qquad i = 1, 2, ..., n_b \end{aligned} \tag{2.31}$$

where Q_{Gi} and Q_{Li} are the reactive power generation and load at bus i, respectively. Q_{Si} and Q_{Ti} are the reactive power consumptions that are determined by shunt susceptance b_{si} and transformer tap-ratios t_i. The linear incremental form of the reactive power balance equation is then obtained by differentiating the above equation(2.31) with respect to all variables:

$$J\Delta\theta + L\Delta V - \Delta Q_G + S\Delta b_s + T\Delta t = 0 \tag{2.32}$$

where J and L are the part of Jacobian matrix. Their elements can be derived from (2.8) as the derivatives of the reactive power injection versus the phase angle or the voltage,

$$J_{ii} = \frac{\partial Q_i}{\partial \theta_i} = V_i \sum_{j\in i} V_j[G_{ij}\cos(\theta_i - \theta_j) + B_{ij}\sin(\theta_i - \theta_j)] \tag{2.33}$$

$$J_{ij} = \frac{\partial Q_i}{\partial \theta_j} = -V_i V_j[G_{ij}\cos(\theta_i - \theta_j) + B_{ij}\sin(\theta_i - \theta_j)] \tag{2.34}$$

$$L_{ii} = \frac{\partial Q_i}{\partial V_i} = -2V_i B_{ii} + \sum_{j\in i} V_j[G_{ij}\sin(\theta_i - \theta_j) - B_{ij}\cos(\theta_i - \theta_j)] \tag{2.35}$$

$$L_{ij} = \frac{\partial Q_i}{\partial V_j} = V_i[G_{ij}\sin(\theta_i - \theta_j) - B_{ij}\cos(\theta_i - \theta_j)] \tag{2.36}$$

And S is a diagonal matrix whose elements are the reactive power sensitivities with respect to the changes in the switchable VAR source:

$$S_{ii} = \frac{\partial Q_{Si}}{\partial b_{si}} = -V_i^2 \Delta b_{si} \tag{2.37}$$

The elements of the matrix T are the sensitivities of the reactive power change with respect to the transformer tap change. They can be derived from its π equivalent circuit. Assuming a transformer is connected between bus i and j with tap ratio t_i and admittance $g_{ij} + jb_{ij}$, its basic reactive power equation can be expressed as

$$\begin{aligned} Q_{Ti} &= -V_i^2 b_{ij}/t_i^2 - V_i V_j[g_{ij}\sin(\theta_i - \theta_j) - b_{ij}\cos(\theta_i - \theta_j)]/t_i & (2.38) \\ Q_{Tj} &= -V_j^2 b_{ij} + V_i V_j[g_{ij}\sin(\theta_i - \theta_j) + b_{ij}\cos(\theta_i - \theta_j)]/t_i & (2.39) \end{aligned}$$

where Q_{Ti} and Q_{Tj} are the reactive power going into the transformer from its two terminal buses. Then, the reactive power sensitivity with respect to the tap change is obtained by differentiating the above equations against the transformer tap ratio:

$$\begin{aligned} T_{ii} &= \frac{\partial Q_{Ti}}{\partial t_i} = 2V_i^2 b_{ij}/t_i^3 + V_i V_j[g_{ij}\sin(\theta_i - \theta_j) - b_{ij}\cos(\theta_i - \theta_j)]/t_i^2 & (2.40) \\ T_{ji} &= \frac{\partial Q_{Tj}}{\partial t_i} = -V_i V_j[g_{ij}\sin(\theta_i - \theta_j) + b_{ij}\cos(\theta_i - \theta_j)]/t_i^2 & (2.41) \end{aligned}$$

2.4.3 Reactive-Power Security Constraints

The security constraints on branch flows can be derived from the line flow equations presented in Section 2.3.3. However, since the real power controls are not changed, the real power flow can be assumed constant. Therefore, only reactive power flows are considered in the security constraints. In addition, due to their coupling effect, both voltage and phase angle become variables in these constraints.

Neglecting the changes in real power flows, the first-order Taylor expansion of line flow equation (2.14) is

$$F_l = F_l^{(0)} + m_i \Delta\theta_i + m_k \Delta\theta_k + d_i \Delta V_i + d_k \Delta V_k \tag{2.42}$$

where m_i, m_k, d_i and d_k are the line flow sensitivity coefficients with respect to the changes in the phase angle and voltage. They are calculated by differentiating equation (2.14) and (2.16) against phase angle and voltage, respectively,

$$\begin{aligned}
m_i &\approx \frac{1}{F_l^{(0)}}\frac{\partial Q_l}{\partial \theta_i} = -V_i V_k[g_{ik}\cos(\theta_i - \theta_k) + b_{ik}\sin(\theta_i - \theta_k)]/F_l^{(0)} && (2.43)\\
m_k &\approx \frac{1}{F_l^{(0)}}\frac{\partial Q_l}{\partial \theta_k} = V_i V_k[g_{ik}\cos(\theta_i - \theta_k) + b_{ik}\sin(\theta_i - \theta_k)]/F_l^{(0)} && (2.44)\\
d_i &\approx \frac{1}{F_l^{(0)}}\frac{\partial Q_l}{\partial V_i} = \{-2V_i(b_{ii} + b_{ik}) + \\
&\qquad -V_k[g_{ik}\sin(\theta_i - \theta_k) - b_{ik}\cos(\theta_i - \theta_k)]\}/F_l^{(0)} && (2.45)\\
d_k &\approx \frac{1}{F_l^{(0)}}\frac{\partial Q_l}{\partial V_k} = -V_i[g_{ik}\sin(\theta_i - \theta_k) - b_{ik}\cos(\theta_i - \theta_k)]/F_l^{(0)} && (2.46)
\end{aligned}$$

Finally, the linear incremental form of functional constraint (2.13) becomes

$$\Delta F_l^{min} \leq m_i \Delta\theta_i + m_k \Delta\theta_k + d_i \Delta V_i + d_k \Delta V_k \leq \Delta F_l^{max} \tag{2.47}$$

where
$$\begin{aligned}
\Delta F_l^{min} &= -F_l^{max} - F_l^{(0)}\\
\Delta F_l^{max} &= F_l^{max} - F_l^{(0)};
\end{aligned}$$

and $F_l^{(0)}$ is the line flow at the current linearization point for the reactive power dispatch problem.

2.4.4 Summary of Linear RPD Formulation

The linear problem formulation for the minimum transmission loss reactive power dispatch can then be presented as follows:

$$\begin{aligned}
&\min c_\theta^T \Delta\theta + c_v^T \Delta V \\
\text{subject to}\quad & \\
&H\Delta\theta + N\Delta V = 0 \\
&J\Delta\theta + L\Delta V - \Delta Q_G + S\Delta b_s + T\Delta t = 0 \\
&\Delta F^{min} \leq M\Delta\theta + D\Delta V \leq \Delta F^{max} \qquad (2.48) \\
&\Delta Q_G^{min} \leq \Delta Q_G \leq \Delta Q_G^{max} \\
&\Delta V^{min} \leq \Delta V \leq \Delta V^{max} \\
&\Delta b_s^{min} \leq \Delta b_s \leq \Delta b_s^{max} \\
&\Delta t^{min} \leq \Delta t \leq \Delta t^{max}
\end{aligned}$$

where the components of the vectors are defined as

$\Delta\theta_i$, ΔV_i are the variations of the voltage magnitude and phase angle at bus i.

ΔQ_{Gk} is the change of the reactive power generation at bus k.

Δb_{sj} is the variation of the shunt VAR source at bus j.

Δt_k is the tap-ratio change of the transformer at bus k.

$c_{vk}, c_{\theta k}$ are the loss sensitivity coefficients with respect to the voltage and angle changes at bus k. They are determined by (2.26 - 2.27).

H, N, J, L are the sub-Jacobian matrices whose elements are the coefficients of power balance constraints, given by (2.11 - 2.12), (2.29 - 2.30), and (2.33 - 2.36).

S, T are the reactive power sensitivity matrices with respect to the changes in VAR sources and transformer taps, determined by (2.37) and (2.40 - 2.41)

M, D are the line flow sensitivity matrices; their element are coefficients of security inequality constraints, defined by (2.43 - 2.44) and (2.45 - 2.46).

2.5 General Iterative Solution Strategy

We have presented the linear formulation for the real and reactive power dispatch problems, in particular, the security-constrained economic dispatch (SCED) and the minimum transmission loss reactive power dispatch (RPD) problems. For the SCED problem, a decoupled load flow model is used to reduce the problem size and, therefore, improve the solution efficiency. As a result, the linear SCED formulation involves only the variables of real power generations and phase angles. The problem constraints considered include the real power balance equations as well as the real power security constraints on the transmission line flows.

For the RPD problem, however, a full load flow model is adopted to deal with its highly nonlinear nature and strong phase-angle coupling. Some simplifications are made to reduce the computational work without sacrificing the accuracy. Consequently, the linear RPD problem is formulated in terms of all reactive power controls plus voltages and phase angles. Both real and reactive power balance conditions are used as the equality constraints. Also, the reactive power security constraints are included to relieve the overloading of transmission lines. The distinct feature of the above linear formulations is that the network sparsity is retained. Therefore, various sparse techniques can be used to explore the problem data structure.

Due to their nonlinear objective function and constraints, both the SCED and RPD problems are nonlinear programming problems. The linear formulations presented above are only approximations to the original problems. The validity of these models is limited to a small region around a given operating point. Therefore, an iterative procedure is required to update the system operating point as well as the linear models used in the problem formulation. Then, a sequence of the problem solutions can be attempted to determine the optimal solution for the original nonlinear programming problem. In the meantime, a power flow solution should be obtained at every iteration to update the current system operating status.

Note that since the real power dispatch has more economic benefits than its reactive power counterpart does, a practical way is to make the economic generation scheduling first and then do the reactive power scheduling. This common practice is also followed in our solution strategy, i.e., the SCED problem is solved and, then, followed by the RPD problem. Each of these nonlinear problems is solved using the above successive linear programming procedure. The iterative steps involved are described briefly as follows.

step 1 Input the network data such as transmission line and transformer parameters, the generation and load at each bus, etc.

step 2 Run a load flow program to set up an initial operating point or update the current operating point.

step 3 Check the feasibility and optimality conditions. If the convergence is achieved, then stop; otherwise proceed to the next step.

step 4 Formulate the linear model of the optimal power flow problem by using either decoupled or full load flow equations. A step length limit is imposed on the variation of each variable to assure the validity of the model.

step 5 Solve the linear sub-problem by using the advanced interior point algorithms described in Chapter 3. If the sub-problem is found infeasible, then the original problem is considered infeasible and the execution stops.

step 6 Once the optimal solution of the sub-problem is obtained, the set of system control variables are updated and then go back to **step 2**.

Chapter 3

Interior Point Methods

3.1 Introduction

In the successive linear solution of the optimal power flow problem presented in Chapter 2, the most intensive computation part is the repeated solution of sub-linear programming problems. Therefore, to reduce the overall solution time it is essential to use an efficient mathematical programming method. In the last decade, interior point methods have become a viable alternative to the simplex method for solving large sparse linear programming problems [73, 93, 98]. It has been shown that an interior point algorithm not only has polynominal-time complexity but is extremely efficient in practical computations [51, 59]. Thus, interior point methods are used in this thesis to solve sub-linear optimal power flow problems.

The interior point method differs fundamentally from the simplex method in the way they solve a linear programming (LP) problem. The simplex method finds the optimal solution by moving from vertex to vertex along the boundary of the feasible region, which leads to an increasing number of iterations as problem size

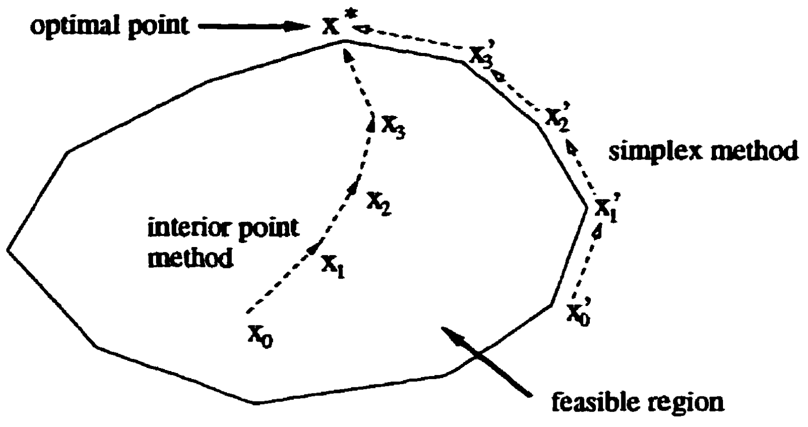

Figure 3.1: Searching approaches of simplex and interior point methods

increases. In contrast, an interior point method solves an LP problem by taking a path through the interior of the feasible region. This results in a remarkable speed up to approach the optimal point. Figure 3.1 illustrates the different approaches used by these two methods in searching for an optimal solution.

This chapter describes interior point methods (IPMs) for the solution of the optimal power flow problems (OPF). Firstly, the recent developments in the theory and implementation of the IPMs are briefly reviewed. It is intended to show that, among the various IPMs, the primal-dual path following method is one of the best IPM found so far. Then, two advanced versions of the primal-dual methods are studied in detail, namely, the infeasible primal-dual [89] and the predictor-corrector primal-dual path following [60] methods. Both algorithms are extended to incorporate lower and upper bounds for special needs in OPF problems. In addition, several important issues closely related to their efficient implementations are discussed, including the adjustment of barrier parameter, the determination of the Newton step length and the initial point, and the improvement of the search direction accuracy. Some heuristics of customizing these parameters to the OPF problems are proposed to reduce the number of iterations and computational time.

3.2 Development of Interior Point Methods

For decades, the simplex method proposed by Dantzig in 1947 has been the most widely used algorithm for solving linear programming problems. However, due to its vertex following property, the solution time of the simplex method may grow exponentially for some specifically-constructed problems (see Klee and Minty [54]). This has motivated researchers to develop a linear programming method with the lower combinatorial complexity. In 1978, Khachiyan first developed a polynominal algorithm by applying the ellipsoid method of Shor *et al.* to linear programming [53]. Although his method can not compete with the simplex method practically, it indeed has significant theoretical implications for combinatorial optimization.

In 1984, Karmarkar [51] introduced his projective algorithm which not only had a polynominal time property but was much faster than the simplex method in practice. His method is called *interior point method* because it searches an optimal point through the interior of the feasible region. Since then, a substantial number of contributions have been made towards the theoretical analysis and practical implementations of the interior point method and its many variants. These variants can be classified into four categories: *projective* methods, *affine scaling* methods, *potential reduction* methods, and *path following* methods.

3.2.1 Projective Methods

Projective methods stem from Karmarkar's projective method [51]. His method is based on two fundamental ideas:

- If the current solution is near the center of the feasible region, then it would get closer to the optimal solution by moving in the steepest descent direction of the objective function.

- The solution space can be transformed so as to place the current solution point near the center of the transformed feasible region. Without changing the problem in any essential way, such a transformation can be done by using an appropriate type of projective transformations.

By formulating a linear programming (LP) problem as a special canonical form, Karmarkar assumed that its optimal value is known, which is very restrictive. He later relaxed this assumption by using a lower bound to estimate the optimal value, and updating the lower bound at each iteration. Todd and Burrell proposed a method to obtain the lower bound from the dual problem [88]. Karmarkar's algorithm requires $O(nL)$ iterations, where n is the number of variables and L is the number of bits required to record the problem. Each iteration involves the calculation of a projection step which, in turn, needs $O(n^3)$ arithmetic operations. He proposed the idea of inexact projection that leads to an average reduction of $O(\sqrt{n})$ in the worst case bound, resulting in $O(n^{2.5})$ arithmetic operations per iteration. Gay also applied the same idea to the dual problem [34]. Anstreicher developed a combined phase I - phase II projective algorithm to relax the initial feasibility assumption [9]. Nevertheless, all of these algorithms need to convert a standard LP problem to Karmarkar's canonical form which causes some loss of sparsity. Moreover, they require to estimate the lower bound of the optimal value and to do the nonlinear projective transformation at every iteration [59].

3.2.2 Affine Scaling Methods

Affine Scaling methods were originally proposed by Dikin in 1967 [23], and later studied by several researchers. Barnes [10] and Vanderbei *et al* [91] proposed a *pri-*

mal affine scaling method as a variant of Karmarkar's projective method. Adler *et al.* suggested applying the affine scaling method to the dual problem, resulting in a *dual affine scaling* method [3]. Both methods do not require Karmarkar's canonical form and can work on the general linear programming problem. Moreover, they use a linear transformation rather than the costly nonlinear projective transformation. The global convergence of the affine scaling methods has been proved by several researchers [23, 10, 91]. Although there has been no evidence of polynominal complexity for this class of algorithms, the algorithms perform practically quite well [2, 3]. The main disadvantage of the affine scaling methods is that, since they do not have the centering direction to keep variables far away from the boundary, a small step size must be imposed to avoid numerical instability. This often causes the algorithms to take more iterations [61].

3.2.3 Potential Reduction Methods

Potential Reduction method was first proposed by Todd and Ye [87]. Their method adopts Karmarkar's idea of using an appropriate potential function to measure the progress of an algorithm but avoids applying expensive projective transformation at each iteration. The method also uses the idea of affine scaling method to reduce the potential function by searching along the projected gradient of the potential function. Therefore, the potential reduction method has the features of both projective methods and affine-scaling methods. However, in order to determine the optimal step size, a line search has to be carried out at every iteration, which can be costly in computations. The potential reduction methods were later studied by several researchers including Ye [104], Freund [30], Gonzaga [43], Anstreicher [8], etc. For this class of methods, the best complexity achieved so far is $O(\sqrt{n}L)$ iterations [87]. Nevertheless, their computational performance highly relys on a proper potential function as well as an efficient line search algorithm.

3.2.4 Path Following (Logarithmic Barrier) Methods

Path Following methods are based on applying Newton's method to follow the central path of the feasible region. This central path is formed by the optimal solutions of a family of problems defined by a logarithmic barrier function. The logarithmic barrier function approach is attributed to Frisch [31] and is studied in detail by Fiacco and McCormick [28] for nonlinear optimization. The notion of central trajectories was proposed by Karmarkar [51] and has been studied extensively by Bayer and Lagarais [11], and Megiddo and Shub [63]. Megiddo suggested applying the logarithmic barrier method to the primal and dual problems simultaneously [64]. His idea was developed by Kojima *et al.* into a primal-dual path following algorithm which requires $O(nL)$ iterations [55]. Later, Monteiro and Adler [69] improved Kojima *et al.*'s results by using ideas of Gonzaga [42] and Karmarkar [51] to obtain a primal-dual algorithm which requires $O(\sqrt{n}L)$ iterations, the best worst-case complexity to date. Since then, several variants of the primal-dual path following method have been proposed and extensively studied, including the primal-dual algorithms of McShane [61] and Lustig *et al.* [57], the Predictor-Corrector algorithms of Mizuno *et al.* [67] and Mehrotra [65], and the infeasible algorithms of Zhang [105], Mizuno [68] and Vanderbei [89].

The distinctive features of the path following methods come from several aspects [59]: (a) following the central path allows the algorithms to take a large step toward the optimal point; (b) applying Newton direction produces quadratic convergence speed; and (c) using different step lengths in primal and dual space results in fast convergence. As a result, this class of methods performs extremely well in practice, comparing favorably to other interior point methods [7, 61]. Since these methods not only have the best polynominal complexity but are computationally most efficient, they have been chosen in this thesis to solve the optimal power flow problems. In the following sections, their two advanced versions are presented in detail, namely, the infeasible primal-dual path following algorithm and its predictor-corrector variant.

3.3 An Infeasible Primal-Dual Algorithm

The infeasible primal-dual interior point algorithm (PDIPA) is based on the one-phase primal-dual path following method [89]. The original algorithm operates on linear programming (LP) problems that have only upper bounds. Since in the sub-linear optimal power flow (OPF) problem all variables are subject to the low and upper limits, we make an extension so that the algorithm can handle a general LP problem with both lower and upper bounds. Such an LP problem is normally formulated in the standard form as,

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & l \leq x \leq u \end{aligned} \tag{3.1}$$

where c is n-vector cost-coefficients; x is n-vector of unknowns; A is $m \times n$ constraint matrix; b is m-vector right-hand-sides; l and u are n-vector lower and upper bounds; n and m are the number of variables and equalities, respectively. It is obvious that the linear OPF formulations (2.21) and (2.48) comply with the above form except for the "functional constraints": $\Delta F_l^{min} \leq w^T \Delta\theta \leq \Delta F_l^{max}$. We replace such a constraint with an equality constraint plus a bound constraint by introducing a new variable $\Delta\theta_f$,

$$\begin{aligned} w^T\Delta\theta - \Delta\theta_f &= 0 \\ \Delta F_l^{min} \leq \Delta\theta_f &\leq \Delta F_l^{max} \end{aligned} \tag{3.2}$$

Thus, it is possible to transform the LP OPF problem into the exact form of (3.1).

The basic concept of the primal-dual path following method is to solve a constrained optimization problem as a sequence of unconstrained problems. Its theoretical foundation consists of three important parts [97]: logarithmic barrier method for optimization with inequalities, Lagrange's method for optimization with equalities, and Newton's method for solving the nonlinear equations of Karush-Kuhn-Tucker (KKT) first-order necessary conditions. Based on these observations, the infeasible primal-dual path following algorithm for LP problem (3.1) can be derived as follows. By introducing slack variables s and v to convert the bound constraints into equality constraints, the LP problem (3.1) is rewritten as

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x - v = l \\ & x + s = u \\ & v \geq 0, s \geq 0 \end{aligned} \tag{3.3}$$

Its dual problem is:

$$\begin{aligned} \max \quad & b^T y - u^T w + l^T z \\ \text{subject to} \quad & A^T y - w + z = c \\ & z \geq 0, w \geq 0 \end{aligned} \tag{3.4}$$

where y is an m-vector of dual variables; z and w are n-vectors of dual slack variables. It is assumed that the constraint matrix A has full row rank, and that both primal and dual problems are feasible and have bounded solutions.

Applying the barrier method to the primal problem (3.3) to eliminate the inequality constraints by incorporating them into a logarithmic barrier term that is appended to the objective function, the original problem is converted to a sequence of problems parameterized by the barrier parameter μ, i.e.,

$$\begin{array}{ll} \min & c^T x - \mu(\sum_{j=1}^n \ln v_j + \sum_{j=1}^n \ln s_j) \\ \text{subject to} & Ax = b \\ & x - v = l \\ & x + s = u \end{array} \tag{3.5}$$

The Lagrangian function associated with (3.5) is,

$$\begin{aligned} L(x, v, s, y, z, w, \mu) &= c^T x - \mu(\sum_{j=1}^n \ln v_j + \sum_{j=1}^n \ln s_j) - y^T(Ax - b) \\ &\quad - z^T(x - v - l) + w^T(x + s - u) \end{aligned} \tag{3.6}$$

where $\mu > 0$ and is monotonically reduced toward zero as the algorithm iterations progress. Accordingly, the solutions to the above family of problems define the central path of the pair of primal and dual problems (3.3 - 3.4), and finally converge to the optimal solution of the original constrained problem as $\mu \longrightarrow 0$.

For a given μ, the solution of (3.5) is a stationary point of (3.6) which is defined by the Karush-Kuhn-Tucker (KKT) first-order necessary conditions,

$$\begin{aligned} Ax &= b \\ x - v &= l \\ x + s &= u \\ A^T y - w + z &= c \\ SWe &= \mu e \\ VZe &= \mu e \end{aligned} \tag{3.7}$$

where e denotes the n-vector of ones, and capital letters S, W, V, and Z denote the diagonal matrices with the diagonal elements s_j, w_j, v_j, and z_j, respectively. Note that the first four of the above equations are linear and represent the primal and dual feasibility conditions. The last two equations are nonlinear and depend on the barrier parameter μ. They become the complementary conditions when $\mu = 0$, which together with the feasibility constraints provides optimality of the solution.

The primal-dual path following method does not solve the above nonlinear KKT equations exactly. Rather, it applies the one-step Newton method to find the search directions, which yields the following linear equations:

$$\begin{aligned} A\Delta x &= r_x \\ \Delta x - \Delta v &= r_l \\ \Delta x + \Delta s &= r_u \\ A^T\Delta y - \Delta w + \Delta z &= r_y \\ S\Delta w + W\Delta s &= \mu e - SWe \\ V\Delta z + Z\Delta v &= \mu e - VZe \end{aligned} \tag{3.8}$$

where

$$r_x = b - Ax \tag{3.9}$$

$$r_l = l - x + v \tag{3.10}$$

$$r_u = u - x - s \tag{3.11}$$

and $$r_y = c - A^T y + w - z \tag{3.12}$$

denote the residuals of the primal and dual infeasibility (the violations of the primal and dual feasibility constraints), respectively. Since the infeasible primal-dual method does not require feasible points during the optimization process, the above residuals may not be zero. This is the major difference between the infeasible and feasible variants of the path following method. As will be mentioned in a later section, these nonzero items result in a feasibility component in the final search direction. The solution of linear equations (3.8) can be proved to be

$$\begin{aligned} \Delta y &= (ADA^T)^{-1}(r_x + AD\rho) \\ \Delta x &= D(A^T\Delta y - \rho) \\ \Delta s &= r_u - \Delta x \\ \Delta v &= \Delta x - r_l \\ \Delta w &= \sigma - S^{-1}W\Delta s \\ \Delta z &= \gamma - V^{-1}Z\Delta v \end{aligned} \tag{3.13}$$

where

$$\begin{aligned} D &= (S^{-1}W + V^{-1}Z)^{-1} \\ \rho &= r_y + (\sigma - \gamma) - (S^{-1}Wr_u + V^{-1}Zr_l) \\ \sigma &= \mu S^{-1}e - w \\ \gamma &= \mu V^{-1}e - z \end{aligned} \tag{3.14}$$

The new point is then defined by,

$$\begin{aligned} x &\Leftarrow x + \alpha_p\Delta x \\ v &\Leftarrow v + \alpha_p\Delta v \\ s &\Leftarrow s + \alpha_p\Delta s \\ y &\Leftarrow y + \alpha_d\Delta y \\ z &\Leftarrow z + \alpha_d\Delta z \\ w &\Leftarrow w + \alpha_d\Delta w \end{aligned} \tag{3.15}$$

where α_p and α_d are respective step lengths in the primal and dual spaces chosen to assure the nonnegativity of the primal and dual slack variables (v, s, z, w). Since this new point (3.15) is an approximate solution of the KKT condition (3.7), it resides in a certain neighborhood of the central path for a given μ. As μ decreases to zero, this point approaches the optimal solution of (3.1) by approximately following the central path of the feasible region [89].

In summary, the infeasible primal-dual path following algorithm can now be described quite simply. Start with any initial point satisfying $v, s, w, z \geq 0$ and with $\mu > 0$, we apply one Newton step to equation (3.7) to find a point closer to the central path. We then let this new point be the current point, reduce μ appropriately and start over again until primal and dual feasibility is attained and the duality gap is smaller than a predetermined tolerance.

3.3.1 Stopping Criteria

The above algorithm terminates when the following feasibility and optimality conditions are satisfied [57]. The feasibility conditions are expressed in terms of the relative primal feasibility,

$$\frac{\|b - Ax\|}{1 + \|x\|} \leq \epsilon_f \tag{3.16}$$

the bound feasibility,

$$\frac{\|l - x + v\|}{1 + \|x\| + \|v\|} \leq \epsilon_f \tag{3.17}$$

$$\frac{\|u - x - s\|}{1 + \|x\| + \|s\|} \leq \epsilon_f \tag{3.18}$$

and the dual feasibility

$$\frac{\|c - A^T y + w - z\|}{1 + \|y\| + \|w\| + \|z\|} \leq \epsilon_f \tag{3.19}$$

The optimality condition is defined as the relative duality gap:

$$\frac{\|c^T x - (b^T y - u^T w + l^T z)\|}{1 + \|b^T y - u^T w + l^T z\|} \leq \epsilon_o \tag{3.20}$$

where $\| \cdot \|$ is *1-norm*. The ϵ_f and ϵ_o are the feasibility and optimality tolerances, defined by 10^{-p} where p is the number of digits accuracy in the solution.

3.3.2 Search Directions

In order to more closely examine the infeasible primal-dual path following algorithm, it is helpful to express its search direction (3.13) in terms of the gradient of the cost function c, the barrier parameter μ, and the infeasibility residuals r_x. By defining P_A as the orthogonal projection matrix onto the null space of A,

$$P_A = D - DA^T(ADA^T)^{-1}AD \tag{3.21}$$

the search direction for primal variables x can be decomposed as

$$\begin{aligned}\Delta x &= -P_A c - P_A(S^{-1} - V^{-1})\mu e + \\ &\quad [DA^T(ADA^T)^{-1}r_x + P_A(S^{-1}Wr_u + V^{-1}Zr_l)]\end{aligned} \tag{3.22}$$

where r_x, r_u, and r_l are the residuals of primal infeasibility defined by (3.9 - 3.11).

The first term of the search direction (3.22) is called the *affine direction* which is the projection of the steepest descent of the cost function. Since the affine direction aligns in the null space of A, it reduces the cost function while preserving the current primal feasibility status. The second term is the *centering direction* that forces the next point $x + \Delta x$ away from the boundary of the feasible region ($s = 0$, $v = 0$) so that a large step can be taken in an effort to get more reduction in the cost function. Since the centering direction is also in the null space of A, it does not change the degree of primal infeasibility of the current point. The third term is associated with the residuals of primal and bound infeasibility, and, therefore, is called *feasibility direction* which drives the current point towards the feasible region. It is interesting to note that with the orthogonal projection P_A the bound feasibility r_l and r_u are improved without affecting primal feasibility r_x. Due to the presence of the above three directions, the infeasible primal-dual path following method searches for the optimal point by improving the feasibility and optimality simultaneously.

Note that when the current point is feasible, all infeasibility residuals become zero (i.e., $r_x = r_l = r_u = 0$). In this case, there is no feasibility direction. The resulting search direction contains the affine and centering components, which yields the feasible primal-dual path following algorithm. Furthermore, if $\mu = 0$, then the search direction consists of only the affine direction. Consequently, the algorithm becomes the primal-dual affine-scaling method. Therefore, the feasible primal-dual path following and affine scaling algorithms are just special cases and can be derived from the infeasible path following algorithm directly.

A similar analysis can be done on the search direction for dual variables y, resulting in the following decomposition consisting of three components,

$$\begin{aligned}\Delta y &= (ADA^T)^{-1}[b - AD(S^{-1}Wu + V^{-1}Zl)] + \\ &\quad (ADA^T)^{-1}AD(S^{-1} - V^{-1})\mu e + (ADA^T)^{-1}ADr_y \qquad (3.23)\end{aligned}$$

where r_y is the residual of dual infeasibility defined by (3.12). Each component of the above search direction (3.23) corresponds to the affine, centering and feasibility direction in the dual space, respectively.

In the following sections, several issues that are critical to the successful implementation of the infeasible primal-dual algorithm are discussed in detail, including the determination of the Newton step size α, the adjustment of the barrier parameter μ, and the choice of an initial point x_0.

3.3.3 Step Length α

One advantage of the primal-dual method is that it allows separate step lengths in the primal and dual spaces, as shown in (3.15). This has been proven highly efficient in practice, significantly reducing the number of iterations to convergence [61].

The step lengths α_p and α_d are determined in such a way that the nonnegativity conditions on the variables (v, s) and (z, w) are preserved, respectively. This is done through the following ratio test:

$$\begin{aligned}
\alpha_p &= \min\left\{1,\ \beta \min_{1\le j\le n}\left(-\frac{v_j}{\Delta v_j}, -\frac{s_j}{\Delta s_j},\ \Delta v_j < 0, \Delta s_j < 0\right)\right\} && (3.24)\\
\alpha_d &= \min\left\{1,\ \beta \min_{1\le j\le n}\left(-\frac{z_j}{\Delta z_j}, -\frac{w_j}{\Delta w_j},\ \Delta z_j < 0, \Delta w_j < 0\right)\right\} && (3.25)
\end{aligned}$$

where $\beta \in (0, 1)$ is a scalar factor chosen to prevent nonnegative variables from being zero and, therefore, avoid hitting a boundary. In our computational experience, we initially set $\beta = 0.95$, and then aggressively increase it to $\beta = 0.9995$ when the primal and dual infeasibility is less than a certain tolerance (say 10^{-2}). This has proved more efficient than using a constant value (as suggested in [89]).

3.3.4 Barrier Parameter μ

A crucial step in the infeasible primal-dual path following algorithm is the choice of the barrier parameter μ. In linear programming, several schemes are proposed to choose μ. They are either based on the duality gap [57, 61] or the complementary gap of the LP problem [59, 89]. In our implementation, we use the complementary gap because it is directly related to μ in (3.7). By pre-multiplying both sides of the last two equations of (3.7) by vector e^T, and adding the resultant equations, we get

$$v^T z + s^T w = 2n\mu \tag{3.26}$$

or

$$\mu = \frac{v^T z + s^T w}{2n} \tag{3.27}$$

It is obvious that equation (3.27) gives a measure of "μ" value for the current point. The theory behind the path following method requires that barrier parameter μ

must approach zero as the iterations progress. Therefore, the new value of "μ" should be substantially less than the current value. Following the idea in [89] but including lower bounds, we choose

$$\mu = \lambda \frac{v^T z + s^T w}{2n} \tag{3.28}$$

where $\lambda = 0.1$ unless the primal objective value is less than the dual objective value (which could happen when the primal and dual feasibility has not been achieved); in that case, we boost μ by setting $\lambda = 10$. In [89], the author suggests using $\lambda = 2$; however, we find that such a value is not large enough to prevent the above phenomena from repeating which slows down convergence.

3.3.5 Initial Point x_0

The important feature of the infeasible primal-dual path following algorithm is that no initial feasible point is required. However, the primal and dual slack variables (v, s, z, w) must be strictly positive. There are many sophisticated ways to produce such a starting point [61, 57, 60, 89]. Our numerical experiments show that the approach in [89] is slightly better than the others. Following the ideas in [89] but making some revision, we set x initially as:

$$x = 10\beta\tilde{x} \tag{3.29}$$

where the vector $\tilde{x}$ and scalar β are defined as

$$\tilde{x}_j = \frac{1}{\|A_j\| + 1} \tag{3.30}$$

$$\beta = \frac{\|b\| + 1}{\|A\tilde{x}\| + 1} \tag{3.31}$$

where A_j is the j-th column of constraint matrix A and $\|\cdot\|$ is the *1-norm*. If any component $x_j > \frac{l_j+u_j}{2}$, then this component is reset to $\frac{l_j+u_j}{2}$. The primal slack vectors (v, s) are first set to satisfy bound constraints $v = x - l$ and $s = u - x$ but then modified so that any v_j (s_j) that is less than one is reset to one. The dual vector y is simply set to zero. The dual slack vector (z) is initialized as follows. If $c_j < 0$, then z_j is set to one; otherwise is set to $c_j + 1$. The dual surplus vector w is initialized to meet the dual feasibility constraints: $w = A^T y + z - c$ (note that $y = 0$). In the above equations, we use the *1–norm* instead of *2–norm* [89] because our numerical test shows that the former is much better than the latter.

3.3.6 The PDIPA Algorithm

Based on the above description, the infeasible primal-dual interior point algorithm (PDIPA) can be stated as follows:

step 1 Set an initial point x_0 and y_0 using the procedure described in Section 3.3.5 so that $s_0, v_0, w_0, z_0 > 0$, and initialize the iteration count $k = 0$;

step 2 Check feasibility and optimality conditions (3.16 - 3.20).
If they are satisfied, then stop; otherwise, go to the next step.

step 3 Adjust the barrier parameter μ^k using (3.28);

step 4 Compute the search direction $\Delta x^k, \Delta s^k, \Delta v^k, \Delta y^k, \Delta w^k, \Delta z^k$
by solving the normal equation (3.13);

step 5 Find step sizes α_p^k, $\alpha_d^k \in (0, 1)$ from the ratio test (3.24 - 3.25);

step 6 Update the current point $x^k, s^k, v^k, y^k, w^k, z^k$ from (3.15).

step 7 Set $k = k + 1$ and go to **step 2**.

3.4 A Predictor-Corrector Primal-Dual Algorithm

The infeasible primal-dual algorithm presented in Section 3.3 is based on applying one step of Newton's method to find an approximation solution to the Karush-Kuhn-Tucker conditions (3.7). As a result, the solution at each iteration contains only the first-order information of the primal and dual center trajectory. To improve the algorithm performance, an obvious idea is to introduce the higher-order information to more closely follow the central path. The first higher-order method is due to the work of Karmarkar *et al.* [52]. They developed a power series variant of a dual affine-scaling method. Following their idea, Mehrotra [65] introduced an efficient higher-order predictor-corrector primal-dual algorithm which uses the second-order derivatives to approximate the primal-dual trajectory. His method was later extended by Lustig *et al.* [60] and proved the most efficient in practice.

The predictor-corrector algorithm presented in this section is built on the work of Mehrotra [65] and Lustig *et al.* [60] but extended to incorporate both lower and upper bound for optimal power flow (OPF) problems. Since in their implementations the important algorithm parameters such as the barrier parameter and the initial point are chosen based on a wide spectrum of problems, their approaches may not be suitable to our particular application. To identify the better parameter setting for the OPF problems, extensive numerical experiments have been conducted to investigate the impact of those parameters on the solution efficiency. Some heuristics of adaptively adjusting the barrier parameter and effectively choosing the initial point are proposed to reduce the number of iterations as well as solution time. The detailed description of the algorithm is provided next.

Like the infeasible primal-dual algorithm, the predictor-corrector primal-dual interior point algorithm (PC-PDIPA) is also derived from the KKT first-order necessary conditions (3.7). However, instead of applying Newton's method to the nonlinear equations (3.7) to generate correction terms to the current estimate, we substitute the new point into (3.7) directly, yielding

$$\begin{aligned} A(x+\Delta x) &= b \\ (x+\Delta x)-(v+\Delta v) &= l \\ (x+\Delta x)+(s+\Delta s) &= u \\ A^T(y+\Delta y)-(w+\Delta w)+(z+\Delta z) &= c \\ (S+\Delta S)(W+\Delta W)e &= \mu e \\ (V+\Delta V)(Z+\Delta Z)e &= \mu e \end{aligned} \tag{3.32}$$

where ΔV, ΔZ, ΔS and ΔW are diagonal matrices having the diagonal elements Δv_j, Δz_j, Δs_j, and Δw_j, respectively. By simple algebraic manipulation, the above equation (3.32) is reduced to the equivalent system:

$$\begin{aligned} A\Delta x &= r_x \\ \Delta x-\Delta v &= r_l \\ \Delta x+\Delta s &= r_u \\ A^T\Delta y-\Delta w+\Delta z &= r_y \\ S\Delta w+W\Delta s &= \mu e-SWe-\Delta S\Delta We \\ V\Delta z+Z\Delta v &= \mu e-VZe-\Delta V\Delta Ze \end{aligned} \tag{3.33}$$

where the r_x, r_l, r_u and r_y are the residuals of the primal and dual infeasibility, defined by

$$r_x = b-Ax \tag{3.34}$$

$$r_l = l-x+v \tag{3.35}$$

$$r_u = u-x-s \tag{3.36}$$

and $$r_y = c-A^Ty+w-z \tag{3.37}$$

Note that equation (3.33) is almost identical to Newton equation (3.8) except for the right hand sides of (3.33) that contain the additional nonlinear terms $\Delta S \Delta W$ and $\Delta V \Delta Z$. This is the major difference between the predictor-corrector method and the pure primal-dual method. Since these nonlinear terms are unknown, the step $\Delta x, \Delta v, \Delta s, \Delta y, \Delta z, \Delta w$ can not be solved explicitly from (3.33). To determine a step approximately satisfying (3.33), we apply Mehrotra's *predictor* and *corrector* scheme. In the *predictor* step, we drop the μ and the nonlinear terms and, then, solve the defining equations for a primal-dual affine direction:

$$\begin{aligned} A\Delta\hat{x} &= r_x \\ \Delta\hat{x} - \Delta\hat{v} &= r_l \\ \Delta\hat{x} + \Delta\hat{s} &= r_u \\ A^T\Delta\hat{y} - \Delta\hat{w} + \Delta\hat{z} &= r_y \\ S\Delta\hat{w} + W\Delta\hat{s} &= -SWe \\ V\Delta\hat{z} + Z\Delta\hat{v} &= -VZe \end{aligned} \tag{3.38}$$

The solution can be found as

$$\begin{aligned} \Delta\hat{y} &= (ADA^T)^{-1}(r_x + AD\rho) \\ \Delta\hat{x} &= D(A^T\Delta\hat{y} - \rho) \\ \Delta\hat{s} &= r_u - \Delta\hat{x} \\ \Delta\hat{v} &= \Delta\hat{x} - r_l \\ \Delta\hat{w} &= -w - S^{-1}W\Delta\hat{s} \\ \Delta\hat{z} &= -z - V^{-1}Z\Delta\hat{v} \end{aligned} \tag{3.39}$$

where

$$\begin{aligned} D &= (S^{-1}W + V^{-1}Z)^{-1} \\ \rho &= r_y - w + z - (V^{-1}Zr_l + S^{-1}Wr_u) \end{aligned} \tag{3.40}$$

In the *corrector* step, we use the affine direction in two different ways: (a) to approximate the nonlinear terms in the right-hand sides of (3.33); and (b) to dynamically estimate the barrier parameter μ [see Section 3.4.1]. Once the estimates of nonlinear terms and parameter μ are determined, the actual search direction $(\Delta x, \Delta v, \Delta s, \Delta y, \Delta z, \Delta w)$ are obtained by solving the following linear equations:

$$\begin{aligned} A\Delta x &= r_x \\ \Delta x - \Delta v &= r_l \\ \Delta x + \Delta s &= r_u \\ A^T\Delta y - \Delta w + \Delta z &= r_y \\ S\Delta w + W\Delta s &= \mu e - SWe - \Delta\hat{S}\Delta\hat{W}e \\ V\Delta z + Z\Delta v &= \mu e - VZe - \Delta\hat{V}\Delta\hat{Z}e \end{aligned} \tag{3.41}$$

The final solution thus become,

$$\begin{aligned} \Delta y &= (ADA^T)^{-1}(r_x + AD\eta) \\ \Delta x &= D(A^T\Delta y - \eta) \\ \Delta s &= r_u - \Delta x \\ \Delta v &= \Delta x - r_l \\ \Delta w &= \sigma_1 - w - S^{-1}W\Delta s \\ \Delta z &= \sigma_2 - z - V^{-1}Z\Delta v \end{aligned} \tag{3.42}$$

where

$$\begin{aligned} \sigma_1 &= S^{-1}(\mu - \Delta\hat{S}\Delta\hat{W})e \\ \sigma_2 &= V^{-1}(\mu - \Delta\hat{V}\Delta\hat{Z})e \\ \eta &= \rho + \sigma_1 - \sigma_2 \end{aligned} \tag{3.43}$$

Comparing the final solution (3.42) with the affine solution (3.39), we found that both predictor and corrector steps use the same factorization of the matrix (ADA^T). Therefore, the additional work of the predictor-corrector method is in the extra forward and backward substitution to compute the affine direction (plus the extra ratio test to estimate μ [see Section 3.4.1]). However, what is gained from this extra work is approximate second-order information concerning the central trajectory from the current estimate to the optimal point as μ is varied continuously. With the actual search direction (3.42), the current point is updated by,

$$\begin{aligned} x &\Leftarrow x + \alpha_p \Delta x \\ v &\Leftarrow v + \alpha_p \Delta v \\ s &\Leftarrow s + \alpha_p \Delta s \\ y &\Leftarrow y + \alpha_d \Delta y \\ z &\Leftarrow z + \alpha_d \Delta z \\ w &\Leftarrow w + \alpha_d \Delta w \end{aligned} \tag{3.44}$$

3.4.1 Step Length α

Again, the step lengths α_p and α_d are chosen to preserve nonnegativity conditions on the slack variables v, s, z, w. This is done by first determining the maximum possible step sizes in the primal and dual space,

$$\bar{\alpha}_p = \min_{1\le j\le n}\{-\frac{v_j}{\Delta v_j}, -\frac{s_j}{\Delta s_j}, \ \Delta v_j < 0, \Delta s_j < 0\} \tag{3.45}$$

$$\bar{\alpha}_d = \min_{1\le j\le n}\{-\frac{z_j}{\Delta z_j}, -\frac{w_j}{\Delta w_j}, \ \Delta z_j < 0, \Delta w_j < 0\} \tag{3.46}$$

and then reducing them slightly with a factor $\beta \in (0,1)$,

$$\alpha_p = \min\{1, \ \beta \ \bar{\alpha}_p\} \tag{3.47}$$

$$\alpha_d = \min\{1, \ \beta \ \bar{\alpha}_d\} \tag{3.48}$$

to ensure that the new point is strictly positive. Unlike the pure primal-dual algorithm, however, the predictor-corrector algorithm can take a longer step to get much closer to the boundary because its search direction contains the higher-order information of the central trajectory. With the use of large factor $\beta = 0.99995$, the algorithm works extremely well [60]. This result has also been verified by our numerical experience.

3.4.2 Barrier Parameter μ

One of Mehrotra's contributions is to use the affine direction to dynamically estimate the barrier parameter μ. He suggested testing the possible reduction in the complementary gap that would result from a step in the affine direction,

$$\hat{cg} = (v + \hat{\alpha}_p \Delta\hat{v})^T (z + \hat{\alpha}_d \Delta\hat{z}) + (s + \hat{\alpha}_p \Delta\hat{s})^T (w + \hat{\alpha}_d \Delta\hat{w}) \tag{3.49}$$

where $\hat{\alpha}_p$ and $\hat{\alpha}_d$ are the steps that would actually be taken if the primal-dual affine direction (3.39) were used; they are determined by the standard ratio test (3.45 - 3.48). Generalized to include lower and upper bounds, Mehrotra's estimate for μ is then defined by

$$\mu = \left(\frac{\hat{cg}}{v^T z + s^T w} \right)^2 \left(\frac{\hat{cg}}{n} \right) \tag{3.50}$$

which chooses μ to be small when good progress (a large decrease in complementarity) can be made in the affine direction, and chooses μ to be large when the affine direction produces little improvement. This is justified by the fact that poor progress in the affine direction generally indicates the need for more centering and hence a large value of μ [60].

Lustig *et al.* found that choosing μ according to (3.50) can result in numerically unstable systems as the optimum is approached on poorly conditioned problems [59]. Thus, they initially define μ by using Mehrotra's estimate (3.50) when the current complementary gap satisfies $(v^T z + s^T w) \geq 1$, and then switch to (3.51)

$$\mu = (v^T z + s^T w)/\phi(n) \tag{3.51}$$

where

$$\phi(n) = \begin{cases} n^2 & \text{if } n \leq 5000 \\ n^{3/2} & \text{if } n > 5000 \end{cases} \tag{3.52}$$

when the complementary gap satisfies $(v^Tz + s^Tw) < 1$. Based on their computational experience, Lustig *et al.* claim that their μ estimate strategy is totally satisfactory and much more stable than always using Mehrotra's method (3.50).

When solving optimal power flow problems, however, we found such a choice may slow down the reduction of the duality gap once the primal and dual feasibility is attained. Therefore, we use the feasibility condition (3.53), rather than complementary gap, to determine whether (3.50) or (3.51) should be employed. Our numerical results have also confirmed that using feasibility condition helps improve convergence by fast reducing the duality gap [see Section 4.3.1].

$$\mu = \begin{cases} \hat{c}g^3/(v^Tz + s^Tw)^2 n & \text{if infeasible} \qquad (a) \\ (v^Tz + s^Tw)/\phi(n) & \text{otherwise} \qquad (b) \end{cases} \tag{3.53}$$

3.4.3 Initial Point x_0

The predictor-corrector method can start from any infeasible point as long as primal and dual slack variables (v, s, z, w) are strictly positive. However, as pointed out by Lustig *et al.* [60], the predictor-corrector algorithm is quite sensitive to the initial guess of the optimal solution. They found that for problems with small upper bounds setting the initial point to satisfy bound feasibility can cause computational instability. Therefore, they devise a starting point such that its slack variables are larger than a certain threshold. Based on their numerical test on a wide spectrum of Netlib problems [33], Lustig *et al.* concluded that a relative large initial estimate works best for the predictor-corrector method. Following the way described in [60] but extending to include lower bound, we define the primal and dual thresholds as

$$\xi_1 = \max\{-\min_{1\le j\le n} \tilde{x}_j,\ 100,\ \|b\|/100\} \tag{3.54}$$

$$\xi_2 = 1 + \|c\| \tag{3.55}$$

where $\tilde{x} = A^T(AA^T)^{-1}b$ and $\|\cdot\|$ is the l_1 norm. Then, for each $j = 1, \cdots, n$, choose the initial primal variables x, v and s as

$$\begin{aligned} x_j &= \xi_1 \\ v_j &= \max(\xi_1, x_j - l_j) \\ s_j &= \max(\xi_1, u_j - x_j) \end{aligned} \tag{3.56}$$

For the dual variables, we set $y = 0$, and the pairs z and w to satisfy dual feasibility condition,

$$\begin{array}{ll} z_j = c_j + \xi_2, \;\; w_j = \xi_2 & \text{if } c_j \geq 0 \\ z_j = \xi_2, \;\; w_j = -c_j + \xi_2 & \text{if } -\xi_2 \leq c_j \leq 0 \\ z_j = -c_j, \;\; w_j = -2c_j & \text{if } c_j < -\xi_2 \end{array} \tag{3.57}$$

Considering our particular applications with $b = 0$, the recommended thresholds would be $\xi_1 = 100$ and $\xi_2 = 1 + \|c\|$. As will be shown later by our computational results, these values do not produce the best performance for the optimal power flow problems. In order to identify the most suitable values, it is necessary to try the thresholds with different magnitudes and evaluate their impact on solution efficiency. To achieve this, we propose the following scheme to change the thresholds:

$$\xi_1 = \xi_3 100 \tag{3.58}$$

$$\xi_2 = 1 + \xi_3\|c\| \tag{3.59}$$

where ξ_3 is the user-specified parameter.

Note that the thresholds defined by (3.58 - 3.59) have two distinct features: (1) they start from the values recommended by Lustig *et al.* and, thus, make use of their extensive numerical experience; and (2) by changing the parameter ξ_3, the relative magnitudes of both primal and dual initial points can be adjusted effectively. We have found that changing either primal or dual threshold alone will produce unsatisfactory results. Therefore, we introduce the parameter ξ_3 to balance the primal and dual thresholds. This has been proved to be very effective in computation.

3.4.4 Multiple Corrector Steps

One may recall that in the primal-dual method the Karush-Kuhn-Tucker (KKT) necessary condition (3.7) is a set of nonlinear equations, whose solutions define the central path of the linear program (3.3) as μ varies continuously. Due to the nonlinearity, its accurate solution requires an iterative process which is time-consuming. Fortunately, it has been theoretically proven that the algorithm does not need to exactly follow the central path in order to converge to the optimal solution. Rather, it only needs to be within a certain neighborhood of the central path by approximately solving (3.7) through one-step Newton method [44]. The predictor-corrector scheme is superior to the one-step Newton method in that, by predicting the nonlinear terms in (3.33) followed by the corrector step, its search directions contain second-order information of the primal and dual central trajectory.

The predictor step is responsible for optimization by reducing the primal and dual infeasibility and duality gap. The corrector step keeps the current iterate away from the boundary of the feasible region (thus close to the central path [44]) to improve the chance for a long step to be made in the next iteration. Both steps need to solve the same large, sparse linear system for different right-hand sides. Assuming that a direct method is used, each iteration involves one factorization and two forward/backward solutions. Since the factorization phase is computationally much more expensive than the solution phase, a natural idea is to reuse the factorization in several iterations [51] or, equivalently, to repeat several forward/backward solutions to guess a better next point [65]. This has led to the introduction of high-order terms when computing search directions.

The predictor-corrector algorithm presented above can easily be extended to a higher-order power series by continuing to substitute at each step the Δv, Δs, Δz, Δw found by solving (3.41) back into its right-hind side so that the algorithm is using multiple corrections. The motivation of using multiple corrector steps is to improve the centrality of the next point so as to increase the step sizes in the primal and dual space. It is believed that the complementary gap will be sufficiently reduced if a long step along a primal-dual affine direction is made [38]. Therefore, driving the primal-dual point as close to the central path as possible is an investment that is expected to pay off in the ability to make a larger step in the next iteration.

In fact, multiple corrector steps do improve the convergence by reducing the number of iterations [60]. However, since each corrector step involves one extra forward/backward solution, the overall solution time may not be reduced. Therefore, the additional computational cost incurred by multiple corrector steps should be justified by the offset work due to a reduction in the number of iterations. In general, the maximum number of corrections the algorithm is encouraged to make depends on the ratio of the efforts to solve and to factorize the KKT system [38]. The harder the factorization, the more advantageous the higher-order corrections might prove to be.

To investigate the impact of using higher-order trajectory information on the solution efficiency for optimal power flow problems, we have also considered applying the multiple corrector steps to improve the approximation of the search directions. However, our experience shows that for small problems, due to the reasons explained above, the algorithm is not as efficient as the case of using one-corrector step. In addition, the algorithm exhibits the unstable behavior due to numerical difficulty when solving ill-conditioned problems with some small bound constraints. Therefore, in our implementation only one-corrector step is adopted.

3.4.5 The PC-PDIPA Algorithm

Based on the above detailed description, the predictor-corrector primal-dual interior point algorithm (PC-PDIPA) can be stated as follows:

step 1 Set an initial point $x_0, s_0, v_0, y_0, w_0, z_0$, using (3.56 - 3.57); and initialize the iteration count $k = 0$;

step 2 Check feasibility and optimality conditions (3.16 - 3.20). If they are satisfied, then stop; otherwise, proceed to next step.

step 3 Predictor:
compute the affine direction $\Delta\hat{x}^k, \Delta\hat{s}^k, \Delta\hat{v}^k, \Delta\hat{y}^k, \Delta\hat{w}^k, \Delta\hat{z}^k$
by factorizing and solving the normal equation (3.39);

step 4 Corrector:
(a) adjust the barrier parameter μ^k according to (3.53);
(b) substitute $\Delta\hat{s}^k, \Delta\hat{v}^k$ and $\Delta\hat{w}^k, \Delta\hat{z}^k$ into the right-hand side of (3.42);
(c) compute the actual search direction: $\Delta x^k, \Delta s^k, \Delta v^k, \Delta y^k, \Delta w^k, \Delta z^k$
by a forward and backward substitution using factors produced in **step 3**;

step 5 Find step sizes α_p^k, $\alpha_d^k \in (0, 1)$ from the ratio test (3.45 - 3.48);

step 6 Update the current point $x^k, s^k, v^k, y^k, w^k, z^k$ from (3.44).

step 7 Set $k = k + 1$ and go to **step 2**.

3.5 Summary

The recent developments of interior point methods have been reviewed with the conclusion that the most efficient interior point method found so far is the *primal-dual path following* (or *logarithmic barrier*) method. Due to its best polynominal complexity and computational efficiency, the algorithm has been chosen as the solution method for optimal power flow problems. The two advanced variants, namely, the *infeasible primal-dual* and the *predictor-corrector primal-dual algorithms* have been presented in detail. Both the algorithms are extended to incorporate lower and upper bounds for special needs in our particular applications.

Several issues closely related to the efficient implementation are discussed in detail, including the adjustment of barrier parameter and the determination of step length and initial point. Some heuristics of adaptively changing these parameters are proposed to improve the performance of the algorithms. For the infeasible primal-dual algorithm, these improvements are represented by: (1) an aggressive step increasing strategy based on feasibility condition; (2) the proper boost of the barrier parameter μ for preventing negative duality gap; and (3) a refined initial point procedure. For the predictor-corrector algorithm, these enhancements are reflected in the following aspects: (1) a heuristic adjustment of barrier parameter based on feasibility condition; (2) an improved approach to balance the primal and dual initial point and to effectively adjust their relative magnitudes.

The major advantage shared by these two algorithm resides on the fact that it is not required to have an initial feasible point to start the algorithms. Instead, the feasibility is attained during the process as optimality is approached. Their common feature is to approximately follow the central path of the feasible region with the only difference in that the former uses the first-order while the latter uses the second-order information to approximate the primal and dual centering trajectory. Therefore, they can take a large step along the search direction to speed up cost reduction. In addition, using separate step lengths in the primal and the dual space also help convergence by fast achieving primal and dual feasibility.

Chapter 4

Experimental Results

4.1 Introduction

This chapter presents numerical results on the use of advanced interior point methods for the solution of optimal power flow (OPF) problems. It is intended to show that the performance of interior point methods can be significantly improved through customizing algorithm parameters to the specific problems under study. To this end, several important issues closely related to the efficient implementation are investigated to evaluate their impact on the solution efficiency. These issues include the adjustments of the barrier parameter and Newton step length, the choice of an initial point and tolerance, and the use of sparse matrix techniques for solving the search direction. In addition, practical issues such as the choice of linear step sizes and convergence criteria are also examined and their influence on the behavior of interior point methods as well as successive linear programming is evaluated. Based on extensive numerical experiments, several heuristics are proposed to reduce the number of iterations and to save computational work in every iteration.

For testing and comparing purposes, both the infeasible primal-dual and the predictor-corrector primal-dual algorithms have been implemented in a C program and compiled with the -O2 option. These algorithms are used to solve two particular cases of OPF problems: (1) the security-constrained economic dispatch (SCED); and (2) the reactive power dispatch (RPD) problems. In the case of SCED problems, the generation cost is minimized by controlling the real power generations, whereas in the RPD problems the total real-power transmission losses are reduced by adjusting all reactive power controls such as the generator terminal voltages, the switchable shunt susceptance and the transformer tap ratios. Each problem is formulated by using the sparse linear model presented in Chapter 2. Then, a successive linear programming (SLP) solution strategy that uses the proposed algorithms is applied to the nonlinear problem until a desired accuracy is achieved.

The above numerical tests are conducted on power systems of various sizes, ranging from 118 to 2124 buses, whose specifications are listed in Table 4.1. For each test system, the SCED problem is solved first, followed by the RPD problem. Table 4.2 provides detailed information about each problem, including the number of constraints (rows), variables (columns), and nonzeros in the constraint matrix A, nonzeros in the normal matrix ADA^T as well as in its Cholesky factor L (note that the Cholesky factor is computed after the normal matrix is reordered using the minimum degree heuristic). To evaluate the performance of the algorithms, both running time and total iterations are reported here. However, in the SLP based method there are two types of iterations: the outer-loop (SLP) iterations for linearization process, and the inner-loop (LP) iterations for solving each linear subproblem. Depending on the testing purpose, either or both iterations are provided. Accordingly, computational time corresponds to the relevant type of iterations. The results presented in this chapter are obtained on a SUN SPARCstation 2.

Table 4.1: Specifications of test power systems

Buses	Lines	Transformers	Shunts	Generators	Compensators
118	170	9	14	18	54
236	343	18	28	36	108
354	519	27	42	54	162
708	1082	54	84	108	324
1062	1602	81	126	162	486
2124	3210	162	252	324	972

Table 4.2: Statistical data for the test cases

	Constraint	Nonzeros	Nonzeros	Nonzeros
Case	matrix A	in A	in ADA^T	in L
SCED-118	119 × 136	490	570	953
SCED-236	238 × 273	993	1176	2197
SCED-354	357 × 410	1502	1812	3603
SCED-708	719 × 825	3073	3770	10779
SCED-1062	1079 × 1237	4631	5797	19280
SCED-2124	2150 × 2473	9263	11667	39014
RPD-118	235 × 305	1944	2327	3749
RPD-236	471 × 612	3952	4845	8289
RPD-354	707 × 918	5983	7481	13825
RPD-708	1417 × 1839	12207	15509	41880
RPD-1062	2127 × 2333	18004	23903	77019
RPD-2124	4247 × 5505	36853	48107	157824

4.2 Results with The PDIPA Algorithm

In this study, the infeasible primal-dual interior point algorithm (PDIPA) is tested on the SCED and RPD problems. To investigate the impact of dynamically adjusting the algorithm parameters on the performance for OPF problems, a linear sub-problem for each case is solved using the PDIPA algorithm. Both the number of iterations and solution time reported here are referred to those required by the algorithm to solve one linear sub-problem. From a rigorous mathematical point of view, it is necessary to use a higher standard of stopping criteria to more precisely reflect the effects of different parameter settings. Therefore, a small convergence tolerance $\epsilon = 10^{-8}$ is employed for the feasibility and optimality criteria so that all sub-problems are solved to eight significant digits accuracy.

4.2.1 Influence of Different Parameters

At the beginning, the approaches recommended by [89] are used to choose all algorithm parameters, i.e., the step reduction factor β is set to 0.95; the barrier parameter μ is determined by formula (3.28) with $\lambda = 0.1$, which is boosted to $\lambda = 2$ whenever duality gap becomes negative; and the initial point is chosen according to (3.29-3.31) using the *2–norm*. Table 4.3 shows the iterative process of the generation cost minimization for the 236-bus system. The data listed include the primal and dual objective functions, the absolute value of the duality gap, the absolute values of the primal, bound and dual infeasibility. From these results, several observations can be made as follows.

- It is found that when the feasibility of solutions has not been attained, the duality gap can be negative if the barrier parameter μ is decreased too much.

Table 4.3: Convergence process of PDIPA algorithm on SCED-236 problem

iter	Objective		Duality	Infeasibility		
	primal	dual	gap	primal	bounds	dual
0	2.0070194e+04	1.1614238e+04	8.4560e+03	1.22e+02	1.33e+02	0.00e+00
1	2.4029642e+04	1.8386536e+04	5.6431e+03	5.37e+01	5.88e+01	2.48e-10
2	2.4046782e+04	2.0158588e+04	3.8882e+03	5.26e+01	5.76e+01	3.12e-10
3	2.4186226e+04	2.1842698e+04	2.3435e+03	4.89e+01	5.35e+01	4.61e-10
4	2.5006269e+04	2.4603435e+04	4.0283e+02	3.49e+01	3.82e+01	4.68e-10
5	2.5227610e+04	2.5209867e+04	1.7744e+01	2.95e+01	3.23e+01	4.53e-10
6	2.5514791e+04	2.5788947e+04	$\boxed{\text{-2.7416e+02}}$	2.34e+01	2.56e+01	5.36e-10
7	2.5783130e+04	2.5253805e+04	5.2933e+02	1.90e+01	2.08e+01	6.35e-10
8	2.6097649e+04	2.5871020e+04	2.2663e+02	1.35e+01	1.48e+01	6.75e-10
9	2.6411559e+04	2.6168703e+04	2.4286e+02	8.08e+00	8.84e+00	7.12e-10
10	2.6489089e+04	2.6535316e+04	$\boxed{\text{-4.6227e+01}}$	6.77e+00	7.41e+00	7.14e-10
11	2.6630313e+04	2.5917455e+04	7.1286e+02	4.81e+00	5.26e+00	7.17e-10
12	2.6687111e+04	2.6380988e+04	3.0612e+02	3.88e+00	4.25e+00	7.93e-10
13	2.6787934e+04	2.6691460e+04	9.6475e+01	2.25e+00	2.46e+00	6.64e-10
14	2.6858961e+04	2.6839844e+04	1.9117e+01	1.23e+00	1.35e+00	7.70e-10
15	2.6901829e+04	2.6899931e+04	1.8976e+00	6.58e-01	7.20e-01	7.48e-10
16	2.6905160e+04	2.6939038e+04	$\boxed{\text{-3.3878e+01}}$	6.14e-01	6.72e-01	7.89e-10
17	2.6955533e+04	2.6914418e+04	4.1115e+01	3.07e-02	3.36e-02	8.59e-10
18	2.6954365e+04	2.6948029e+04	6.3357e+00	2.42e-03	2.65e-03	8.76e-10
19	2.6954048e+04	2.6953104e+04	9.4369e-01	1.21e-04	1.33e-04	7.59e-10
20	2.6953998e+04	2.6953860e+04	1.3807e-01	6.06e-06	6.63e-06	7.39e-10
21	2.6953990e+04	2.6953970e+04	2.0083e-02	3.03e-07	3.32e-07	6.85e-10
22	2.6953989e+04	2.6953986e+04	2.9151e-03	1.52e-08	1.66e-08	7.83e-10
23	2.6953989e+04	2.6953989e+04	4.2286e-04	7.81e-10	8.29e-10	7.57e-10
24	2.6953989e+04	2.6953989e+04	6.1309e-05	1.20e-10	4.15e-11	8.24e-10

The remedy for alleviating such a situation is to boost μ in order to increase the centrality of the next point. However, the side effect of doing so is to cause an even larger duality gap which has to be reduced in the subsequent iterations. Therefore, if μ is not boosted large enough, the negative duality gap can occur repeatedly and result in slow convergence, as shown by the data in the boxes.

- The second observation is that a large initial duality gap may also contribute to slow convergence, as shown by the bold data in the first row of the table. This is because, from the duality theory, the duality gap eventually has to be reduced to sufficiently small. Therefore, a large initial gap may need more iterations to decrease. In general, a large duality gap usually results from a large initial point. If a relative small initial guess is adopted, the algorithm may need less iteration efforts to convergence. This is also justified by the fact that in the linearization method the variables are incremental. Thus, its optimal solution is small in magnitude.

- The third observation is that the step reduction factor $\beta = 0.95$ may be too conservative once the feasibility condition is satisfied, which may restrict the progress toward optimality and, therefore, need more iterations to reach the optimal solution.

To improve the performance of the PDIPA algorithm for our particular applications, the approaches of [89] are modified in the following three aspects:

A. The step reduction factor is initially set to 0.95 and then aggressively increased to 0.9995 once the infeasibility is less than a certain tolerance (say, $\epsilon_f \leq 10^{-2}$).

B. A small initial point is chosen by using the *1–norm* instead of *2–norm* to reduce the initial duality gap.

C. The barrier parameter μ is properly boosted by setting $\lambda = 10$ to avoid repeatedly occurring of negative duality gap.

Figure 4.1 shows the effects of using our improved approaches to set algorithm parameters, where case A uses only the above first modification; case A+B is the combination of the above modifications A and B; and A+B+C is the case which incorporates all three modifications. From the results it is clear that our proposed strategies significantly improve the performance of the PDIPA algorithm, reducing the number of iterations over 30% (from 24 to 16).

Table 4.4 and Table 4.5 compare the computational results of the algorithm when this is applied to the SCED and RPD problems, respectively, where the

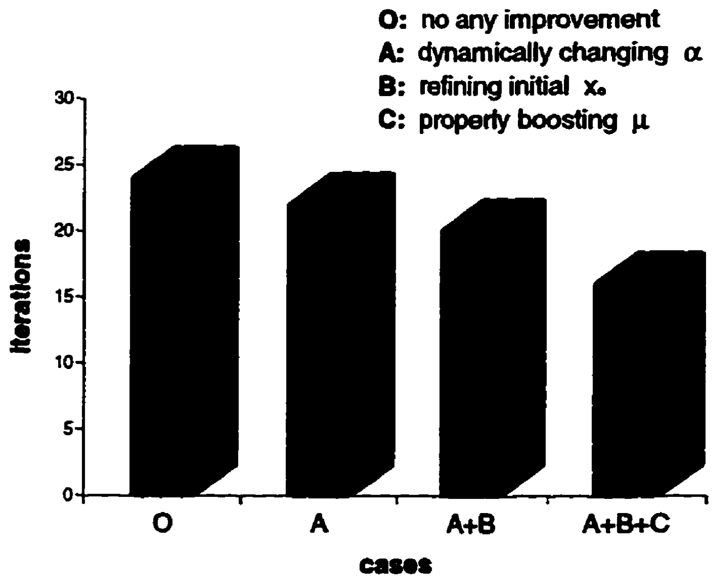

Figure 4.1: Effects of modified approaches on iterations for SCED-236 problem

Table 4.4: Computational results for SCED problems

Cases	PDIPA0		PDIPA1	
	iterations	time(seconds)	iterations	time(seconds)
SCED-236	23	1.07	16	0.78
SCED-354	24	1.85	17	1.33
SCED-708	24	5.28	20	4.48
SCED-1062	28	13.05	25	11.48
SCED-2124	36	39.88	31	33.97

Table 4.5: Computational results for RPD problems

Cases	PDIPA0		PDIPA1	
	iterations	time(seconds)	iterations	time(seconds)
RPD-236	22	4.43	21	4.20
RPD-354	23	7.87	23	7.82
RPD-708	28	34.45	27	33.10
RPD-1062	32	92.90	30	87.08
RPD-2124	45	317.94	41	288.01

PDIPA0 version stands for the algorithm with the parameter settings recommended by [89]; and the PDIPA1 version means the algorithm using our dynamic adjustments described in the above paragraph. For all the test cases, both versions converge to the same solutions, which are given in Tables 4.8 and 4.9. For the SCED problems, the results in Table 4.4 show that our version PDIPA1 improves the algorithm performance, reducing iterations by 10% $\sim$ 30% and saving solution time by 12% $\sim$ 28%. Similarly, as shown by Table 4.5, our version PDIPA1 also produces better performance on RPD problems though the results are less significant than the case of the SCED problems. This is because RPD problems are different from SCED problems in nature. Further study may be necessary to find better parameter settings for RPD problems. In summary, the success of our algorithm comes out of three factors: (1) improving the initial point; (2) properly boosting the barrier parameter μ to prevent repeatedly occurring of negative duality gap; (3) increasing step lengths (α_p, α_d) with the progress of feasibility to maximize the reduction of the duality gap.

4.2.2 Influence of Different Stopping Criteria

As pointed out earlier, to evaluate the impact of different parameter settings more accurately, a very small tolerance $\epsilon = 10^{-8}$ is adopted when conducting the above tests. Practically, however, it is not necessary to use such high convergence criteria since the linear models are only approximations to the nonlinear problems. Therefore, the optimization process can be terminated much earlier by using a relative large tolerance. To see how different stop criteria influence the PDIPA algorithm, Tables 4.6 and 4.7 compare the iteration number and running time obtained by the use of two tolerances values: $\epsilon = 10^{-8}$ and $\epsilon = 10^{-4}$. The results show that for both the SCED and the RPD problems, another 13% $\sim$ 26% reduction in iteration count and CPU time can be achieved by employing the large tolerance $\epsilon = 10^{-4}$. On the other hand, using relative lower accuracy makes little difference in the obtained solutions, as demonstrated in Tables 4.8 and 4.9. For the SCED problems, the maximum relative error is 1.28E-5, with most cases less than 2.0E-6; while for the RPD problems, the maximum relative error is 1.77E-5. Thus, the extra computational efforts due to a small tolerance is not justified in practice.

Table 4.6: Effects of different tolerance for SCED problems

Cases	$\epsilon = 10^{-8}$		$\epsilon = 10^{-4}$	
	iterations	time(seconds)	iterations	time(seconds)
SCED-236	16	0.78	12	0.60
SCED-354	17	1.33	13	1.03
SCED-708	20	4.48	15	3.47
SCED-1062	25	11.48	20	9.40
SCED-2124	31	33.97	27	29.45

Table 4.7: Effects of different tolerance for RPD problems

Cases	$\epsilon = 10^{-8}$		$\epsilon = 10^{-4}$	
	iterations	time(seconds)	iterations	time(seconds)
RPD-236	21	4.20	16	3.32
RPD-354	23	7.82	18	6.28
RPD-708	27	33.10	21	26.42
RPD-1062	30	87.08	24	70.18
RPD-2124	41	288.01	30	211.45

Table 4.8: Minimum cost with different tolerance

Cases	Objective function ($/hr.)	
	$\epsilon = 10^{-8}$	$\epsilon = 10^{-4}$
SCED-236	2.6953989E+4	2.6954050E+4
SCED-354	4.0407611E+4	4.0407906E+4
SCED-708	8.1020767E+4	8.1019724E+4
SCED-1062	1.2097146E+5	1.2097233E+5
SCED-2124	2.4155489E+5	2.4155533E+5

Table 4.9: Minimum losses with different tolerance

Cases	Objective function (×100MW)	
	$\epsilon = 10^{-8}$	$\epsilon = 10^{-4}$
RPD-236	1.9032970	1.9033201
RPD-354	2.7673168	2.7673646
RPD-708	5.4090000	5.4090343
RPD-1062	7.9158772	7.9159350
RPD-2124	16.164108	16.164394

4.2.3 Summary

The infeasible primal-dual interior point algorithm PDIPA has been implemented and tested on the large-scale optimal power flow problems. Several important issues, such as the choices of Newton step length, initial point, and barrier parameter, are addressed and investigated. These parameters are critical for successful implementation of the algorithm. Some suggestions of customizing the above parameters for OPF problems are given to exploit the full potential of the interior point method as applied to the power system optimization problems. These ideas include:

- Aggressively increasing Newton step size based on feasibility condition to maximize the possible reduction of the objective function.
- Properly boosting the barrier parameter to prevent the negative duality gap and, therefore, smoothing the optimization process.
- Refining the starting point by adopting a relatively small initial point in order to reduce the initial duality gap.
- Employing a relatively large convergence tolerance for feasibility and optimality conditions in order to save unnecessary computational work.

Numerical results on 236- to 2124-bus test systems suggest that the above proposed ideas are very effective for improving the performance of the algorithm, significantly reducing the number of iterations as well as solution time.

4.3 Results with The PC-PDIPA Algorithm

The promising results with the PDIPA algorithm encourage us to carry on further studies on a more advanced interior point method—the predictor-corrector primal-dual interior point algorithm (PC-PDIPA). Since the predictor-corrector variant is quite different from the pure primal-dual algorithm discussed earlier, all important implementation issues related to the algorithm should be thoroughly investigated. These issues include the adjustment of barrier parameter, the determination of initial point, and the use of multiple corrector steps. Again, a small tolerance $\epsilon = 10^{-8}$ is used for the feasibility and optimality criteria (3.16 - 3.20) in order to more precisely reflect the impact of different parameter settings. Numerical experiments are conducted on the same set of test problems. For each case, both total iterations and running time are reported to evaluate the algorithm performance.

4.3.1 Effects of Barrier Parameter

Initially, as suggested by [60], we use the complementary gap condition to choose the way of computing the barrier parameter μ. That is, we use (3.53a) when $v^T z + s^T w > 1$; and (3.53b) otherwise. Table 4.10 shows the statistics of iteration counts required by the algorithm, where Column 2 is the number of iterations when primal infeasibility is less than 10^{-6}; Column 3 is the number of iterations to satisfy dual feasibility condition; and Column 4 is the iteration count when the duality gap is less than 1. Finally, Column 5 is the total iterations for solving the problems. From the results, it can be seen that, for all cases (except SCED-708), feasibility is attained at the very beginning of the optimization process, taking less than 34% of the total iterations. The algorithm spends over 90% of the iterations to reduce the duality gap to less than 1. From there on, it takes only very little efforts to reach

Table 4.10: Iteration counts at different stages

Cases	Feasibility		Duality gap	Optimality
	Primal	Dual	(< 1)	
SCED-236	2	0	10	10
SCED-354	2	0	11	12
SCED-708	10	0	18	20
SCED-1062	8	0	24	25
SCED-2124	2	0	32	33

the optimality condition. All of these indicate two facts: (1) using formula (3.53a) to compute μ is effective for obtaining feasibility but may slow down reduction of the duality gap once feasibility is achieved; (2) in this case, however, formula (3.53b) is more efficient to reduce the duality gap.

Instead of using the complementary gap, we use the feasibility condition to change the way of computing μ, as shown in (3.53), i.e., when infeasibility is less than 10^{-6}, we switch from (3.53a) to (3.53b). Table 4.11 compares the number

Table 4.11: Effects of μ adjustment strategies

Cases	Criteria	
	Complementarity	Feasibility
SCED-236	10	10
SCED-354	12	11
SCED-708	20	20
SCED-1062	25	20
SCED-2124	33	27

of iterations required by using these two strategies. The results shows that our strategy that uses the feasibility criterion outperforms the other one by reducing the iterations up to 20%. In addition, it seems that the larger the problem, the more the savings.

4.3.2 Effects of Initial Points

Unlike the pure primal-dual method, the predictor-corrector method is quite sensitive to the starting point. In [60], based on extensive numerical experiments, Lustig *et al.* conclude that the primal and slack variables should be set above a certain threshold to avoid numerical instability, and that a relative large initial estimate works best with the predictor-corrector algorithm. However, our experience shows that this does not apply to the OPF problems. To show how the magnitudes of initial points affect the algorithm, we use different values of ξ_3 in (3.58 - 3.59). Note that in [60] Equations (3.58) and (3.59) are defined as $\xi_1 = 100$ and $\xi_2 = 1 + ||c||$, respectively. We introduce ξ_3 in the above equations to balance the thresholds ξ_1 and ξ_2 for the primal and dual variables. This proves much more effective than changing either ξ_1 or ξ_2 alone. Starting with the values suggested in [60], which correspond to the case $\xi_3 = 1$, we reduce ξ_3 by a factor of 10 until the negative effects appear. Table 4.12 shows the results in terms of the number of iterations, where the last row lists the summations of iterations for all cases using the same value of ξ_3. It is obvious that a small threshold, i.e., a small initial point is preferred for the OPF problems. The best results are given in the case when $\xi_3 = 0.01$, where the iterations are reduced by $20 \sim 30\%$, comparing to the case of $\xi_3 = 1$. Also, this improvement grows with problem size.

The success of using small initial point for SCED problems may result from two

Table 4.12: Effects of different initial points

Cases	$\xi_3 = 1$	$\xi_3 = 0.1$	$\xi_3 = 0.01$	$\xi_3 = 0.001$
SCED-236	10	9	8	10
SCED-354	11	11	10	11
SCED-708	20	18	12	12
SCED-1062	20	20	15	19
SCED-2124	27	30	16	24
Summation	88	88	61	76

reasons. (1) In the linearization method, incremental variables are restricted within certain limits to ensure the validity of a linear model. The solution of such an LP problem is usually small in values. Therefore, it may be helpful to start with a small initial point, hopefully to get closer to the solution. (2) Balancing the primal and the dual thresholds results in small initial complementary gap which requires less computational efforts to reduce it (we found that in this case the initial gap is reduced at least by a factor of 10).

Now, we demonstrate the overall effects of our proposed ideas, i.e., adaptively adjusting the barrier parameter based on the feasibility condition and customizing the initial point by applying small and balanced thresholds. Figure 4.2 compares the number of iterations required by using two versions of the predictor-corrector algorithm, where PC-PDIPA0 uses the approach described in [60] to set the parameters, and PC-PDIPA1 is the version using our suggestions. From the results, one can see that as the problem size increases, the iterations required by PC-PDIPA1 are reduced dramatically (by up to 50%). Table 4.13 shows the running times for the tested problems, from which a similar conclusion can be drawn in terms of computational time.

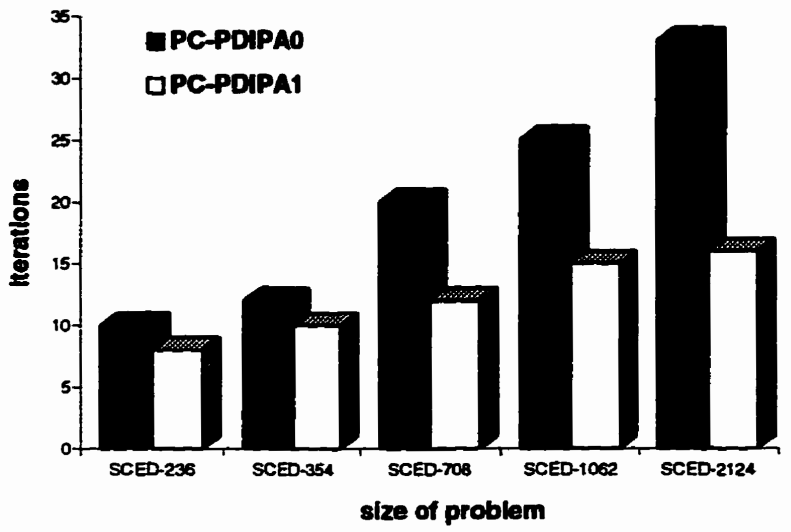

Figure 4.2: Overall effects on iteration counts

Table 4.13: Overall effects on computational time

Cases	time (seconds)		PC-PDIPA1
	PC-PDIPA0	PC-PDIPA1	speedup
SCED-236	0.63	0.53	1.2
SCED-354	1.18	1.07	1.1
SCED-708	5.42	3.50	1.5
SCED-1062	13.85	8.73	1.6
SCED-2124	41.70	20.87	2.0

4.3.3 Effects of Multi-Corrector Steps

As mentioned in Chapter 3., the primal-dual interior point algorithm needs to solve Newton equations to find search directions at every iteration, which involves the numerical factorization of a linear system, followed by the forward/backward solutions. Because the factorization phase is more computationally intensive than the solution phase, we hope to save computational work by re-using such expensive factors in several forward/backward solutions. The benefit of doing so is that the search direction will contain the high-order information of the central trajectory and, hence, increase the centrality of the next point. Eventually, we expect to take large steps to reduce the total iterations (therefore, the number of factorizations). This is the main idea of the higher-order primal-dual methods.

The predictor-corrector algorithm can be easily extended to the higher-order method by applying multiple corrector steps. Each corrector step involves an extra forward/backward solution and an extra ratio test. To see how multiple correctors affect the algorithm performance on optimal power flow problems, Table 4.14 shows the results in terms of the number of iterations and solution time required when different corrector steps are taken, where "— —" means no result is obtained due to numerical difficulty. It is found that the algorithm using more than one corrector shows an unstable behavior. For example, when applying two correctors on problem SCED-354 and RPD-118 the algorithm encounters an ill-conditioning problem. Despite this fact, using multiple corrector steps do reduce iterations. Generally, the more corrector steps are used, the less iterations required. In terms of solution time, however, the results are quite different. In this case, the best choice of the number of corrector steps varies from problem to problem, as shown by the data in bold font. For most problems the algorithm applying one corrector produces the

Table 4.14: Effects of using multiple corrector steps

Cases	1 corrector		2 corrector		3 corrector		4 corrector		5 corrector	
	iters	time	iters	time	iters	time	iters	time	iters	time
SCED-118	**8**	**0.23**	8	0.30	7	0.27	—[a]	—[a]	6	0.33
SCED-236	8	0.60	**7**	**0.58**	7	0.73	7	0.76	6	0.75
SCED-354	**10**	**1.05**	—	—	8	1.18	8	1.37	6	1.20
RPD-118	13	1.62	—	—	10	1.60	8	1.48	**7**	**1.38**
RPD-236	**13**	**4.77**	13	5.43	—	—	12	5.78	9	4.77
RPD-354	**13**	**7.87**	13	8.48	—	—	—	—	10	8.87

[a] "—" no result obtained due to numerical difficulty

best timing performance even though it takes more iterations. This is because each corrector step needs additional computation work, which may not be paid off by the savings due to less iterations. Consequentially, the overall solution time may not be reduced.

One may notice that Table 4.14 presents only the results of small problems. In fact, we have conducted the same test on large problems. However, our experience shows that using any more than one corrector on these problems will encounter numerical difficulty. Closely examining the linear system used for solving search directions (3.41), we found that two factors influence the condition of the system: the constraint matrix A and the diagonal scaling matrix $D = (S^{-1}W + V^{-1}Z)^{-1}$. Poor condition of any of these matrices will cause numerical difficulty. In the solution of large problems, we found that both factors contribute to the ill-conditioned linear system. Firstly, the constraint matrices of all large problems suffer from poor conditioning ($> 10^4$). Secondly, the use of multiple correctors not only increases the centrality but also enforces the feasibility. For those variables with small bound constraints, this will result in very small s_j and/or v_j (large value of $1/s_j$ and/or $1/v_j$) and, thus, worsen the condition of matrix D. Consequently, the entire linear system is severely ill-conditioned, which causes the algorithm using multiple correctors to break down. From the above discussion, we conclude that for our particular problems, the algorithm that uses multiple correctors is not as efficient and stable as the case that employs one corrector. Therefore, in our implementation only one corrector step is adopted.

4.3.4 Comparison with the PDIPA algorithm

As a final note, we compare relative efficiency of the predictor-correct algorithm PC-PDIPA with the pure primal-dual interior point algorithm PDIPA. In this study, the convergence tolerance for feasibility and optimality conditions is also set to $\epsilon = 10^{-8}$. For all test problems, both algorithms converge to the same solutions with eight significant digits. Figure 4.3 compares the number of iterations required by PDIPA and PC-PDIPA algorithms when solving SCED problems. The results show that the predictor-corrector algorithm PC-PDIPA converges much faster than the pure primal-dual algorithm PDIPA, taking 40% $\sim$ 50% less iterations as compared to the latter. Table 4.15 shows the computational time required by both algorithms, where the PC-PDIPA algorithm needs 20% to 40% less CPU time than the PDIPA algorithm. The amount of time (iteration) reductions depends on the problem size; the larger the problem, the more the reduction. One may notice that the time savings of the PC-PDIPA are not as large as the savings in iterations. This is because in every iteration the predictor-corrector algorithm needs an extra forward/backward solution to compute the affine direction and an extra ratio test to estimate the barrier parameter μ.

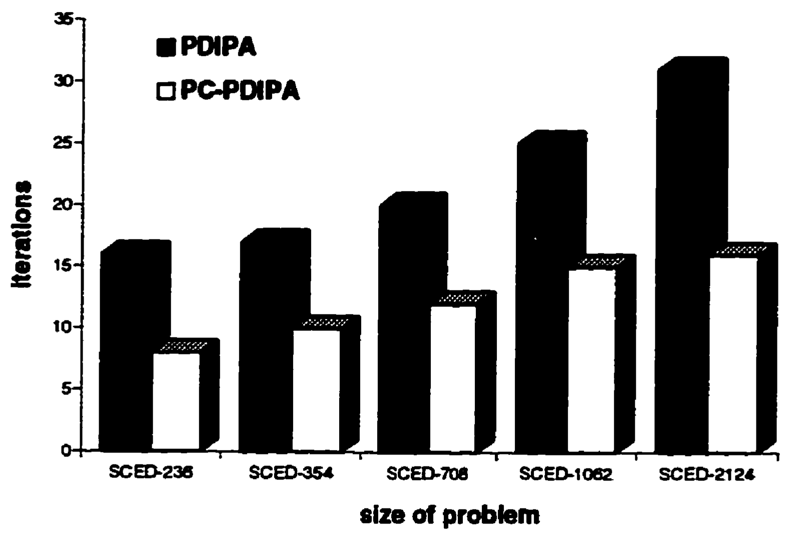

Figure 4.3: Comparison of the number of iterations on SCED problems

Table 4.15: Comparison of solution time on SCED problems

Cases	time (seconds)		ratio (%)
	PDIPA	PC-PDIPA	$\frac{PC-PDIPA}{PDIPA}$
SCED-236	0.78	0.53	68%
SCED-354	1.33	1.07	80%
SCED-708	4.48	3.50	78%
SCED-1062	11.48	8.73	76%
SCED-2124	33.97	20.87	61%

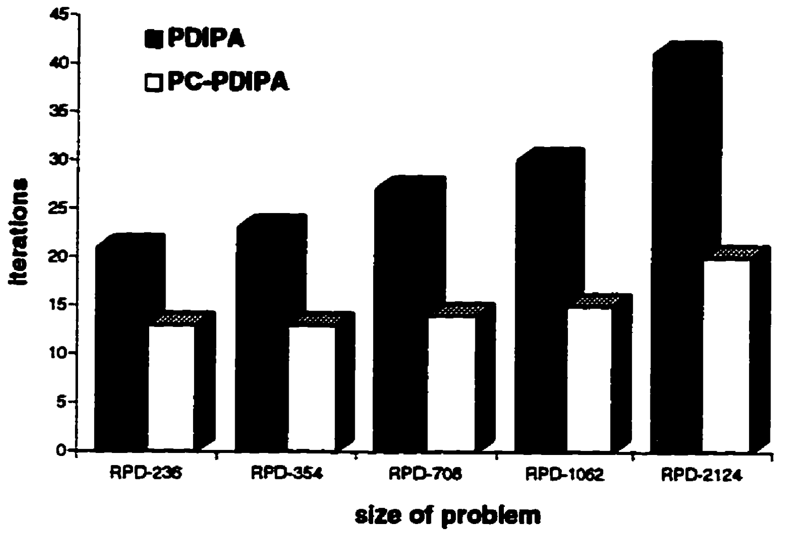

Figure 4.4: Comparison of the number of iterations on RPD problems

Table 4.16: Comparison of solution time on RPD problems

Cases	time (seconds)		ratio (%)
	PDIPA	PC-PDIPA	$\frac{PC-PDIPA}{PDIPA}$
RPD-236	4.20	3.27	78%
RPD-354	7.82	5.50	70%
RPD-708	33.10	20.83	63%
RPD-1062	87.08	49.70	57%
RPD-2124	288.01	157.88	55%

The performance comparisons between the PC-PDIPA and PDIPA algorithms are also conducted on the RPD problems, as shown in Figure 4.4, where the predictor-corrector algorithm PC-PDIPA is over twice as fast as the pure primal-dual algorithm PDIPA, requiring less than half the iterations of the latter. With regard to solution time, as shown in Table 4.16, the PC-PDIPA algorithm still outperforms the PDIPA algorithm, saving computational time by 20% $\sim$ 45%. In addition, the superiority of the PC-PDIPA algorithm over the PDIPA algorithm becomes more evident as the problem size grows.

4.3.5 Summary

The predictor-corrector primal-dual algorithm has been implemented and tested on real and reactive power dispatch problems. Those issues closely related to its efficient implementation, such as the adjustment of barrier parameter, the determination of initial point, and the use of multiple corrector steps, are investigated to evaluate their impact on the optimal power flow problems. Numerical experiments on 118- to 2124-bus systems demonstrate that these issues are critical to the performance of the algorithm. Some ideas are proposed to improve the solution speed. Based on our numerical results, several conclusions can be drawn as follows:

- Using the feasibility condition to adjust the way of computing the barrier parameter can save the total iterations by fast reducing the duality gap.
- Customizing the initial points, by adopting relative small and balanced primal and dual thresholds, significantly reduces the number of iterations.
- Combining the above two strategies shows very promising results, reducing both iterations and computational time by up to 50%. The larger the system,

the more the savings.

- The effectiveness of multiple correctors is dependent on the problem conditioning. As far as the test problems are concerned, using multiple corrector steps is not as efficient and stable as applying one corrector step.

- The number of iterations required by the algorithm is not sensitive to the problem size. The algorithm is numerically reliable.

Comparison with the pure primal-dual interior point method is also conducted. The results reconfirm the superiority of the predictor-corrector method.

4.4 Improvements of OPF Solution Efficiency

In Section 4.2 and 4.3, we have conducted numerical experiments on the pure primal-dual and the predictor-corrector primal-dual algorithms for solving real and reactive power dispatch problems. Our emphasis has concentrated on how to improve the performance of the algorithms by customizing the algorithm parameters to the optimal power flow (OPF) problems. We have also shown that the predictor-corrector method is superior to the pure primal-dual method, about twice as fast as the latter. Now, we are ready to move forward to other practical issues that are directly associated with the successive linear programming (SLP) solution of OPF problems. We will discuss how to use sparse matrix techniques for an efficient solution of large-scale linear equations, which is needed in almost any interior point method. Then, we will demonstrate how to determine linear step sizes and inner/outer-loop convergence tolerances to reduce the total OPF iterations and save computational work in every iteration. Because of its better performance, the predictor-corrector algorithm PC-PDIPA is used in the following study.

4.4.1 Sparse Matrix Techniques

The computational bottle-neck of the primal-dual interior point algorithms is the need to repeatedly solve the Newton equations (3.8) for the search directions. The solution methods for such equations can be classified as either the *augmented equation method* [45, 97] or the *normal equation method* [59, 89]. In our implementation (see equations (3.13)), we use the latter because it is numerically more stable due to its positive-definite matrix ADA^T. Moreover, it needs to compute and store only half of the LU factorization due to symmetry. Furthermore, since only the diagonal matrix D changes from iteration to iteration, the structures of ADA^T and its LU factors remain fixed and thus can be re-used during the optimization process. However, the major difficulty associated with the direct factorization is that many fill-in's will be generated during Gaussian elimination. The common practice is to use a heuristic to reorder the matrix so that fill-in's can be significantly reduced. We follow this practice by applying the most popular heuristic—*minimum degree ordering*. Then, a symbolic factorization is conducted to create a static data structure for the Cholesky factors. Since the matrix structure is fixed through the iterations, the ordering and symbolic analysis are done only once. Finally, the numerical factorization is carried out at every iteration in an efficient way [26, 36].

To examine the relative efficiency of the normal equation method in solving the Newton equations, Table 4.17 shows the total CPU time required to solve one sub-linear programming problem for all cases. The results are obtained by using PC-PDIPA algorithm with the tolerance set to $\epsilon = 10^{-8}$. Table 4.17 also includes the time percentage spent on various tasks in the direct solution of normal equations, such as the formation of matrix ADA^T, the minimum degree ordering, the symbolic and numerical factorization, and the forward and backward solutions. It is obvious that among these tasks the most time-consuming part is the numerical factorization, taking up to 77% of entire solution time. The larger the problem, the more time this part needs. On the contrary, all other tasks require relative less time with the common characteristic that their sharings decrease constantly as the problem size increases; for instance, forming normal matrix ADA^T takes 6 to 15% of the solution time; forward/backward solutions take a similar percentage, around $5 \sim 15\%$; ordering and symbolic factorizing are the least computational intensive parts, requiring only $5 \sim 6\%$ of the total time. To give an idea of how much time is spent in each iteration, Table 4.17 provides this information in the last column where in the SCED problems, the average solution time per iteration is less than

Table 4.17: Percentage of time spent in certain subroutines (%)

Case	Form ADA^T	MMD ordering	Factorization symbolic	Factorization numeric	Solution	Time (sec) total	Time (sec) aver.
SCED-236	10.7	2.8	5.6	22.2	1.3	0.6	0.08
SCED-354	15.1	4.5	4.5	30.3	15.2	1.1	0.11
SCED-708	11.1	2.8	3.2	44.4	11.1	3.6	0.30
SCED-1062	5.9	2.4	2.8	56.6	8.9	8.4	0.56
SCED-2124	7.5	1.7	2.3	60.4	7.5	21.2	1.33
RPD-236	13.5	2.1	2.1	36.1	4.5	4.8	0.37
RPD-354	13.7	1.9	2.1	38.4	5.5	7.9	0.61
RPD-708	12.3	1.8	2.4	63.9	5.6	20.8	1.49
RPD-1062	7.5	1.4	1.8	75.5	5.5	49.7	3.31
RPD-2124	6.3	1.0	1.4	76.8	4.6	157.9	7.89

2 seconds. For the RPD problems this timing in most cases is less than 4 seconds, while for the largest problem RPD-2124 it takes less than 8 seconds to solve the normal equations of 5,000 row/column with 157,824 nonzeros.

4.4.2 Linear Step Sizes Δ

In the successive linear programming (SLP) procedure, incremental variables at every iteration must be restricted within certain limits to ensure the validity of a linear model and convergence of the procedure. These limits (linear step sizes) have large influence on the SLP solution process. Figure 4.5 shows how linear step sizes affect the convergence behavior of the SCED-118 problem, where Case 2 uses a relative small step of 20 MW, and Case 3 employs a large step of 60 MW. It is clear that, initially, using a large step size dramatically speeds up reduction in the cost function. However, after reaching a certain stage, continuously applying large steps cause an oscillatory behavior. Such phenomena can be eliminated by reducing the step size whenever an increase in cost function is observed. Nevertheless, the step size should not be reduced too much so as to cause either slow down or false convergence. It should be larger than a certain threshold. Therefore, we have devised a dynamic adjustment scheme to adaptively change the linear step size. We start with a large step and then reduce it in half whenever the cost function begins to increase, until the step size reaches a certain threshold. The use of this heuristic significantly reduces the number of iterations, as shown by Case 1 where a step size of 60 MW is initially used and then gradually reduced to 20 MW. Figure 4.6 shows the influence of the linear step sizes for the RPD-118 problem, where a similar convergence behavior is observed except that the step sizes used in this case are much smaller than those on the SCED-118 problem.

Table 4.18 shows the convergence results of the SCED-118 and the RPD-118 problems using our heuristic with different initial step sizes. It is obvious that starting with relative large steps plus proper step adjustments generally improve the convergence of SLP process. Notice that in the above results, the thresholds for SCED and RPD problems are chosen as 20 MW and 5 MVAR based on our numerical experiments. The smaller threshold for the RPD problems is due to the highly nonlinear nature of the problems. With regard to sub-linear programming, our experience shows that the predictor-corrector algorithm PC-PDIPA is less sensitive to the bounds (determined by the linear step size). For instance, in the SCED-118 problem, there is only one LP iteration difference when step sizes of 60 MW and 20 MW are used, respectively. Notice also that in conducting the above study both SLP and LP (outer and inner-loop) tolerances are set to 10^{-3}.

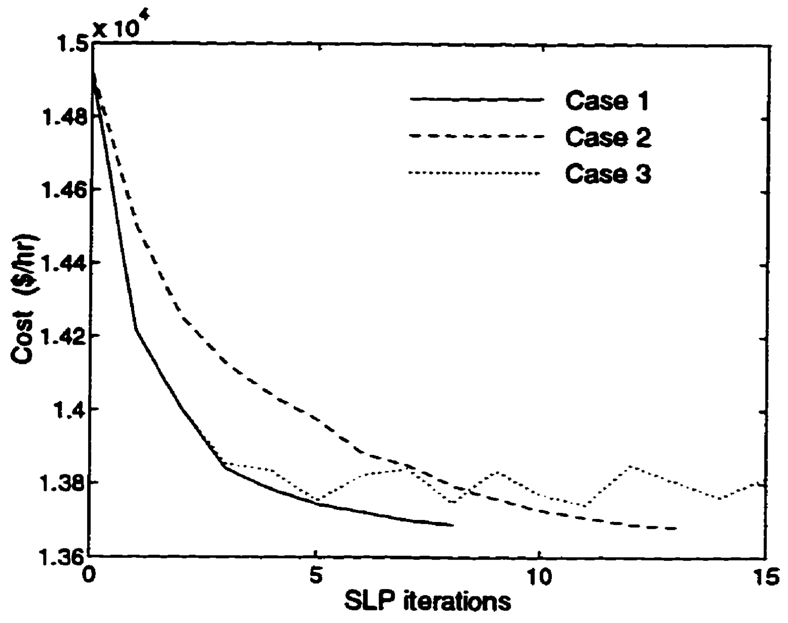

Figure 4.5: The solution process of SCED-118 problem

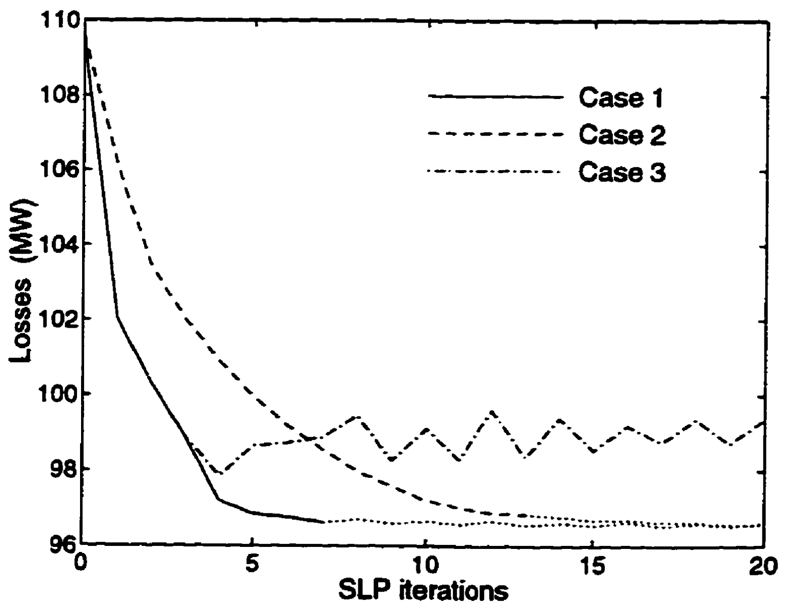

Figure 4.6: The solution process of RPD-118 problem

Table 4.18: Effects of using different initial step sizes Δ

SCED-118		RPD-118	
ΔP_G $\Delta\theta = 15°$	SLP iterations	ΔQ_G, ΔV $\Delta\theta = 7.5°$	SLP iterations
60 MW	8	20 MVAR, 10%	8
20 MW	13	10 MVAR, 5%	9
10 MW	21	5 MVAR, 2.5%	13

4.4.3 Tolerance for Sub-Linear Programming: ϵ

In the successive linear programming (SLP) method, each linear sub-problem (LP) is solved by the PC-PDIPA algorithm, based on feasibility and optimality criteria ϵ which determine the accuracy of the LP solution. In the early stage of the SLP process, as LP solutions are far from the optimal solution of a nonlinear problem (NLP), it is not necessary to solve LP problems very accurately. However, as linear points approach the optimal solution, we may wish to use smaller tolerance to get more accurate LP solutions. Motivated by this fact, we develop a heuristic of dynamically changing LP tolerance ϵ to achieve proper accuracy on the different stages of the SLP process. As shown in the following results, this technique is very effective to reduce computational efforts without sacrifice of accuracy. Figure 4.7 and 4.8 show how tolerance ϵ affects the LP iterations for SCED-118 and RPD-118 problems, respectively, where "Fixed ϵ" means that ϵ is set to 10^{-3} throughout SLP procedure; and "Changing ϵ" is the case where ϵ_0 is initially set to 10^{-1}, and then reduced by half at every SLP iteration until $\epsilon < 10^{-3}$. In both cases the OPF tolerance is set to $\delta = 10^{-3}$. As we expected, in the early stage the LP iterations of

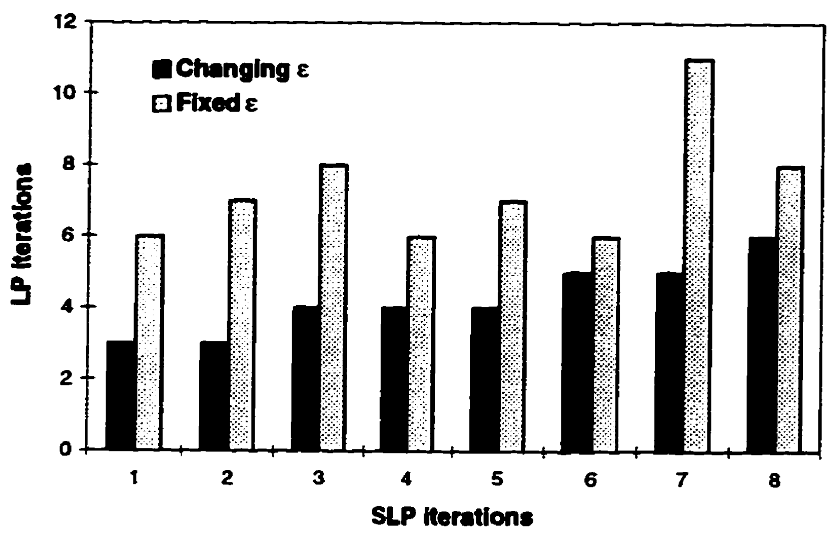

Figure 4.7: LP/SLP iterations for SCED-118 problem

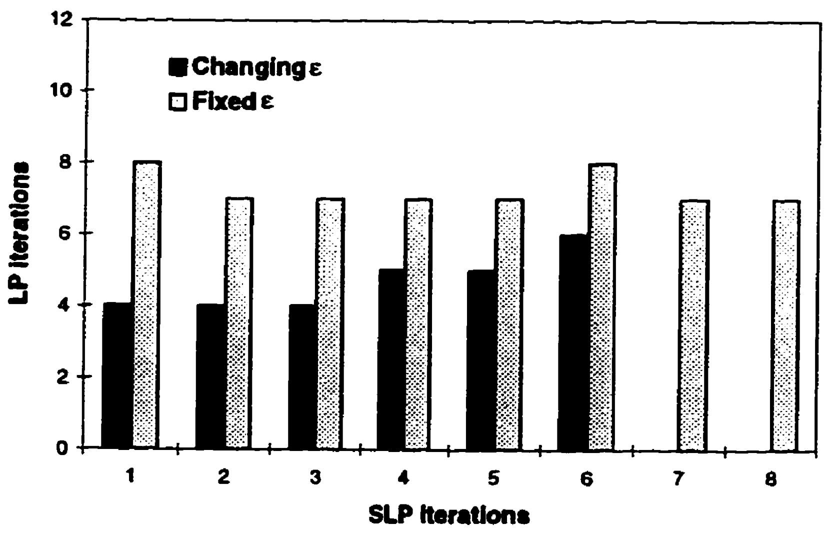

Figure 4.8: LP/SLP iterations for RPD-118 problem

Table 4.19: Effects of changing LP tolerance ϵ

Cases	Total LP/SLP iterations		CPU time savings
	Fixed ϵ	Changing ϵ	
SCED-118	59/8	34/8	45%
SCED-1062	81/9	62/8	22%
RPD-118	58/8	28/6	52%
RPD-1062	87/8	56/7	36%

"Changing ϵ" are much lower than that of " Fixed ϵ"; and they gradually increase with the decrease of ϵ, as the SLP process approaches the solution of the nonlinear OPF problems.

Table 4.19 compares the total LP/SLP iterations required by using the above two schemes. The listing also includes the time saving (%) that is achieved by "Changing ϵ". It can be seen that the proposed heuristic is much faster than the case of "Fixed ϵ", saving 24 $\sim$ 52% in both LP iterations and solution time. In addition, it is found that using the proposed technique not only reduces the sub-LP iterations but also reduces the SLP iterations, as shown in Figure 4.8, though the reason for such phenomenon is unclear so far.

4.4.4 Tolerance for Optimal Power Flow: δ

One advantage of the SLP based methods is that the optimization process can be terminated at an earlier stage, based on the user-specified tolerance δ. This can be done because the power flow equations are satisfied at every linearization step; any sub-linear programming solution can be considered as a sub-optimal solution. We exploit this feature and show that the algorithm performance can be further

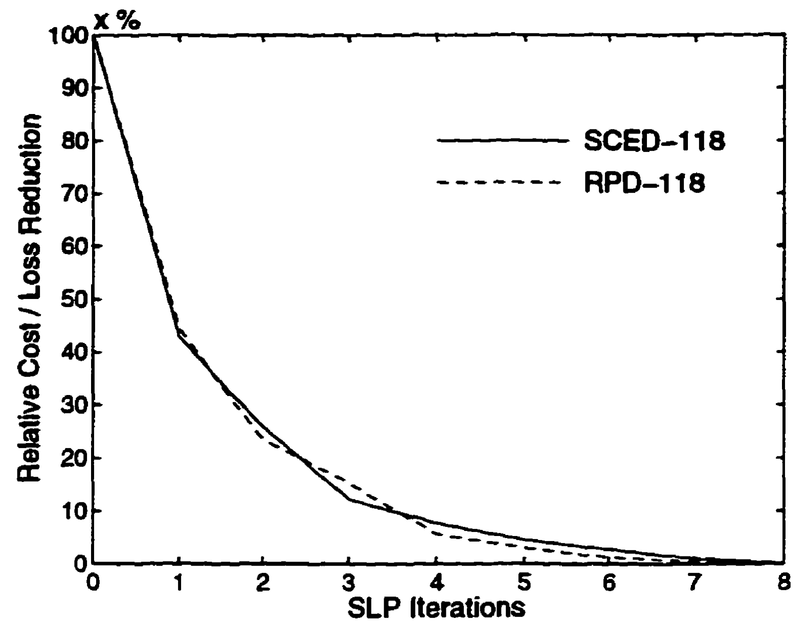

Figure 4.9: OPF solution progress of 118-bus system

improved. Figure 4.9 shows the relative objective reduction of SCED-118 and RPD-118 problems as the SLP iterations proceed, where the OPF tolerance δ is set to 10^{-3}. One can see that objective values decrease rapidly (about 90%) in the first 3 iterations and then slow down reduction for the subsequent iterations. Therefore, in situations where accurate solutions are not required, the SLP process can be terminated at an early stage by either fixing the maximum number of iterations or using a relative large tolerance δ. In this study, we use the latter because it can better control the solution accuracy. Table 4.20 compares the number of SLP iterations and running time obtained by using two different OPF tolerances: $\delta = 10^{-3}$ and $\delta = 10^{-2}$. For all tested problems, around 50% savings in iterations and CPU time are achieved by using the larger tolerance, $\delta = 10^{-2}$. Besides, employing a relative low accuracy is also justified from a practical point of view since extra computational efforts due to small tolerance produce little improvement in the optimal solutions, as demonstrated by the results in Table 4.21 where the maximum relative error is 0.74%.

Table 4.20: SLP iterations and time (seconds) with different δ

Cases	$\delta = 10^{-3}$		$\delta = 10^{-2}$	
	iterations	time (sec)	iterations	time (sec)
SCED-118	8	1.85	4	0.80
SCED-1062	8	42.99	4	20.66
RPD-118	6	3.55	4	2.15
RPD-1062	7	187.16	4	103.61

Table 4.21: OPF solutions with different tolerance δ

Cases	minimum cost (\$/hr)/ losses (MW)		
	$\delta = 10^{-3}$	$\delta = 10^{-2}$	error (%)
SCED-118	13695	13784	0.65%
SCED-1062	122998	123718	0.59%
RPD-118	97.00	97.27	0.28%
RPD-1062	811.80	817.78	0.74%

4.4.5 Summary

The efficient predictor-corrector primal-dual interior point algorithm PC-PDIPA has been successfully applied in the sequential linear solutions of real and reactive power dispatch problems. Practical aspects related to the successive linear programming (SLP) are thoroughly investigated, such as the determination of linear step sizes and outer/inner-loop stopping criteria. Their impacts on the convergence behavior of the PC-PDIPA algorithm as well as SLP procedures are evaluated. Numerical experiments indicate that these factors are crucial to the performance of SLP-based optimal power flow methods. Some heuristics of adaptively changing the linear step size and tolerances are proposed in order to accelerate the convergence of the SLP method, and to reduce the computational work of every iteration. Test results on the 118-bus and 1062-bus systems show that these ideas are very effective, saving up to 50% iterations and computational time.

The major computational work of almost any interior point method is the need to repeatedly solve a set of linear equations for the Newton search directions. The computational time required for solving such linear equations can be prohibitively high for a very large-scale problem. Therefore, it is essential to explore sparsity at every stage by applying various sparse matrix techniques. In addition, numerical reliability should also be considered with high priority when developing a robust algorithm. In our implementations, the *normal equation method* is selected as the solution method due to its numerical stability. The sparsity methods are extensively used in all aspects involved in its solution procedure, including the formation of normal equation ADA^T, the application of minimum degree ordering, symbolic and numerical factorization, and forward and backward solutions. Numerical results on large-scale power systems have verified its efficiency and reliability.

Chapter 5

Numerical Experience with Advanced Simplex

5.1 Introduction

In the past, the simplex method, as an important linear programming technique, was widely used in power system operations and planning [82, 4, 22]. Since its introduced by Dantzig in 1947, the simplex method has experienced many improvements. Various simplifications, extensions and refinements have been made to accelerate its solution speed; and custom-designed algorithms have been implemented to exploit specific problem structures [27, 80, 79]. Due to its practical efficiency, simplex has become the dominant linear programming method for about thirty years. However, as problem size keeps increasing, the traditional simplex codes may need excessive computation time to solve a problem, which makes them uncompetitive as compared to newly developed interior point algorithms [51, 3, 61].

The break-through developments of the simplex method have taken place just in recent years when there have been dramatic changes in computer hardware and software technology. These changes have allowed a wider variety of simplex strategies to be implemented and much larger problems to be studied in detail [12]. As a result, significant advances in the computational efficiency of the simplex method have been achieved, dramatically reducing both computational time and the number of iterations [13]. These advances on the simplex method come from such improvements as better crashing basis procedures, better handling of degeneracy, better partial pricing, implementation of primal and dual steepest edge algorithms, faster and more stable factorizations, better exploitation of cache memory, and better combined phase 1- phase 2 algorithms [59]. One of the new simplex codes, CPLEX, represents such major improvements in the simplex technology.

The rapid progress in simplex methods has raised the following serious questions: how these new techniques affect the solution efficiency for power engineering problems; what are their potential application in power system planning and operations; and, more importantly, what about their relative performance as compared to advanced interior point algorithms. To our knowledge, these concerns have not been sufficiently addressed nor have they been extensively studied. Therefore, it is our belief that there is an urgent need to thoroughly investigate these important issues. This chapter serves such a purpose by presenting our numerical experience of using the state-of-the-art simplex code CPLEX to solve optimal power flow problems. The chapter starts by introducing the CPLEX software as well as its various advanced features. Then, numerical tests on these features are conducted to evaluate their impact on solution time as well as iteration count for power engineering problems. Finally, the comparison of this simplex codes with a predictor-corrector interior point algorithm is carried out to identify the advantages of each method.

5.2 CPLEX™ —The State-of-The-Art Simplex

The CPLEX software package [19] is designed to solve large and difficult problems where other linear programming solvers fail or are unacceptably slow. The package uses several modified simplex algorithms, including primal, dual and network simplex algorithms, with multiple algorithm options for crashing, pricing and factor'zation [12]. An optional preprocessor is available for problem reduction. Besides, CPLEX has many other features such as advanced basis starting, scaling and so on. Most algorithmic parameters can be manually adjusted by the user, although preset defaults with built-in dynamic adjustment often provide the best performance [19].

CPLEX algorithms solve a general linear programming problem with equality and bound constraints. Such a problem can be stated as follows:

$$\begin{aligned} \max \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x_l \leq x \leq x_u \end{aligned} \tag{5.1}$$

Since CPLEX treats bound constraints implicitly, its base matrix is of the order equal to the number of equality constraints. Therefore, the dimension of constraint matrix A determines the size of the problem. Generally, a large problem needs more solution time than a small one does. CPLEX provides a preprocessor to help reduce the problem size. This is achieved by using the Presolver and Aggregator options. The Presolver will work to reduce the number of columns and rows in a problem by simplifying, reducing and eliminating redundancies, whereas the Aggregator will try to eliminate rows by using substitution. CPLEX also has several scaling options to overcome possible numerical difficulties during solution process. These scalings are helpful, especially when a problem is ill-conditioned.

In the solution of simplex methods, one critical factor to the performance is how to construct an initial basis so that the number of iterations can be reduced. This operation is known as "crash". There are several crash parameters to bias the way in which CPLEX orders variables when selecting an initial basis [19]. The essential idea here is to construct a sparse and well-behaved basis, with as much freedom as possible, and having as few artificial variables as possible [12]. One should do some experiments to determine if changing the crash parameter will benefit the problem solution efficiency.

Another critical factor is how to choose a nonbasic variable entering a basis, known as "pricing". CPLEX provides several pricing choices for its primal and dual simplex algorithms. For the primal simplex, they include Reduce-Cost, Steepest-Edge, and Devex pricing (Devex comes from the Latin *devexus* – steep). The Reduced-Cost selects the nonbasic variable that has the most negative reduce cost [21]; the Steepest-Edge is a kind of *normalized* pricing, in which the reduced costs are scaled before selecting the entering variable [35, 13]. The Devex can be viewed as an approximation to the Steepest-Edge pricing [48]. The pricing strategies for the dual simplex include Standard-Dual and variants of the Steepest-Edge [29].

To investigate the performance of these advanced features as applied to power engineering, we use both the primal and dual simplex algorithms to solve optimal power flow problems. The impacts of the preprocessing and scaling are examined by turning them on and off separately. Then, the effects of warm start versus cold start are studied by using advanced bases or the bases constructed by using the CPLEX crash procedures. Also, different pricing techniques such as Reduce-Cost, Steepest-Edge, and Devex are evaluated in terms of their relative efficiency. Finally, the comparison of this simplex code with an advanced interior point algorithm is conducted to identify the merits of each method.

5.3 Testing Results on Advanced Features

In this study we consider four power systems of different sizes, ranging from 118 to 2124 buses [103]. The major data for the test systems are listed in Table 5.1. Several cases studied are shown in Table 5.2, including security-constrained economic dispatch (SCED) as well as reactive power dispatch (RPD)—minimum active power transmission losses. For each test system, the SCED problem is solved first, and then followed by the RPD problem. Both computational time and the number of iterations are used to evaluate the performance of various features of the simplex algorithms on power system optimization problems. The results are obtained on a SUN SPARCstation 2 using the CPLEX software version 3.0.

In CPLEX, there is provision for convergence tolerance ranging from 10^{-9} to 10^{-4}. We have experimented with tolerances of 10^{-4} and 10^{-6}. The results show only a small difference in execution time and the number of iterations. Computational results shown in the following tables correspond to a convergence tolerance of 10^{-6}. Also, it should be pointed out that in this extensive study, both the primal and the dual simplex algorithms are used to solve the same set of problems. Our experience shows that the primal simplex is better than the dual simplex for the

Table 5.1: Specifications of test power systems

Buses	Lines	Transformers	Shunt Capacitors	Generators	Compensators
118	170	9	14	18	54
354	519	27	42	54	162
1062	1602	81	126	162	486
2124	3210	162	252	324	972

Table 5.2: Test cases and problem sizes

	Problem Size		Nonzeros in
Problem-System	Constraints	Variables	Constraints
SCED-118	119	136	490
SCED-354	357	410	1502
SCED-1062	1079	1237	4631
SCED-2124	2150	2473	9263
RPD-118	235	305	1944
RPD-354	707	918	5983
RPD-1062	2127	2333	18004
RPD-2124	4247	5505	36853

cases studied. Therefore, only the results of using the primal simplex are presented hereafter.

5.3.1 Problem Preprocessing

With default parameter settings, if there is no advanced starting basis, CPLEX will first automatically look for opportunities to reduce the size of a problem by using its preprocessor — the Presolver and Aggregator options. The impacts of these preprocessings can be evaluated by turning them on and off. Table 5.3 compares the solution times of the primal simplex with and without these preprocessings. Note that negative value under "Time Savings" column means time increase rather than decrease. Also, the table includes the changes in problem sizes before and after the preprocessings.

From Table 5.3, one can see that with the use of the preprocessings, all the

Table 5.3: Effect of preprocessing on problem size and solution time (seconds)

Cases	No Preprocessing		Presolver and Aggregator	
	Problem Size	CPU Time	Size Reduced	Time Savings
SCED-118	119 × 137	0.25	6 rows 7 columns	- 0.22
SCED-354	357 × 409	2.13	18 rows 17 columns	- 0.05
SCED-1062	1079 × 1225	24.57	52 rows 40 columns	+ 1.20
SCED-2124	2150 × 2449	88.30	95 rows 71 columns	- 4.47
RPD-118	235 × 314	3.15	9 columns	- 0.05
RPD-354	707 × 940	22.73	22 columns	+ 0.68
RPD-1062	2127 × 2818	286.32	64 columns	+ 14.25
RPD-2124	4247 × 5635	1287.90	130 columns	+ 121.73

problems get some reduction in size. In general, the larger a problem, the more the reduction. With regard to the solution time, the results show that the preprocessings have almost no impact on small problems. For large problems, however, two types of problems show quite different results. The real power dispatch problems get no benefits from the preprocessings. Rather, the SCED-2124 problem takes even slightly more CPU time, despite of the fact that 95 rows and 71 columns have been reduced. On the other hand, the large reactive power dispatch problems, such as RPD-1062 and RPD-2124, do benefit from these options, reducing the solution time by 5% to 10%, respectively. In this case, the larger the problem, the more the time savings. One possible reason for this is that the preprocessing may change the structure of a constraint matrix. Therefore, in certain circumstances the reduced problem may become more difficult to solve than the original one. Another reason is that because the preprocessings involve additional computation work, their use may not be justified for the problems either not large enough or unable to take the advantages of preprocessings. From the above discussions, it follows that the impact of the preprocessings varies from problem to problem, depending on the size and nature of a problem under study.

5.3.2 Problem Scaling

A scaling option is provided to scale the constraint matrix when CPLEX reads a problem. This option is mainly used to overcome numerical difficulties that arise from the solution process. Therefore, a poorly conditioned problem (such as the optimal power flow under a heavy loading condition) may benefit from scaling option. Nevertheless, our numerical experience shows that scaling can not only improve numerical stability but sometimes boost performance significantly. Table 5.4 compares the computational time obtained with/without scaling, where Column 2 gives the matrix condition for each problem. The results show that using scaling

Table 5.4: Effect of scaling on solution time (seconds)

	Condition Number	Scaling	
Cases	of Matrix A	No	Yes
SCED-118	1.6773E+3	0.25	0.25
SCED-354	7.5823E+3	2.13	1.80
SCED-1062	2.3725E+4	24.57	20.13
SCED-2124	6.2189E+4	88.30	71.35
RPD-118	8.9232E+3	3.15	2.75
RPD-354	5.6200E+4	22.73	18.88
RPD-1062	1.7601E+5	286.32	219.73
RPD-2124	5.4405E+5	1287.90	920.45

does improve the performance on all cases (except the smallest problem SCED-118). The time savings range from 12% to 29% with increase of problem sizes. Moreover, it is observed that problem RPD-2124, that has the worst matrix condition, benefits most from scaling. These results indicate that using scaling can help solve optimal power flow problems, especially for those problems with poor conditioning.

5.3.3 Crashing and Advanced Basis Starting

CPLEX can start either from an initial basis constructed through its crash procedure (cold start) or from an advanced basis — the solution of a previously solved problem (warm start). The warm start feature is extremely useful when solving optimal power flow problems by successive linear programming (LP). Since each subsequent LP problem is a perturbation of its previous LP problem, the solution of a previous linear step can be used as a starting basis for its subsequent linear step. Table 5.5 shows the number of iterations required by using cold and warm start, respectively, where "Phase I" means the iterations required to satisfy feasibility condition and "Total Iter" means the iterations to reach optimality.

Let us first examine the results obtained by using the CPLEX crash procedure, which are listed under the "Cold Start" column of Table 5.5. One may notice that the total iteration count for each case is roughly equal to the number of rows of the constraint matrix in the relevant problem. As mentioned earlier, a simplex basis also has the size equal to the number of rows. Thus, the number of iterations approximating the size of the simplex base can be interpreted as a good performance. This is because without knowing optimal columns in advance, it will take that many iterations just to pivot in the columns of an optimal basis [12]. Although the CPLEX crash procedure can be considered efficient, its performance

Table 5.5: Iterations of simplex with cold/warm start

	Cold Start		Warm Start	
Cases	Phase I	Total Iters	Phase I	Total Iters
SCED-118	32	43	2	12
SCED-354	108	147	16	54
SCED-1062	328	513	48	189
SCED-2124	623	945	99	307
RPD-118	138	214	17	102
RPD-354	387	616	75	309
RPD-1062	1170	2159	204	1033
RPD-2124	2323	4449	450	2207

is still uncompetitive with the warm start, as shown in the following section.

Now let us compare the relative performance between cold and warm starts. From Table 5.5, one can see that, for all cases, using warm start dramatically speeds up convergence in achieving feasibility as well as optimality, reducing Phase I iterations by a factor of 5 $\sim$ 12 and total iterations by a factor of 2. Table 5.6 compares the computational time of using cold and warm start, where warm start saves at least half CPU time as compared to cold start. These results show that if an advanced basis is available, iterations of simplex can be reduced significantly, particularly if the current problem is similar to the previous problem. Therefore, whenever solving the same or similar problems repeatedly, one should always consider starting from an advanced basis.

Table 5.6: Solution time (seconds) of simplex with cold/warm start

Cases	Cold Start	Warm Start
SCED-118	0.25	0.12
SCED-354	2.13	0.97
SCED-1062	24.57	12.35
SCED-2124	88.30	44.42
RPD-118	2.80	1.23
RPD-354	22.73	9.95
RPD-1062	286.32	134.49
RPD-2124	1287.90	640.51

5.3.4 Steepest-Edge Pricing

Pricing strategies determine the way of how to select a nonbasic variable into basis, and are most likely to impact the simplex performance. Table 5.7 summarizes the iteration counts of the primal simplex using three different pricing techniques: Reduced-cost, Devex and Steepest-edge. From the results shown in Table 5.7, one obvious observation is that, for all test cases, the Steepest-Edge requires the least number of iterations to converge, while Reduced-Cost needs the most iterations. In the cases of SCED problems, such savings are around 20% $\sim$ 35%, and for the RPD problems are 35% $\sim$ 45%. These results indicate that the Steepest-Edge provides faster convergence than the other two pricing techniques. This is due to the fact that geometrically the Reduced-Cost chooses an edge that is "downhill", i.e., along which the objective function decreases, while the Steepest-Edge selects the edge that is "most downhill", i.e., steepest with respect to the objective function. As the Devex is a variant of the latter using approximations to "most downhill", its iterations are between the other two pricings.

With regard to computational time shown in Table 5.8, however, one can see that for small or easy problems, such as RPD-118, RPD-354, and all SCED problems, there is almost no difference among these three pricings due to the lower number of iterations. For the large and hard problems that takes over thousands of iterations to solve, both Steepest-Edge type pricings outperform the traditional Reduced-Cost pricing. In this case, the Devex requires the least computational time while the Reduced-Cost needs the most. Time savings of the Devex are around 15%, as compared to the latter.

The above results show that the Steepest-Edge is not necessarily the best in terms of CPU time although its iteration counts are the least for all cases. On the other hand, the Reduced-Cost seems not so bad despite its higher iteration count. This is because the former is the most costly in computation while the latter is the least expensive. Since for small or easy problems less number of iterations is required to get a solution, the overhead per iteration incurred by the Steepest-Edge eliminates its savings in reducing the iteration count. However, for large and hard problems, as the iteration number is extremely high, the overall work due to lower iteration count of Steepest-Edge type pricings is less than that of the higher iteration count but cheaper computation of the Reduced-Cost. Moreover, since the Devex has the good features of the Steepest-Edge but substantially reduces the computational intensity, it produces the overall best results for all tested problems.

Table 5.7: Effect of pricing techniques on iterations

Cases	Pricing Strategies		
	Reduced Cost	Devex	Steepest Edge
SCED-118	16	16	13
SCED-354	86	72	61
SCED-1062	291	226	188
SCED-2124	502	443	356
RPD-118	111	75	73
RPD-354	392	311	257
RPD-1062	1562	1124	892
RPD-2124	2756	2102	1616

Table 5.8: Effect of pricing techniques on solution time (seconds)

Cases	Pricing Strategies		
	Reduced Cost	Devex	Steepest Edge
SCED-118	0.13	0.12	0.13
SCED-354	0.88	0.95	1.02
SCED-1062	11.57	10.83	11.72
SCED-2124	43.50	45.33	48.99
RPD-118	1.15	1.07	1.22
RPD-354	10.68	10.83	12.18
RPD-1062	177.72	150.35	172.68
RPD-2124	795.38	669.35	747.13

5.4 Comparison with Interior Point Algorithm

In the last decade, the interior point method has become a viable alternative to the simplex method due to its computational efficiency. However, unlike simplex methods, interior point methods can not produce an optimal basic solution, i.e., a basic solution which is both primal and dual optimal. Thus, certain information that is useful for post-optimality analysis is not available. Moreover, interior point methods do not have warm start capability, a very attractive feature of simplex methods. Therefore, it is believed that a good solver should combine the advantages of both methods [76]. CPLEX provides not only the state-of-the-art simplex but an advanced interior point method — a predictor-corrector primal-dual logarithmic-barrier algorithm (Barrier) [60]. In addition, an efficient "barrier-simplex crossover" [62] is implemented to recover bases from (non-basic) solutions of the barrier method, and to switch to the simplex method by warm start. Table 5.9 compares the number of iterations and computational time obtained by using the primal simplex with warm start as well as the barrier method with the crossover.

In general, the results are in favor of the barrier with crossover except for the small cases (such as SCED-118, SCED-354, and RPD-118) where the simplex algorithm is faster than the barrier with crossover in terms of CPU time. For the large SCED problems tested, both algorithms perform equally well; however, for large-scale RPD problems like RPD-1062 and RPD-2124, the barrier with crossover outperforms the simplex method in both CPU time and total iterations. The former requires only 30% $\sim$ 40% computational time of the latter. The main reason is that the barrier with crossover algorithm takes advantage of barrier's fast convergence speed, an efficient basis recovery procedure, and simplex's warm start capabilities. Note that in this study all problems are solved to 10^{-6} accuracy with six significant

Table 5.9: Comparison of simplex with interior point algorithm

Cases	Simplex (warm start)		Barrier with Crossover					
	Iterations	Time	Iterations			Time (seconds)		
		(seconds)	barrier	simplex	total	barrier	crossover	total
SCED-118	12	0.12	9	0	9	0.47	0.10	0.57
SCED-354	54	0.97	11	0	11	1.70	0.32	2.03
SCED-1062	189	12.35	16	0	16	10.23	2.30	12.58
SCED-2124	307	44.42	23	2	25	31.57	10.52	42.20
RPD-118	102	1.23	11	0	11	1.58	0.50	2.10
RPD-354	309	9.95	14	0	14	6.87	1.77	8.68
RPD-1062	1033	134.49	17	0	17	41.85	10.62	52.63
RPD-2124	2207	640.51	22	11	33	120.13	70.63	191.08

digits. Table 5.9 also includes other detailed results for barrier with crossover, such as the crossover time, and the iterations and time of barrier and simplex.

5.5 Summary

Numerical experience of using advanced simplex features is presented for the solutions of large-scale optimal power flow problems. Some newest advances in operation research as well as in sparse matrix techniques are investigated to evaluate their impact on the performance of simplex methods for power engineering problems. The numerical tests are conducted on power systems whose sizes range from hundred to thousands of buses. Based on our extensive study, several conclusions can be drawn as follows:

- The preprocessings, including presolver and aggregator, can reduce the problem size and may save the solution time for large-scale OPF problems. Their effectiveness varies with the size and nature of the problems under concern.

- The scaling can improve numerical stability and sometimes boost simplex performance significantly. It helps to solve OPF problems, especially for those problems with poor conditions.
- Although the crashing can produce an efficient initial basis (cold start), its performance is still uncompetitive as compared to using the advanced basis (warm start). Therefore, the use of crashing should be avoided unless absolutely necessary.
- Using advanced bases (warm start) can reduce at least half solution time as compared to cold start. Therefore, whenever solving same or similar problems, warm start should always be considered.
- Among three pricing techniques, the Devex pricing—a variant of steepest-edge — produces the best results for all test cases, especially for large and difficult problems that need many iterations to reach feasibility and optimality.
- Although the above options are examined on individual basis, the combination of those better parameter settings usually produce the overall best performance.
- The barrier with crossover, that combines the advantages of both simplex and interior point methods, outperforms the simplex method, and may be the best choice as far as solution speed and information completeness are concerned.

Chapter 6

Creation of Network Data for Testing Algorithms

6.1 Introduction

Power system operation and planning relies greatly on computer simulation programs such as load flow, contingency analysis, state estimation, optimal power flow, etc. With the expansion of power networks, many new power system analysis algorithms have been developed to solve problems with ever increasing size and complexity. To evaluate the performance and robustness of the new algorithms, extensive numerical tests should be carried out on a large set of power networks of various types and sizes.

In practice, however, it is not easy (or not possible at all in most cases) to obtain real network data, especially for very large-scale systems. These difficulties arise from either technical or security reasons. Numerical testing, therefore, is often restricted to the relatively small IEEE test networks or to a limited set of special power networks whose data are not available to the general research community.

To alleviate the difficulty of collecting realistic data, in reference [32] an algorithm is developed to synthetically generate power networks of arbitrary size and complexity. The network is created from scratch, and network data are chosen from a predetermined range. However, as noted by the authors of [32], such generated power networks may face convergence problem during load flow runs. Elaborate adjustments of the system state and control variables are required, based on a trial and error method. For creating a very large power system with thousands of buses, the above procedure may take substantial computation time to obtain a load flow solution. Therefore, in order to avoid this problem it is necessary to seek more efficient approachs to creating power networks.

This chapter presents an efficient technique that adopts a different way to create realistic network data. Instead of starting from scratch, the technique uses any available small power system, such as IEEE 118-bus system, to construct a large power network. The created network can be of arbitrary size, different topology and sparsity. Its network data are obtained directly from the small system with only minor modifications. By employing the load flow information of the small power system, the created large system has no converge difficulties when solving a load flow problem. In this case, elaborate variable adjustments are not required. Therefore, the proposed technique is very efficient and robust. Our test results show that creating a system of thousands of buses takes only a couple of seconds. This technique has been successfully used in this thesis for evaluating the performance of different optimization methods for optimal power flow problems. Nevertheless, it should also find other applications in power system analysis where testing of algorithms is necessary on large-scale systems.

6.2 Data Specifications in the OPF Problem

Depending on their roles in the problem formulation, the data for OPF purpose can be classified into two categories. The first category contains the basic network data for load flow purpose, including the network topology, the line parameters, and the bus data. The second category covers the data related to optimization process, such as the physical and operating limits on various system components and the coefficients of cost functions. To formulate an OPF problem, all the above data need to be provided.

Network topology is a graphic representation of a power network with each node standing for a generator/load bus and with each line representing a transmission line/transformer. Its structure is defined by the way how those buses are interconnected through transmission lines or transformers. Network topology is of great importance because it has large impact on the network sparsity pattern which, in turn, affects computational work as well as memory requirements.

The line parameters are the data of transmission lines and transformers. They include series resistance and reactance of a transmission line or a transformer, shunt susceptance of the line, and tap ratio of the transformer. The bus data consist of loads, real and reactive power generations located at every bus. In addition, each bus is assigned one of three bus types based on its characteristic: (1) load buses — with given real and reactive loads; (2) generation buses — with given voltages and real power generations; and (3) slack bus — with its voltage and angle fixed. These bus types are used in solving load flow problems.

The physical and operating constraints include lower and upper limits of real and reactive power generations, transformer tap ratios and shunt capacitors, the load ratings of transmission line and transformers, and limits on voltage. The objective function is normally expressed in terms of total production cost or total real power transmission losses. The former can be represented by a quadratic or piecewise linear function of real-power generations, whereas the latter is described as a nonlinear and nonconvex function of voltages and angles. These data are only employed during optimization process.

In summary, the formulation of the optimal power flow problem not only needs the basic network data, consisting of network topology, line parameters, and bus data, but also requires the optimization data, such as component physical/operating limits and objective function coefficients.

6.3 The Network-Data Creating Technique

This section starts describing an efficient technique for generating realistic large-scale power networks. The emphasis is mainly on creating network data for the Optimal Power Flow (OPF) problem. However, the data may also serve to test algorithms related to other types of power system problems.

6.3.1 The General Approach

The main idea behind the proposed technique comes from the fact that a bulk power network is usually formed by a set of local sub-networks interconnected through tie lines. Therefore, a natural approach to create a large-scale power network is to connect existing small networks via transmission lines. To make such a created network more realistic, the building sub-network should be a *true* power system with real network data. The topology of the created system is determined by the way how sub-networks are inter-connected. It has significant impact on the sparsity pattern of the relevant admittance matrix. The sparsity of the network is controlled through the number of tie lines used between any two related sub-networks. Because real power systems are employed as building sub-systems, all network data of the created system can be inherited directly from the real sub-systems whose data are generally available. With the help of sub-system load flow results, the created system can easily get a meaningful load flow solution without many adjustment efforts. So, the proposed technique has the advantages of not only reducing computational work in the network creating stage but during the load flow run as well. The following sections provide a detailed description on the above technique and its implementation.

6.3.2 Network Topology and Sparsity

Graphically, a power network consists of nodes and lines. The network topology defines how nodes are connected to each other through lines (either transmission lines or transformers). However, in the proposed technique, each small network is treated as a "node". Accordingly, topology means the way how a set of small networks are interconnected. Here, two types of power networks are introduced: (1) Block network and (2) Mesh networks.

Figure 6.1 shows a "Block" power network where each circle represent a sub-network and each line represents a tie line that connects two related sub-networks. In a Block network, connections are not balanced in the sense that there are more links in some areas of the system than in the others. On the contrary, in a "Mesh" network the connections are more balanced for the entire system, as demonstrated in Figure 6.2. To see how the network topology affects matrix sparsity pattern, Figure 6.3 displays nonzero elements of the admittance matrix for block and mesh

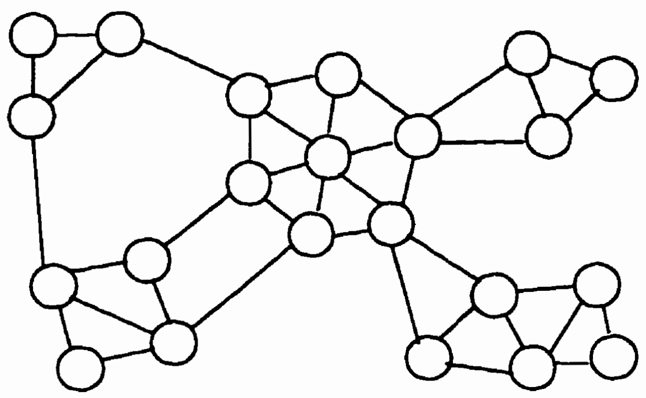

Figure 6.1: A power network with block structure

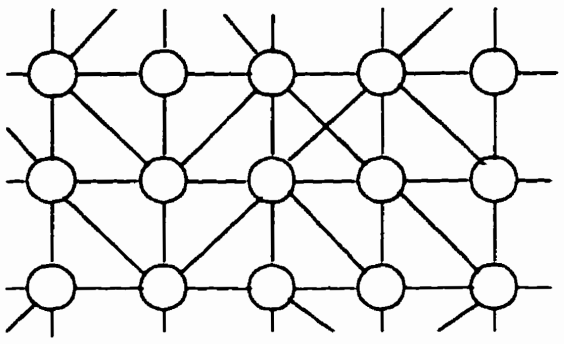

Figure 6.2: A power network with mesh structure

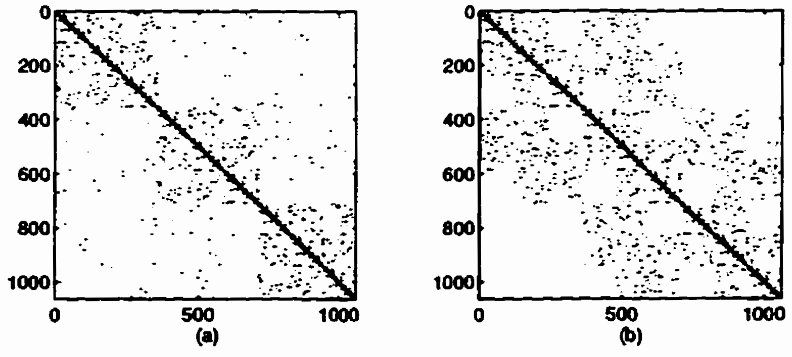

Figure 6.3: Sparsity patterns of admittance matrix: (a) Block and (b) Mesh

networks, respectively. In Figure 6.3 (a), one can observe that the matrix is of a "Block" structure with more elements (connections) inside each block than areas outside blocks. Unlike the "Block" network, the "Mesh" network has an admittance matrix whose elements (connections) are uniformly distributed within a certain diagonal band, as shown in Figure 6.3 (b). The following subsections describe the approach to generate both network types and the way to control their sparsity.

Block Network

For simplicity and efficiency the block network is built in a recursive manner. On every stage of the creation process, the network to be built on the current step is constructed by using the sub-network created in the previous step. To be more specific, let us use the IEEE-118 system as an initial building sub-network to illustrate this procedure. The algorithm starts by connecting several (say 3) IEEE-118 networks with each other to build a large network; then this newly created network is used as a building sub-network to generate a even larger network in the next step. This process keeps going until the number of buses or system size reaches the desired value. In the above procedure, the number of building sub-networks to be used in each step can be adjusted according to system size and sparsity requirements.

Mesh Network

The approach to creating the mesh network is quite different. Given a required system size (the total number of buses), the algorithm first calculates how many building sub-networks are needed based on the size of the sub-network currently in use. From this information, a topology matrix is created with each element representing one sub-network. The dimension of the matrix is then chosen to determine the width of diagonal band for the resulting system (see Figure 6.3 (b)). In the meantime, each topology matrix element is assigned a number which will be used later to name all buses in the related system. Finally, those sub-networks who are neighbors of each other are connected through transmission lines. It should be noted that the configuration among sub-networks can be further defined by specifying whether those diagonal related sub-network should be connected.

Network Sparsity

The network sparsity refers to the sparsity of a network admittance matrix, which is largely influenced by the total number of lines and transformers in a system. Although the network topology can affect the sparsity pattern of the matrix, the most effective way to control its sparsity is to increase or decrease transmission lines and/or transformers used in a system. In the proposed technique, this objective can be achieved by controlling the number of tie lines to be used between any two related sub-networks. To make these tie-lines more realistic, their connecting locations in the sub-networks are selected through a random procedure.

6.3.3 Network Data Creation

As mentioned earlier, the large-scale network is created from a set of small real systems whose data are generally available. Therefore, all data of the created network (see in Section 3.2) can be quickly duplicated from those small systems. One does not need to specify any of the data unless he/she wants to make some changes for a specific purpose. By using the load flow results of the small system, the created system can easily converge to a load flow solution without the need to adjust its control variables. As a result, this approach not only saves computational time during the network creation but also when finding its load flow solution.

In summary, the technique proposed herein can create realistic large-scale power networks. It allows the user to choose four important parameters: 1) network topology, 2) network size, 3) network sparsity, and 4) slack bus number. The topology of a network to be created is chosen by running the relevant algorithm. The network size is defined as the number of buses in the system. The algorithms use this parameter to determine how many sub-networks should be used in order to create a network of specified size. The sparsity of the matrix is controlled by giving the number of tie lines used to connect any two sub-networks. Since, originally, each sub-network has its own slack bus, one needs to select one of them as the slack bus of the entire system for load flow purpose (the rest of the slack buses are changed to generation buses). Once these parameters are given, the algorithms will automatically create the desired system without the need of any intervention. Finally, the created network data is exported to an ASCII file with a specified format (see [103]).

6.4 Summary

An efficient technique is presented and implemented to create realistic network data for power system analysis. The developed algorithm can generate large-scale power networks with the following features:

- The program can create networks with different network dimension, topology, and sparsity.
- The program can use any existing real power system as a building sub-network to make the creation process more productive.

- The network data can be inherited directly from real sub-network data without the need of many adjustments.
- All sub-networks are randomly connected and real network data are employed to make the created system more realistic.
- The created network can easily get a meaningful load flow solution with no convergence difficulty.
- There is no limit on the size of the network that can be generated.

Our numerical experiments have verified that this algorithm is fast and robust. It has been used in this thesis research, and has proved to be a very useful tool for testing power system programs.

Chapter 7

Conclusions

The main objective of this thesis has been to research and develop the advanced interior point methods for the efficient solution of optimal power flow problems. Detailed study has been conducted on the real and reactive power dispatch problems, i.e., the security-constrained economic dispatch and the minimum transmission active-power loss reactive power dispatch. The successive linear programming has been applied to the underlying nonlinear problems, and the resulting linear sub-problems are solved by infeasible primal-dual interior point methods. The research on the infeasible primal-dual algorithms has been oriented to explore their full potential for power engineering problems. Intensive study has focused on all issues that influence the performance of interior point algorithms as well as successive linearization procedure. The use of sparse linear formulation and techniques has been investigated to improve the computational efficiency of the algorithms.

The linear real power dispatch problem has been formulated based on a decoupled load flow model to improve solution efficiency. The resulting linear formulation involves the variables of only real power generations and phase angles. The minimization of total production cost has been employed as the objective function of the problem. The security constraints on branch flow have been considered to limit the real power flow on transmission lines. On the other hand, the linear reactive power dispatch problem has been formulated in terms of all reactive controls and state variables, where a full load flow model has been used to improve the solution accuracy. The total real-power system losses has been chosen as the objective function to be minimized during optimization process. Also, the limits on branch power flow have been considered to satisfy security constraints.

The most elegant interior point methods known so far are the primal-dual path following algorithms. They enjoy not only the best theoretical complexity but also prove computationally very efficient. Two advanced variants in this class of IPMs have been studied in detail, namely, the infeasible primal-dual algorithm and the predictor-corrector primal-dual algorithm. The major advantage of these algorithms is that an initial feasible point is not required to start the algorithm. The feasibility of solutions is attained during the process as optimality is approached. Both algorithms share the common feature of approximately following the central path of the feasible regions except that the former uses the first-order while the latter uses the second-order information of the primal-dual trajectory. Therefore, they can take large step along search direction to achieve fast objective reduction. The algorithms have been extended to incorporate lower and upper bounds for special needs in our particular application.

The detailed study of the primal-dual algorithm has indicated that the choices of Newton step sizes, initial point, and barrier parameter have large influence on its performance. The size of Newton step determines how much reduction in the objective function can be made in each iteration. A conservative step size may restrict the progress toward optimality once feasibility of the solution is attained.

The strategy of aggressively increasing the step size based on feasibility condition can avoid the above problem and speed up convergence. An initial point with large magnitude usually causes a large initial duality gap which needs more iteration efforts to reduce. The refined start point with small magnitudes can help convergence for problems with small optimal solutions. The barrier parameter should be decreased as iterations progress. However, its over-reduction in the early stage will cause negative duality gap and slow convergence. In this circumstance, properly boosting the barrier parameter can avoid repeated occurring of such phenomenon and hence smooth optimization process.

The investigation of predictor-corrector primal-dual algorithm has focused on those issues that are critical to its efficient implementation, such as the adjustment of barrier parameter, the determination of initial point, and the use of multiple corrector steps. Some heuristics have been proposed to customize the algorithm parameters to our particular application, including (1) an improved barrier parameter adjusting scheme based on feasibility criterion, and (2) a refined initial point procedure using small and balanced primal and dual thresholds. Test results have indicated that the proposed ideas significantly improve the algorithm performance, reducing over half iterations and solution time. Also, it has been found that using multiple correctors generally requires less iterations, but its overall performance is not as efficient and stable as using one corrector step, especially for ill-conditioning problems. The comparison with the pure primal-dual algorithm has been conducted, which reconfirms the superiority of the predictor-corrector method.

The practical issues related to successive linear programming have been thoroughly investigated. The influence of linear step sizes and LP/OPF stopping criteria has been evaluated on both real and reactive power dispatch problems. The in-depth analysis on these issues has found that the proper adjustments of linear steps and tolerances are crucial for achieving fast solution speed while maintaining solution accuracy. The strategy of applying a large initial step plus adaptive step reduction improves the convergence behavior of successive linear processes (SLP), significantly reducing the SLP iterations. In addition, the idea of employing large initial LP tolerance and then gradually decreasing its value as linear points approach the optimum can save computational work in solving each linear sub-problem, dramatically reducing the total LP iterations, even improving SLP performance.

The computational bottleneck of the interior point algorithm is to solve a large-scale system of linear equations at every iteration. Therefore, solution speed and numerical stability are two concerns for developing fast and robust interior point algorithms. In our implementation, the normal equation method has been selected as the solution method due to its good numerical characteristics. Sparse techniques have been applied to every stage in the solution of normal equations. Test results on large-scale problems have verified the computational efficiency and reliability of our developed interior point algorithms.

As part of this research, the recent developments in the simplex technology have been investigated to evaluate their impact on power engineering problems. A state-of-art simplex code has been used to solve the large-scale real and reactive power dispatch problems. Extensive numerical tests have been conducted on such advanced features as preprocessing, scaling, crashing, advanced base starting, and steepest-edge pricings. Our experience has shown that these advances do improve simplex performance significantly. Their influence varies widely, depending on the size and nature of the problem under study. Also, test results have shown that combining the advantages of both simplex and interior point methods may be the best choice as far as solution speed and information completeness are concerned.

Finally, an efficient network creating technique has been developed for testing algorithms. The program can generate realistic network data based on any available real system. The created network can be of different size, topology, and sparsity.

7.1 Recommendations for Future Research

The research work presented in this thesis has demonstrated the successful application of interior point algorithms for optimal power flow problems. It also provides the possibility of continued research in the following directions:

1. It has been observed that using multiple correctors can save iterations required by the primal-dual algorithm, provided that the OPF problems have good conditioning. Although such savings has little impact on solution time for small size problems, it may bring significant benefit for large-scale problems where the numerical factorization is much more expensive than the solution phase. More detailed investigation is necessary to explore its potential application.

2. An attractive feature of Simplex method is its warm start capability, which is very useful when solving similar problems repeatedly, such as the successive linear solution of OPF problems. So far, there have been little progress in this area of interior point algorithms, especially for power engineering problems. Further study should be directed toward exploring the possibility of improving the algorithm performance through warm start.

3. In the circumstance where high solution accuracy is required, extending the algorithms to nonlinear programming should be considered. In this case, however, the normal equation method is not suitable for solving the search

direction. Other methods such as argumented equation methods should be investigated. Without lower and upper bounds on most state variables, special techniques of treating those free variables may be required to overcome possible numerical difficulties. Other issues such as customizing barrier parameter and initial points for nonlinear problems also need to be investigated.

Bibliography

[1] K.H. Abdul-Rahman and S.M. Shahidehpour, "A fuzzy-based optimal reactive power control", IEEE Trans. on Power Systems, Vol. 8, No. 2 662-670, May 1993.

[2] I. Adler, N. Karmarkar, M.G.C. Resende and G. Veiga, "Data structures and programming techniques for the implementation of Karmarkar's algorithm", ORSA Journal of Computing, Vol. 106, 1989, pp. 1-84.

[3] I. Adler, N. Karmarkar, M.G.C. Resende and G. Veiga,"An implementation of Karmarkar's algorithm for linear programming", Mathematical Programming 44, 1989, pp. 297-335.

[4] O. Alsac, J. Bright, M. Prais, and B. Stott, "Further developments in LP-based optimal power flow", IEEE Trans. on Power Systems, Vol. 5, No. 3, August 1990, pp. 697-706.

[5] F. L. Alvarado, "State estimation using augmented block matrices", IEEE Trans. on Power Systems, Vol. PWRS-5, 1990, pp. 911-921.

[6] E.D. Andersen, J. Gondzio, C. Mészáros, and X. Xu, "Implementation of interior point methods for large scale linear programming", Technical Report 96.3,

Logilab, Section of Management Studies, University of Geneva, Switzerland, January 24, 1996.

[7] K.M. Anstreicher and P. Watteyne, "A family of search directions for Karmarkar's algorithm", Operations Research 43, 1993, pp. 759-767.

[8] K.M. Anstreicher, "A combined Phase I-Phase II scaled potential algorithm for linear programming", Mathematical Programming 52, 1991, pp. 424-439.

[9] K.M. Anstreicher, "A combined phase I - phase II projective algorithm for linear programming", Mathematical Programming 43, 1989, pp. 209-223.

[10] E.R. Barnes, "A variation on Karmarkar's algorithm for solving linear programming problems", Mathematical Programming 36, 1986, pp. 174-182.

[11] D.A. Bayer and J.C. Lagarais, "The nonlinear geometry of linear programming, Part I: Affine and projective scaling trajectories; Part II: Legendre Transform Coordinates; Part III: Central trajectories; Part IV: Karmarkar's linear programming algorithm and Newton's method", Trans. Amer. Math. Soc., 314, 1989, pp. 499-581.

[12] R.E. Bixby, "Implementing the simplex method: the initial basis", ORSA J. Comput. Vol. 4, No. 3, 1992, pp. 267-284.

[13] R.E. Bixby, "Progress in linear programming", ORSA J. Comput. Vol. 6, No. 1, 1994, pp. 15-22.

[14] J. Carpenter, "Contribution to the economic dispatch problem" (in French), Bull. Soc. France. Elec., Vol. 8, August 1962, pp. 431-447.

[15] J. Carpenter, "Towards a secure and optimal automatic operation of power systems", in PICA Proc., 1987, pp. 2-37.

[16] B. H. Chowdhury and Salfur Rahman, "A review of recent advances in economic dispatch", IEEE Trans. on Power Systems, Vol. 5, No. 4, November 1990, pp. 1248-1257.

[17] K.A. Clements, P.W. Davis, and K.D. Frey, "An interior point algorithm for weighted least absolute value power system state estimation", IEEE Winter Power Meeting, Paper. 91-WM 235-2 PWRS, New York, February 1991.

[18] G.C. Contaxis, C. Delkis and G. Korres, "Decoupled optimal load flow using linear or quadratic programming", IEEE Trans. on Power Systems, Vol. PWRS-1, No. 2, May 1986, pp. 1-7.

[19] CPLEX Optimization, Inc. *Using the $CPLEX^{TM}$ Callable Library — including Using the $CPLEX^{TM}$ Linear Optimizer with $CPLEX^{TM}$ Barrier and Mixed Integer Solvers*, Incline Village, Nevada, 1994.

[20] J. Czyzyk, S. Mehrotra, and S.J. Wright, "PCx User Guide", Technical Report OTC 96/01, Optimization Technology Center, May 15, 1996.

[21] George B. Dantzig, *Linear programming and extensions*, Princeton University Press, Princeton, New Jersey, 1963.

[22] J.K. Delson and S.M. Shahidehpour, "Linear programming applications to power system economics, planning and operations", IEEE Trans. on Power Systems, Vol. 7, No. 3, August 1992, pp. 1155-1162.

[23] I.I. Dikin, "Iterative solution of problems of linear and quadratic programming", Society Mathematics Doklady 8, 1967, pp. 674-675.

[24] H.W. Dommel, and W.F. Tinney, "Optimal power flow solutions", IEEE Trans. on Power Apparatus and Systems, Vol. PAS-87, No. 10, October 1968, pp. 1866-1876.

[25] I.S. Duff, A.M. Erisman, and J.K. Reid,"A Comparison of Some Methods for the Solution of Sparse Overdetermined Systems of Linear Equations", Journal Inst. Maths Applics., Vol. 17, 1976, pp. 267-280.

[26] S.C. Eisenstat, M.H. Schultz and A.H. Sherman, "Algorithms and data structures for sparse symmetric Gaussian elimination", SIAM J. SCI. STAT. COMPUT., Vol. 2, No. 2, June 1981, pp. 225-237.

[27] G.C. Ejebe, W.R. Puntel and B.F. Wollenberg, "A load curtailment algorithm for the evaluation of power transmission system adequacy", IEEE PES Summer Meeting, Mexico City, July, 1977, Paper A 77 505-1.

[28] A.V. Fiacco and G.P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley & Sons, New York, 1968.

[29] J.J.H. Forrest and D. Goldfarb, "Steepest edge simplex algorithms for linear programming", Mathematical programming 57, 1992, pp. 341-374.

[30] R. Freund, "Polynominal-time algorithms for linear programming based only on primal scaling and projected gradients of a potential function", Mathematical Programming 51, 1991, pp. 203-222.

[31] K.R. Frisch, "The logarithmic potential method of convex programming", Technical Report, University Institute of Economics, Oslo, Norway, 1955.

[32] F.D. Galiana, H. Javidi and S. Mcfee, "Application of a preconditioned conjugate gradient algorithm to power network analysis", PICA, 1993, and IEEE Trans. on Power Systems, May 1994.

[33] D. Gay, "Electronic mail distribution of linear programming test problems", Mathematical Programming Society COAL Newsletter.

[34] D. Gay, "A variant of Karmarkar's linear programming algorithm for problems in standard form", Mathematical Programming 52, 1991, pp. 441-446.

[35] D. Goldfarb and J. Reid, "A practical steepest-edge simplex algorithm", Mathematical programming 12, 1977, pp. 361-371.

[36] G.H. Golub and C.F. Van Loan, *Matrix Computations*, John Hopkins University Press, Baltimore 1989.

[37] J. Gondzio and M. Makowski, "HOPDM modular solver for LP problems", User's Guide to version 2.12, WP-95-50, International Institute for Applied Systems Analysis (IIASA), A-2361 Laxenburg, Austria, June 1995.

[38] J. Gondzio, "Multiple centrality corrections in a primal-dual method for linear programming", Technical Report 94.20, Logilab, Section of Management Studies, University of Geneva, Switzerland, May 6, 1995.

[39] J. Gondzio, "Presolve analysis of linear programs prior to applying an interior point method", Technical Report 94.3, Logilab, Section of Management Studies, University of Geneva, Switzerland, December 20, 1994.

[40] J. Gondzio, "Splitting dense columns of constraint matrix in interior point methods for large scale linear programming", Optimization, Vol. 24, 1992, pp. 285-297.

[41] J. Gondzio, "Warm start of the primal-dual method applied in the cutting plane scheme", Technical Report 96.3, Logilab, Section of Management Studies, University of Geneva, Switzerland, May 19, 1996.

[42] C.C. Gonzaga, "An algorithm for solving linear programming in $O(n^3L)$ operations", Technical Report UCB/ERL M87/10, Electronic Research Laboratory, University of California, Berkeley, CA 94720, 1987.

[43] C. Gonzaga, "Large step path-following methods for linear programming, Part II: Potential reduction methods", SIAM J. Optimization 1, 1991, pp. 280-292.

[44] C.C. Gonzaga, "Path following methods for linear programming", SIAM Review 34:2, 1992, pp. 167-227.

[45] Sergio Granville, "Optimal reactive dispatch through interior point methods", IEEE Trans. on Power Systems, Vol. 9, No. 1, February 1994, pp. 136-142.

[46] Charles A. Gross, *Power System Analysis*, Second Edition, John Wiley & Sons, Inc., New York, 1986.

[47] H. H. Happ, "Optimal power dispatch — a comprehensive survey", IEEE Trans. on Power Apparatus and Systems", Vol. PAS-96, No. 3, May/June 1977, pp. 841-850.

[48] P.M.J. Harris, "Pivot selection methods of the Devex LP code", Mathematical Programming 5, 1973, pp. 1-28.

[49] E.C. Housos and G.D. Irisarri, "A sparse variable metric optimization method applied to the solution of power system problems", IEEE Trans. on Power Apparatus and System, Vol. PAS-101, No.1, 1982, pp. 195-202.

[50] M. Huneault and F.D. Galiana, "An investigation of the solution to the optimal power flow problem incorporating continuation methods", IEEE Trans. on Power Systems, Vol. 5, No. 1, February 1990, pp. 103-110.

[51] N. Karmarkar, "A new polynominal-time algorithm for linear programming", Combinatorica, Vol. 4, 1984, pp. 373-395.

[52] N. Karmarkar, J.C. Lagarias, L. Slutsman, and P. Wang, "Power series variants of Karmarkar-type algorithm", At&T Tech. J., May/June 1989, pp. 20-36.

[53] L.G. Khachiyan, "A polynominal algorithm in linear programming", Society Mathematics Doklady 20, 1979, pp. 191-194.

[54] V. Klee and G.J. Minty, O.Shiske (ed.), *How Good Is the Simplex Algorithm, in the Inequality III*, Academic Press, New York, 1972.

[55] M. Kojima, S. Mizuno and A. Yoshise, "A primal-dual interior point algorithm for linear programming", Progress in Mathematical Programming 29-47, N. Megiddo, ed., Springer-Verlag, New York, 1989.

[56] C.N. Lu, M.R. Unum, "Network constrained security control using an interior point method", IEEE Trans. on Power Systems, Vol. 8, No. 3, August 1993, pp. 1068-1076.

[57] I.J. Lustig, R.E. Marsten, D.F. Shanno, "Computational experience with a primal-dual interior point method for linear programming", Linear Algebra and Its Applications, 152, 1991, pp. 191-222.

[58] I.J. Lustig, R.E. Marsten, D.F. Shanno, "Computational experience with a globally convergent primal-dual predictor-corrector algorithm for linear programming", Mathematical Programming 66, 1994, pp. 123-135.

[59] I.J. Lustig, R.E. Marsten, D.F. Shanno, "Interior point methods for linear programming: computational state of the art", ORSA J. Comput. Vol. 6, No. 1, 1994, pp. 1-14.

[60] I.J. Lustig, R.E. Marsten, D.F. Shanno, "On implementing Mehrotra's predictor-corrector interior point method for linear programming", SIAM J. Optimization, Vol. 2, No. 3, 1992, pp. 435-449.

[61] K.A. McShane, C.L. Monma and D.F. Shanno, "An implementation of a primal-dual interior point method for linear programming", ORSA Journal on Computing Vol. 1, 1989, pp. 70-83.

[62] N. Megiddo, "On finding primal- and dual-optimal bases", ORSA J. Comput. Vol. 3, No. 1, 1991, pp. 63-65.

[63] N. Megiddo and M. Shub, "Boundary behavior of interior point algorithms in linear programming", Technical Report RJ 5319, IBM Thomas J. Watson Research Center, Yorktown Heights, NY. 1986.

[64] N. Megiddo, "Pathways to the optimal set in linear programming", in Progress in Mathematical Programming, Interior-Point and Related Methods, 131-158, N. Megiddo, ed., Springer-Verlag, New York, 1989.

[65] S. Mehrotra, "On the implementation of a primal-dual interior point method", SIAM Journal on Optimization 2, 1992, pp. 575-601.

[66] V. Miranda and J.T. Saraiva, "Fuzzy modeling of power system optimal load flow", IEEE Trans. on Power systems, Vol. 7, No. 2, May 1992, pp. 843-849.

[67] S. Mizuno, M.J. Todd and Y. Ye, "On adaptive step primal-dual interior point algorithms for linear programming", Technical Report 944, School of OR and IE, Cornell University, Ithaca, NY. 1990.

[68] S. Mizuno, "Polynominality of the Kojima-Megiddo-Mizuno infeasible interior point algorithm for linear programming", Technical Report 1006, School of OR and IE, Cornell University, Ithaca, NY. 1992.

[69] R.C. Monteiro and I. Adler, "Interior path-following primal-dual algorithms, Part I: Linear programming", Mathematical Programming 44, 1989, pp. 27-41.

[70] R.C. Monteiro, I. Adler, and M.G.C. Resende, "A polynominal-time primal-dual affine scaling algorithm for linear and convex quadratic programming and its power series extension", Math. Oper. Res. 15, 1990, pp. 191-214.

[71] Jorge J. Moré and Stephen J. Wright, *Optimization Software Guide*, SIAM, Philadelphia, 1993, pp. 76-77.

[72] W.Y. Ng, "Generalized generation distribution factors for power system security evaluation", IEEE Trans. on Power Apparatus and Systems, Vol. PAS-100, No.3, 1981, pp. 1001-1005.

[73] K. Ponnambalam, V.H. Quintana, A. Vannelli, "A fast algorithm for power system optimization problems using an interior point method", IEEE Trans. on Power Systems, Vol. 7, No. 2, May 1992, pp. 892-899.

[74] V.H. Quintana and M. Santos-Nieto, "Reactive power dispatch by successive quadratic programming", IEEE Trans. on Energy Conversion, Vol. EC-4, No. 3, September 1989, pp. 425-435.

[75] A.M. Sasson, H.M. Merrill, "Some applications of optimization techniques to power systems problems", Proceedings of the IEEE, Vol. 62, No. 7, July 1974, pp. 959-972.

[76] D.F. Shanno, "Computational methods for linear programming", E. Spedicato (*ed.*)*Algorithms for Continuous Optimization*, 1994, pp. 383-413.

[77] R.R. Shoults and D.T. Sun, "Optimal power flow based upon P-Q decomposition", IEEE Trans. on Power Apparatus and Systems, Vol. PAS-101, No. 2, 1982, pp. 397-405.

[78] H. Singh and F.L. Alvarado, "Weighted least absolute value state estimation using interior point methods", IEEE Trans. on Power Systems, Vol. 9, No. 3, August 1994, pp. 1478-1484.

[79] B. Stott and J. L. Marinho, "Linear programming for power-system network security applications", IEEE Trans. on Power Apparatus and Systems, Vol. PAS-98, No. 3, May/June 1979, pp. 837-844.

[80] B. Stott and E. Hobson, "Power system security control calculations using linear programming, Part I", IEEE Trans. on Power Apparatus and Systems, Vol. PAS-97, No. 5, Sept/Oct 1978, pp. 1713-1720.

[81] B. Stott and E. Hobson, "Power system security control calculations using linear programming, Part II", IEEE Trans. on Power Apparatus and Systems, Vol. PAS-97, No. 5, Sept/Oct 1978, pp. 1721-1729.

[82] B. Stott, J.L. Marinho and O. Alsac, "Review of linear programming applied power system rescheduling", Proc. of 1979 PICA Conference, Minnesota, MN, 1979, pp. 142-154.

[83] B. Stott, "Review of load-flow calculation methods", Proceedings of The IEEE, Vol. 62, No. 7, July 1974, pp. 916-929.

[84] D.I. Sun, and B. Ashley, B. Brewer, A. Hughes and W.F. Tinney, "Optimal power flow by Newton approach", IEEE Trans. on Power Apparatus and Systems, Vol. PAS-103, No. 10, October 1984, pp. 1248-1259.

[85] W. F. Tinney, J. M. Bright, K. D. Demaree, and B. A. Hughes, "Some deficiencies in optimal power flow", IEEE Trans. on Power Systems, Vol. 3, No. 2, May 1988, pp. 676-681.

[86] Michael J. Todd, "The affine-scaling direction for linear programming is a limit of projective-scaling directions", Linear Algebra and Its Applications, Vol. 152, 1991, pp. 93-105.

[87] M.J. Todd and Y. Ye, "A centered projective algorithm for linear programming", Math. Oper. Re. 15, 1990, pp. 508-529.

[88] M.J. Todd and B.P. Burrell, "An extension of Karmarkar's algorithm for linear programming using dual variables", Algorithmica 1, 1986, pp. 409-424.

[89] R.J. Vanderbei, "ALPO: Another linear program optimizer", ORSA Journal on Computing, Vol. 5, No. 2, 1993, pp. 134-146.

[90] R.J. Vanderbei, "LOQO: an interior point code for quadratic programming", Program in Statistics and Operations Research, Princeton University, Princeton, NJ 08544.

[91] R.J. Vanderbei, M.S. Meketon and B.A. Freedman, "A modification of Karmarkar's linear programming algorithm", Algorithmica 1, 1986, pp. 395-407.

[92] R.J. Vanderbei, "Splitting dense columns in sparse linear systems", Linear Algebra and its Applications 152, 1991, pp. 107-117.

[93] Anthony Vannelli, "An adaptation of the interior point method for solving the global routine problem", IEEE Trans. on Computer-Aided Design, Vol. 10, No. 2, Feb. 1991, pp. 193-203.

[94] Luis S. Vargas, V.H. Quintana and A. Vannelli, "A tutorial description of an interior point method and its applications to security-constrained economic dispatch", IEEE Trans. on Power System, Vol. 8, No. 3, August 1993, pp. 1315-1323.

[95] H. Wei, H. Sasaki, and R. Yokoyama, "An application of interior point quadratic programming algorithm to power system optimization problems", IEEE Trans. on Power Systems, Vol. 11, No. 1, February 1996, pp. 260-266.

[96] Allen J. Wood and Bruce F. Wollenberg, *Power Generation, Operation and Control*, John Wiley & Sons, New York, 1984.

[97] Y.C. Wu, A.S. Debs, and R.E. Marsten, "A direct nonlinear predictor-corrector primal-dual interior point algorithm for optimal power flows", IEEE Trans. of Power Systems, Vol. 9, No. 2, May 1994, pp. 876-883.

[98] X. Yan and V.H. Quintana, "An efficient predictor-corrector interior point algorithm for security-constrained economic dispatch", 96 SM 506-6 PWRS, IEEE/PES Summer Meeting, Denver, Colorado, July 1996.

[99] X. Yan and V.H. Quintana, "Improving an interior-point-based OPF by dynamic adjustments of step sizes and tolerances", accepted for publication in IEEE Trans. on Power Systems, December, 1996.

[100] X. Yan and V.H. Quintana, "An infeasible interior point algorithm for optimal power flow problems", Electric Power Systems Research Journal, Volume 40, No. 3, 1997.

[101] X. Yan and V.H. Quintana, "An efficient technique for solving dense row problems in security-constrained economic dispatch", Proceedings of the 26th North American Power Symposium (NAPS), Manhattan, Kansas, September 1994.

[102] X. Yan, V.H. Quintana, and Antonio Gómez-Expósito, "Numerical experience using advanced features of simplex codes for power system optimization problems", 5th Hispano-Lusas Conference on Electrical Engineering, Salamanca, Spain, July 1997.

[103] X. Yan and V.H. Quintana, "An efficient technique to create realistic large-scale power networks for power system analysis", Technical Report UW-E&CE #95-09, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada, July, 1995.

[104] Y. Ye, "An $O(n^3L)$ potential reduction algorithm for linear programming", Mathematical Programming 50, 1991, pp. 239-258.

[105] Y. Zhang, "On the convergence of an infeasible interior point algorithm for linear programming and other problems", Department of Mathematics and Statistics, University of Maryland, Baltimore Country, 1992.

[106] Y. Zhang, "User's Guide to LIPSOL — Linear-programming interior point solvers V0.3", Department of Mathematics and Statistics, University of Maryland, Baltimore County, Baltimore, Maryland 21228-5398, U.S.A., July 1995.