Cape Peninsula
University of Technology

A NOVEL OPTIMISATION METHOD FOR VOLTAGE AND REACTIVE POWER CONTROL OF ELECTRIC POWER SYSTEMS

by

HALTOR MATAIFA

Thesis submitted in fulfilment of the requirements for the degree

Doctor of Engineering: **Electrical Engineering**

in the Faculty of Engineering

at the Cape Peninsula University of Technology

Supervisor: Dr. Senthil. Krishnamurthy

Co-supervisor: Dr. Carl. Kriger

Bellville
15th Feburary 2023

DECLARATION

I, Haltor Mataifa, declare that the contents of this dissertation/thesis represent my own unaided work, and that the dissertation/thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

2023/02/15

Signed **Date**

ABSTRACT

Reliable electrical power supply is one of the most important utilities for modern society. This can be seen by the fact that any prolonged interruption of electrical power supply usually leads to enormous disruption of essential services and normal daily activities, and can in fact threaten to cause a lot of damage or losses if not promptly remedied. Moreover, recent developments in the power system, such as the deregulation and restructuring of the electrical power supply industry, the introduction of competitive electricity and power markets, and the rapid growth and expansion of distributed and decentralized electrical power generation, have led to a significant increase in the complexity of modern power systems, adding to the challenge of operating them reliably and efficiently. Thus, the need for optimal strategies for the secure, economical and efficient operation of the power system is arguably even greater now than at any other time in the history of the power system. In line with this identified need, this thesis investigates the theoretical design, development, and practical implementation of efficient algorithms that contribute to the secure, economical and reliable operation of electric power transmission systems.

The focus of the research presented in this thesis is on the development of methods and algorithms for the solution of the Volt/VAR optimization (VVO) problem, which is a very important sub-problem of the optimal power flow (OPF) problem that is primarily concerned with determination of the optimal coordinated dispatch of voltage-regulating devices and reactive power sources, with voltage profile improvement and system power loss minimization as the main objectives (among others). Volt/VAR optimization is one of the most actively researched areas of power system operation. While most researchers consider either classical or heuristic optimization methods in isolation, the research work presented in this thesis investigates the design of efficient Volt/VAR optimization strategies considering both classical and heuristic optimization techniques.

Two main optimization algorithms are developed for the solution of the Volt/VAR optimization problem in this thesis. One is based on the primal-dual interior-point method (PDIPM), which is one of the most efficient classical methods for large-scale nonlinear optimization. The other is based on the particle swarm optimization algorithm, one of the most popular heuristic optimization techniques. To enhance the efficiency of the developed algorithms, the model development for the Volt/VAR optimization problem considers both the polar and rectangular coordinate representations of the system voltages. Although most researchers make use of the polar representation, analysis reveals that the rectangular representation has relatively more favourable mathematical properties from the computational efficiency perspective, particularly for the methods and algorithms developed in this thesis. The efficiency of the developed methods and algorithms is further enhanced by incorporating the Newton-Raphson load flow computation into the Volt/VAR optimization algorithm, which is moreover

also developed using the rectangular model formulation. Five power system case studies, a 3-bus system, a 6-bus system, and the IEEE 14-bus, IEEE 30-bus, and IEEE 118-bus power systems, are used to analyse the performance of the developed algorithms. The results obtained from the performance analysis reveal that the developed algorithms exhibit high computational efficiency and superior convergence characteristics. Moreover, a comparative performance analysis is also conducted between the PDIPM-based VVO algorithm and the PSO-based VVO algorithm. The performance analysis reveals that the primal-dual interior-point method outperforms the particle swarm optimization algorithm in terms of computational efficiency, since on average it requires fewer iterations to converge, and has a shorter running time. The particle swarm optimization, on the other hand, generally achieves a higher percentage real power loss reduction than the primal-dual interior-point method. This suggests that the two classes of methods (i.e. classical and heuristic optimization methods) have complementary performance characteristics, something which could be exploited to devise optimization strategies that seek to combine their relative strengths, and thus have a better prospect of exhibiting performance that is superior to that of the individual algorithms.

The methods, algorithms and software programs developed and presented in this thesis are of great relevance both to industry and to academia, and can serve as a good foundation for further research and development, as suggested in the concluding chapter of the thesis.

Keywords: *Optimal Power Flow, Volt/VAR Optimization, reactive power/voltage control, classical optimization, primal-dual interior-point method, heuristic optimization techniques, computational intelligence, particle swarm optimization.*

ACKNOWLEDGEMENTS

I wish to thank:

- The Centre for Substation Automation and Energy Management Systems (CSAEMS), for the opportunity extended to me to be a member of the CSAEMS, and to conduct my research under their capable stewardship. It has been a tremendously enriching experience.
- My former supervisor, Prof. Raynitchka Tzoneva, for her immeasurable contribution to my research work. The depth of my indebtedness toward her cannot possibly be conveyed by any set of words.
- My supervisors Dr. Senthil Krishnamurthy and Dr. Carl Kriger for their guidance and mentorship throughout the course of this research
- Dr. Zakhele Nkosi (HoP) and Dr. Marco Adonis (HoD) for their immense support, without which the completion of this research would simply not have been possible
- My family for their patience and support of my studies
- All others not mentioned by name who have in some way rendered support to the work done in connection with this thesis.

The financial assistance of the National Research Foundation towards this research is acknowledged. Opinions expressed in this thesis and the conclusions arrived at, are those of the author, and are not necessarily to be attributed to the National Research Foundation.

DEDICATION

To all my dear family members, whose unconditional love, extraordinary patience and unwavering support have made the most challenging times during my study period bearable; to dear Prof. Raynitchka Tzoneva, the most remarkable person I have ever had the honour of knowing; and to Jehovah God Almighty, the source of my life, of all my strength and of all my capabilities.

TABLE OF CONTENTS

CHAPTER ONE: THESIS INTRODUCTION

CHAPTER TWO: LITERATURE REVIEW ON CLASSICAL AND HEURISTIC METHODS FOR VOLT/VAR OPTIMIZATION

CHAPTER THREE: FORMULATION OF THE VOLT/VAR OPTIMIZATION PROBLEM

CHAPTER FOUR: DESIGN AND IMPLEMENTATION OF THE PRIMAL-DUAL INTERIOR-POINT METHOD AND APPLICATION TO A NONLINEAR PROBLEM

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

GLOSSARY

Terms/Acronyms/Abbreviations	Definition/Explanations
AI	Artificial intelligence
B&B	Branch and Bound; an algorithm design paradigm for discrete and combinatorial optimization problems
CG	Conjugate gradient (optimization method)
DC	Direct current
DE	Differential evolution
DG	Distributed generation
EP	Evolutionary programming
EPRI	Electric power research institute
ES	Expert system
FACTS	Flexible AC Transmission System
FERC	Federal Energy Regulatory Commission
GA	Genetic algorithm
Gbest	The global best position of the particle swarm; that is, the best position achieved by any particle in the entire swarm
GRG	Generalized reduced gradient (optimization method)
HVDC	High Voltage Direct Current
IPM	Interior-point method
KKT	Karush-Kuhn-Tucker; first-order derivative tests used to check the optimality of a solution to an optimization problem
Lbest	The local best position achieved within a given neighbourhood of a particle swarm
LP	Linear programming
LTC	Load tap changer
MATLAB	A technical and numerical computing environment, developed by MathWorks
MINLP	Mixed-integer nonlinear programming
MIP	Mixed integer programming
OPF	Optimal power flow
ORD	Optimal reactive (power) dispatch
ORPD	Optimal reactive power dispatch
Pbest	The personal best position of a particle in the particle swarm optimization algorithm
PC-PDIPM	Predictor-corrector primal-dual interior-point method
PDIPA	Primal-dual interior-point algorithm
PDIPM	Primal-dual interior-point method
PQ	Active/reactive power

PSO	Particle swarm optimization
PV	Photovoltaic
QP	Quadratic programming
RG	Reduced gradient (optimization method)
RTDS	Real-Time Digital Simulator
SLP	Sequential linear programming
SPEA2	Strength-pareto evolutionary algorithm
SQP	Sequential quadratic programming
SSR	Sub-synchronous resonance
SVC	Static VAR compensator
ULTC	Under-load tap-changing (transformer)
VAR	Volt-Amp-Reactive; the units of measurement for reactive power
VVO	Volt/VAR optimization

LIST OF MATHEMATICAL NOTATIONS

Symbol	Definition/Explanation
P_{Loss}	Total real power transmission losses
G_k	Series conductance of branch *k*
N_L	Number of branches in a power network
V_i, V_j	Voltage magnitudes at buses *i* and *j* respectively
θ_{ij}	Phase angle of the ij^{th} Y-matrix component
$P_i(V,\delta,\tau)$	Active power injection at bus *i*
$Q_i(V,\delta,\tau)$	Reactive power injection at bus *i*
P_{Gi}	Generator active power output at bus *i*
P_{Li}, P_{di}	Active power demand at bus *i*
Q_{Gi}	Generator reactive power output at bus *i*
Q_{Li}, Q_{di}	Reactive power demand at bus *i*
Q_{Si}, Q_{ci}	Reactive power source/sink magnitude at bus *i*
V_{Gi}	Generator terminal voltage magnitude at bus *i*
V_{Li}	Voltage magnitude at PQ bus *i*
τ_k	Tap position of ULTC connected in branch *k*
S_k	Apparent power flow in branch *k*
Y_{ij}	ij^{th} component of admittance matrix
δ	Voltage phase angle
d_k	Search direction of a gradient-based optimization method at the k^{th} iteration
x_k	Solution of a gradient-based optimization method at the k^{th} iteration
$H(x_k)$	Hessian of the objective function in the Newton-based solution of an optimization problem
$\nabla f(x_k)$	Gradient (*i.e*. 1st order partial derivatives) of the objective function in a gradient-based solution of an optimization problem
$f(x,u)$	Objective function of a general nonlinear programming problem
$g(x,u)$	Equality constraints of a general nonlinear programming problem
$h(x,u)$	Inequality constraints of a general nonlinear programming problem
$x^{\min}$, $x^{\max}$	Lower and upper bounds on the value of the state variable, *x*, respectively
$u^{\min}$, $u^{\max}$	Lower and upper bounds on the value of the control variable, *u*, respectively

$e_i,\ f_i$	Real and imaginary components of bus voltage at i^{th} bus, respectively
G_{ij}	Conductance of transmission line connecting buses *i* and *j* (real component of ij^{th} component of Y matrix)
B_{ij}	Susceptance of transmission line connecting buses *i* and *j* (imaginary component of ij^{th} component of Y matrix)
I_{ij}	Current magnitude in branch *ij*
s	Slack variable, added to an inequality constraint to transform it into an equality constraint
S	Diagonal matrix with the slack variables on the principal diagonal
μ	Barrier parameter in the primal-dual interior-point method
λ_E	Lagrange multiplier for the equality constraint
λ_I	Lagrange multiplier for the inequality constraint
Λ_I	Diagonal matrix with vector of Lagrange multipliers for inequality constraints on the principal diagonal
$\nabla_x L_\mu$	Gradient of the Lagrangian of the logarithmic barrier function with respect to *x*, in the primal-dual interior-point method
$\nabla^2_{xx} L_\mu$	Hessian of the Lagrangian of the logarithmic barrier function with respect to *x*, in the primal-dual interior-point method
∇Y	Gradient (or Jacobian) of generic nonlinear function Y, where Y can be *f* (objective function), *g* (equality constraints) or *h* (inequality constraints)
$\nabla^2_{xx} Y$	Hessian of generic nonlinear function Y, where Y can be *f* (objective function), *g* (equality constraints) or *h* (inequality constraints)
$F(X^k)$	General nonlinear function of independent variable X^k
$J(X^k)$	Jacobian of general nonlinear function $F(X^k)$ in the Newton method
ΔX^k	Increment added to X^k to advance it towards the solution in the Newton method
ΔY	Increment in generic variable Y, where Y can be either a primal or a dual variable in the primal-dual interior-point method
$\alpha_p^k,\ \alpha_d^k$	Step length for the primal and dual variables respectively, in the primal-dual interior-point method
ζ	Safety factor in the primal-dual interior-point method
ρ	Complementarity gap in the primal-dual interior-point method
σ	Centering parameter in the primal-dual interior-point method

γ	Scalar parameter used in the setting of the initial slack variables for the primal-dual interior-point method
g_{Pi}	Equality constraint corresponding to the active power flow balance equation
g_{Qi}	Equality constraint corresponding to the reactive power flow balance equation
h_{Vim}, h_{ViM}	Inequality constraints corresponding to the lower and upper bounds of the bus voltage magnitudes respectively, in the primal-dual interior-point method
h_{Qgim}, h_{QgiM}	Inequality constraints corresponding to the lower and upper bounds of the generator reactive power outputs respectively
h_{qcim}, h_{qciM}	Inequality constraints corresponding to the lower and upper bounds of the shunt reactive power outputs respectively
h_{tim}, h_{tiM}	Inequality constraints corresponding to the lower and upper bounds of the transformer load tap changer position respectively
s_{Vim}, s_{ViM}	Slack variables for the voltage bound inequality constraints
s_{Qgim}, s_{QgiM}	Slack variables for the generator reactive power bound inequality constraints
s_{qcim}, s_{qciM}	Slack variables for the shunt reactive power bound inequality constraints
s_{tim}, s_{tiM}	Slack variables for the load tap changer position bound inequality constraints
λ_{IVim}, λ_{IViM}	Lagrangian multipliers for the voltage bound inequality constraints
λ_{IQgim}, λ_{IQgiM}	Lagrangian multipliers for the generator reactive power bound inequality constraints
λ_{Iqcim}, λ_{IqciM}	Lagrangian multipliers for the shunt reactive power bound inequality constraints
λ_{Itim}, λ_{ItiM}	Lagrangian multipliers for the load tap changer position bound inequality constraints
$P_{i\Delta}$	Real power residue (i.e. mismatch) for the active power balance equation in the Newton-Raphson load flow algorithm
$Q_{i\Delta}$	Reactive power residue (i.e. mismatch) for the reactive power balance equation in the Newton-Raphson load flow algorithm
$V_{i\Delta}^2$	Squared voltage residue (i.e. mismatch) for the squared balance equation in the Newton-Raphson load flow algorithm
$\frac{\partial Y}{\partial x}$	Partial derivative of generic nonlinear function *Y* with respect to independent variable *x*
X_i^k	Current position of particle *i* at iteration *k* in the particle swarm optimization (PSO) algorithm

$p_{best,i}^{k}$	Personal best position of particle *i* at iteration *k* in the PSO algorithm
g_{best}^{k}	Global best position of the (entire) particle swarm at iteration *k* in the PSO algorithm
V_i^k	Velocity of particle *i* at iteration *k* in the PSO algorithm
$\vec{U}(0,\phi_i)$	Vector of uniformly distributed random numbers used to scale the cognitive and social components of the velocity update equation in the PSO algorithm
$\phi_1,\ \phi_2$	Cognitive and social coefficients (also known as acceleration coefficients) in the PSO algorithm
ω	Inertia weight in the PSO algorithm
$\omega^{\max},\ \omega^{\min}$	Initial and final values of the inertia weight, respectively, in the PSO algorithm
$\max\_iter$	Maximum number of iterations in the PSO algorithm
χ	Constriction coefficient in the PSO algorithm
r_i	Uniformly distributed random variable between 0 and 1, applied to particle *i*
$V_{gen,i}^{k}$	Generator voltage magnitude represented in particle *i* at iteration *k* in the PSO algorithm
$V_{gen,i}^{\min},\ V_{gen,i}^{\max}$	Minimum and maximum values of generator voltage magnitude represented in particle *i* in the PSO algorithm

CHAPTER ONE

THESIS INTRODUCTION

1.1 Introduction and background

The electric power system has undergone significant developments over the past century or so. Notable among the many developments are the deregulation and restructuring of the electrical power supply industry, the establishment of competitive electricity markets, and the resurgence of distributed and decentralized electric power generation (Li & Zhong, 2021; Rahimi, 2020; Jha & Dubey, 2020). This has naturally led to an increase in complexity of the modern power system. And in spite of these major developments and the increase in complexity, the primary objective of the power system has remained practically the same: to deliver electric power to the consumer in a secure, efficient, economical, reliable and sustainable way. Indeed, reliability of electric power supply has become so critical to modern society that even momentary unavailability of electric power may threaten to cause enough disruption to essential public (and private) services and normal daily activities to be considered practically intolerable.

Being able to deliver electric power with the required high reliability and security, while being economical, has necessitated the development of a variety of planning and operational strategies over the decades, by means of which the power system can be operated optimally as far as practicable. These strategies are collectively referred to as Optimal Power Flow (OPF). In the course of the operation of the power system, changes in load demand and network configurations may cause the system to operate outside of the nominal range, which could threaten the quality and even security or reliability of supply. To prevent or correct anomalous operating conditions, the system operator continually implements controls to regulate the production, absorption and flow of power at all levels in the system. Some of the controlled variables include generator active and reactive power outputs, reactive power compensation device outputs, transformer tap settings, and phase shifter settings, among others. OPF has evolved into a sophisticated computational tool employed in the determination of the optimal dispatch of all the system control variables so as to ensure the economical and secure operation of the system, while respecting many functional and operational constraints of the system.

The research presented in this thesis is primarily concerned with the theoretical design, development, and practical implementation of efficient algorithms that contribute to the secure, economical and reliable operation of modern complex power systems. This chapter covers the background to the research presented in this thesis. The motivation for the research is briefly discussed in section 1.2, followed by the

problem statement in section 1.3. Section 1.4 outlines the aim and objectives of the research, followed by the hypotheses that underlie the conducted research in section 1.5. Delimitation of the research and the assumptions taken in conducting the research are presented in sections 1.6 and 1.7 respectively. Section 1.8 outlines the research methodology, followed by the main research outputs in section 1.9. The outline of the thesis is then presented in section 1.10, and section 1.11 concludes the chapter with a brief summary of key points from the chapter.

1.2 Motivation for the research

The electrical power system is arguably one of the most complex engineering systems in existence. The complexity is in part due to the need to simultaneously consider several important operational objectives, some of which may be conflicting in nature. Engineering optimization has turned out to be the most effective way of dealing with the complexity associated with the efficient operation of the power system. Indeed, optimization of power system operation as a subject of study has quite a long history, and has benefited over the years from the advances in mathematical programming techniques and computational methods, although it even predates the advent of digital computers, which can be said to have revolutionized numerical optimization and computation in general. One of the most widely studied power system optimization problems is the optimal power flow (OPF) problem, the first complete formulation of which is generally attributed to Carpentier (1962). The OPF problem is concerned with the optimization of some aspect of power system operation (which could be economical, technical, environmental, etc.), while satisfying the functional and operational constraints of the system (Frank & Rebennack, 2016).

Reactive power and voltage control, otherwise referred to as optimal reactive power dispatch (ORPD) or Volt/VAR optimization (VVO), is one of the most important sub-problems of the OPF problem. It is primarily concerned with determination of the optimal coordinated dispatch of voltage-regulating devices and reactive power sources, with the aim of maintaining a secure voltage profile, while also optimizing some aspect of system operation, subject to functional and operational system constraints (Chebbo, 1990). Optimal reactive power dispatch plays a key role in the efficient transfer of real power, especially in the bulk power transmission system, and contributes significantly to the security, reliability, quality and economy of power system operation (Miller, 1982). Volt/VAR optimization has consistently been one of the most active areas of research in the field of optimization of power system operation, which gives evidence to the continued relevance of research in this aspect of power system operation. This is especially true in view of the growing complexity of modern power systems, in part due to such developments as electric power system

deregulation and restructuring, establishment of competitive electricity markets, the proliferation of distributed and decentralized electric power generation, and electric power grid modernization efforts under the smart grid paradigm. All these developments bring about the need for continuous advancement in optimization techniques and computational methods that are to support the secure, reliable, and economical operation of the 21st century power system and beyond (Myrda, 2013).

Volt/VAR optimization has a number of characteristics that make it a very challenging optimization problem, and much effort has been dedicated over the decades to the study of a variety of problem formulations, as well as the development of solution techniques for the various formulations, encompassing both classical or conventional optimization methods, as well as heuristic or intelligent search-based optimization methods. Most researchers consider either a classical or a heuristic optimization method in isolation. Recent works have applied such heuristic optimization methods as particle swarm optimization (Pijarski & Kacejko, 2018; Vitor & Vieira, 2018), genetic algorithm (Choden *et al.*, 2022), and expert system (Lomi & Limpraptono, 2017) to the Volt/VAR optimization problem. Examples of classical methods applied to the Volt/VAR optimization algorithm can be found in (Xu & Wu, 2022; Jha Dubey, 2020; Prabawa & Choi, 2019). The research work presented in this thesis investigates the design of efficient Volt/VAR optimization strategies considering both approaches, that is, classical and heuristic optimization techniques. Moreover, whereas most researchers make use of the polar formulation of the Volt/VAR optimization problem, both the polar and rectangular formulations are analysed in this thesis, and the rectangular formulation is found to have more favourable mathematical properties from the computational efficiency perspective, and is thus used in the studies conducted in this research. Both the model development of the Volt/VAR optimization problem as well as the theoretical design and practical implementation of the developed algorithms are thoroughly covered in the research.

1.3 Problem statement

The main problem dealt with in this research is the theoretical design and practical implementation of efficient methods for Volt/VAR optimization. This is in recognition of the pivotal role played by Volt/VAR optimization in ensuring the security, economy, efficiency and reliability of operation of modern complex power systems. The solution to the research problem is addressed through a thorough and comprehensive investigation of the state-of-the-art in the problem formulation and solution methods for the Volt/VAR optimization problem, followed by the theoretical design and practical implementation of the algorithms developed within the framework of this

research. The problem statement can thus be considered to consist of design and implementation sub-problems, as follows:

1.3.1 Design-oriented sub-problems

i. Study and comparative analysis of the various classical and heuristic optimization methods that have been applied to the Volt/VAR optimization problem over the decades.
ii. Formulation of the Volt/VAR optimization problem, considering both the polar and rectangular representations of the system voltages.
iii. Development and design of the Newton-Raphson load flow algorithm in rectangular coordinates, which is used as a component of the Volt/VAR optimization algorithm.
iv. Theoretical development and design of the primal-dual interior-point algorithm for application to the Volt/VAR optimization problem.
v. Theoretical development and design of the particle swarm optimization algorithm for application to the Volt/VAR optimization algorithm.
vi. Design of power system case studies to be used in the performance analysis of the developed Volt/VAR optimization algorithms.

1.3.2 Implementation-oriented sub-problems

Development of software for the implementation of:

i. The Newton-Raphson load flow algorithm in rectangular coordinates.
ii. The primal-dual interior-point method for solution of the Volt/VAR optimization problem, incorporating the Newton-Raphson load flow computation.
iii. The particle swarm optimization algorithm for solution of the Volt/VAR optimization problem, also incorporating the Newton-Raphson load flow computation.
iv. The power system case studies used to analyse the performance of the developed algorithms.

1.4 Research aim and objectives

1.4.1 Aim

The main aim of this research, in line with the problem statement outlined in the preceding section, is theoretical design and practical implementation of efficient solution methods for the Volt/VAR optimization algorithm, based on both classical/conventional and heuristic/intelligent search-based optimization techniques.

1.4.2 Objectives

The objectives supporting the realization of this aim can be stated as:

1. Thorough investigation of the state-of-the-art in problem formulation and solution techniques for the Volt/VAR optimization problem, considering both classical and heuristic optimization techniques.
2. Critical comparative analysis of classical and heuristic optimization techniques, based on key optimization performance criteria, such as computational efficiency, convergence characteristics, and solution quality.
3. Model development for the Volt/VAR optimization problem, considering both the polar and rectangular form of representing the system voltages.
4. Theoretical development of the primal-dual interior-point method (PDIPM) as the classical optimization technique applied to the solution of the Volt/VAR optimization problem.
5. Practical implementation of the PDIPM-based Volt/VAR optimization (PDIPM-VVO) algorithm, and a comprehensive performance analysis of the developed algorithm by means of a variety of power system case studies.
6. Theoretical development of the particle swarm optimization (PSO) algorithm as the heuristic optimization technique applied to the solution of the Volt/VAR optimization problem.
7. Practical implementation of the PSO-based Volt/VAR optimization (PSO-VVO) algorithm, and a comprehensive performance analysis of the developed algorithm by means of a variety of power system case studies.
8. Comparative analysis of the performance of the PDIPM and PSO algorithms as solution methods for the Volt/VAR optimization problem.
9. Making of recommendations for further research based on the results achieved from the current research.

1.5 Hypothesis

The hypotheses underpinning the research presented in this thesis are based on the investigative literature review that has been conducted in the field of problem formulation and solution techniques for the Volt/VAR optimization problem, spanning several decades, and encompassing both classical and heuristic optimization techniques. The following are the main hypotheses investigated in this research:

- Many different solution approaches for the Volt/VAR optimization problem have been explored and presented in the literature over the past decades. A detailed study of the individual techniques and a critical comparative analysis is conducted with the aim of establishing their key characteristics, as well their

relative strengths and weaknesses, which then forms the basis for the methods and algorithms developed and implemented in this thesis.

- The model formulation for the Volt/VAR optimization problem has a significant impact on the computational efficiency with which it is solved, especially when considered in conjunction with a specific optimization technique. The polar and rectangular representations of the Volt/VAR optimization problem are considered in this thesis, and their relative merits and demerits are discussed. Based on their mathematical properties, the rectangular formulation is found to be especially suitable for the solution algorithms developed in this thesis.
- There are a number of performance criteria on the basis of which the effectiveness of an optimization algorithm can be evaluated, for example, computational speed, accuracy and quality of the solution, convergence properties, and ability to effectively handle inequality constraints, among others. The hypothesis is that classical optimization methods perform relatively better in some of these characteristics, and heuristic methods perform better in others. Thus, the solution to the Volt/VAR optimization algorithm is developed on the basis of both a classical optimization method (i.e. PDIPM) as well as a heuristic optimization method (i.e. PSO), and a comparative analysis is conducted to ascertain their relative performance with respect to the Volt/VAR optimization problem.

The investigation of these hypotheses leads to the design and implementation of efficient and scalable solution methods for the Volt/VAR optimization problem, based on both the classical and heuristic optimization techniques, as presented in this thesis.

1.6 Delimitation of the research

The focus of the research presented in this thesis is on the theoretical study and analysis of classical and heuristic optimization methods, and their application to the Volt/VAR optimization problem. More specifically, the following aspects of the research are emphasized:

- Development and implementation of the Newton-Raphson-based load flow algorithm formulated in rectangular coordinates of system voltages.
- Development and implementation of the primal-dual interior-point method-based Volt/VAR optimization algorithm, which incorporates the Newton-Raphson-based load flow computation.
- Development and implementation of the particle swarm optimization-based Volt/VAR optimization algorithm, which also incorporates the Newton-Raphson-based load flow computation.

- Performance analysis of the developed algorithms by means of several power system case studies varying in size, in order to analyse such characteristics as computational efficiency, solution quality, convergence properties, and scalability of the algorithms.

Many algorithms are studied in the framework of this thesis, but only the ones outlined above are further developed and implemented.

1.7 Assumptions

The following assumptions are made in approaching the problem of developing and implementing solution methods for the Volt/VAR optimization problem:

- Volt/VAR optimization is treated as a static optimization problem, such that the load demand and active power generation are considered to remain constant at the scheduled values throughout the optimization process. This is a simplifying assumption, as in reality load demand tends to vary over time, and active power generation must correspondingly be adjusted to follow the load demand variation.
- The system power loss calculation is based only on the transmission line losses. This is a simplifying assumption, as many other system components, such as generators and transformers, also contribute to system power losses. In this study, however, their contribution is taken to be relatively negligible.
- In developing the model for the Volt/VAR optimization problem, only the aspects relevant to the optimization process are modelled, and only a selection of system state and control variables are incorporated into the model, which is a simplification meant to handle the model complexity.
- The PDIPM algorithm requires initialization of certain parameters (e.g. barrier parameters, slack variables, etc.), which is largely problem-dependent. Once these parameters are initialized, some of them stay constant throughout the optimization process, others are adjusted algorithmically. No re-initialization of the parameters is performed, unless a different run of the algorithm is conducted in order to evaluate it for a different set of parameters.
- The PSO algorithm requires initialization of some random parameters as well as some static parameters (e.g. cognitive and social acceleration coefficients, which are held constant throughout the optimization process), and these are also largely problem-dependent.
- The performance analysis of the developed algorithms focuses on the measurable aspects of the algorithm performance, such as the number of iterations taken by the algorithm to converge and the corresponding running time, and the solution quality as judged by the achieved amount of power loss

reduction and voltage profile improvement. Many other aspects are not considered.

- Other than the primal-dual interior-point and particle swarm optimization algorithms, other optimization algorithms are not considered for implementation in this research.

1.8 Research methodology

The methodology followed in conducting the research presented in this thesis encompasses three main strands, these being (1) a thorough literature review on the various aspects pertaining to the research, (2) a theoretical development of the components of the research as stated in the research objectives, (3) a practical implementation of the developed solutions to the research problem, and comprehensive performance analysis of the developed methods.

1.8.1 Literature review

The review of previous work related to the proposed research has considered the various aspects related to Volt/VAR optimization. The following topics have been the main focus of the literature review:

1. Formulation of the Volt/VAR optimization problem, considering the objectives, constraints, and decision or control variables, and the representation of the system voltages in both polar and rectangular coordinates.
2. The main reactive power and voltage control devices that are typically employed in Volt/VAR optimization.
3. Classical/conventional methods for Volt/VAR optimization.
4. Heuristic/intelligent search-based methods for Volt/VAR optimization.
5. Performance analysis criteria for optimization in general, and Volt/VAR optimization in particular.

1.8.2 Theoretical development

The knowledge and information derived from the literature review has been used as the basis for the theoretical development of the solution to the research problem. Specifically, this has resulted in the:

- Definition of the problem formulation for the Volt/VAR optimization problem for both the polar and rectangular coordinate representations of the system voltages.
- Development of the Newton-Raphson load flow algorithm in rectangular coordinates, which is then incorporated into the Volt/VAR optimization algorithm.

- Development of the primal-dual interior-point method and adaptation to the requirements of the solution algorithm for the Volt/VAR optimization problem.
- Development of the particle swarm optimization algorithm and adaptation to the requirements of the solution algorithm for the Volt/VAR optimization problem.

1.8.3 Practical implementation and performance analysis

The practical implementation and performance analysis entails:

- Development of the software for the Newton-Raphson load flow algorithm that has been theoretically developed, as outlined in section 1.8.2.
- Development of the software for the PDIPM-based Volt/VAR optimization algorithm incorporating the Newton-Raphson algorithm.
- Development of the software for the PSO-based Volt/VAR optimization algorithm incorporating the Newton-Raphson algorithm.
- Comprehensive performance analysis of all the developed algorithms by means of several power system case studies varying in size from small (3-bus system) to large (118-bus system).

1.9 Main research outputs/deliverables

The principal contributions of this research can be enumerated as: (1) comprehensive literature study and review; (2) problem formulation and model development for the Volt/VAR optimization problem; (3) theoretical development and design of the algorithms used in solving the Volt/VAR optimization problem; (4) software development for the implementation of the developed algorithms; and (5) comprehensive performance analysis of the developed algorithms by means of a variety of power system case studies:

1. The comprehensive literature review covers the following:
    a. Problem formulation and solution methods for the Volt/VAR optimization problem, considering both the classical/conventional and heuristic/intelligent search-based optimization techniques.
    b. Critical comparative analysis of the classical and heuristic optimization techniques, highlighting their individual characteristics, as well as their relative strengths and weaknesses.
    c. Study and presentation of the main reactive power and voltage control devices that are typically employed in the solution of the Volt/VAR optimization problem.

2. The problem formulation and model development for the Volt/VAR optimization problem considers both the polar and rectangular coordinate representations of the system voltages.
3. The theoretical algorithm development and design encompasses:
    a. The Newton-Raphson load flow algorithm in the rectangular coordinate representation of the system voltages.
    b. The primal-dual interior-point algorithm for Volt/VAR optimization (PDIPM-VVO), formulated in rectangular coordinates, which incorporates the rectangular-coordinate Newton-Raphson load flow computation.
    c. The particle swarm optimization algorithm for Volt/VAR optimization (PSO-VVO), also formulated in rectangular coordinates, and incorporating the rectangular-coordinate Newton-Raphson load flow computation.
4. The software development and implementation encompasses the:
    a. Rectangular-coordinate Newton-Raphson load flow algorithm.
    b. Primal-dual interior-point method-based Volt/VAR optimization (PDIPM-VVO) algorithm.
    c. Particle swarm optimization-based Volt/VAR optimization (PSO-VVO) algorithm.
5. The comprehensive performance analysis of the developed algorithms makes use of five power system case studies, and analyses the performance of both the PDIPM-VVO and PSO-VVO algorithms, focusing on the following performance aspects:
    a. The quality of the solution in terms of the magnitude of real power loss percentage reduction and the voltage profile improvement.
    b. The computational efficiency of the algorithm in terms of the required number of iterations and runtime.
    c. The scalability of the developed algorithms when applied to test systems ranging in size from 3-bus to 118-bus system.
    d. The impact of the swarm size on the solution quality and the computational cost of the PSO algorithm.

1.10 Thesis outline

The thesis document comprises seven chapters, whose contents can be summarized as follows:

Chapter one (i.e. this chapter) introduces the research presented in this thesis. Among the topics covered are the motivation for the research, the problem statement,

the research aim and objectives, the hypotheses underpinning the research, the delimitation of the research, the assumptions that have been made in the process of conducting the research, the research methodology that has been followed, and the main research outputs.

Chapter two presents a comprehensive literature review on the problem formulation and solution methods for the Volt/VAR optimization problem, covering both the classical/conventional as well as the heuristic/intelligent search-based optimization techniques. A study of the main reactive power and control devices that are typically employed in the Volt/VAR optimization are is also presented. A key result in this chapter is a critical comparative analysis of the performance characteristics of classical and heuristic optimization methods, which emphasizes their relative strengths and weaknesses.

Chapter three focuses on the model development for the Volt/VAR optimization problem, and discusses the objective functions, constraints, as well as decision or control variables for the problem. Two models are presented, one based on the polar representation of the system voltages, the other based on the rectangular representation of the system voltages. The two models are compared in terms of their mathematical properties, and their suitability for application to the current study.

Chapter four presents the theoretical design and practical implementation of the primal-dual interior-point method, as well as an application example based on a general nonlinear programming problem that is used to demonstrate both the theoretical development and practical implementation of the algorithm.

Chapter five presents the theoretical development and practical implementation of the primal-dual interior-point method-based Volt/VAR optimization (PDIPM-VVO) algorithm, as well as the Newton-Raphson load flow algorithm, formulated in rectangular coordinates. Five simulation case studies based on power systems ranging in size from 3-bus to 118-bus are used to analyse the performance of the developed PDIPM-VVO algorithm.

Chapter six presents the theoretical development and practical implementation of the particle swarm optimization-based Volt/VAR optimization (PSO-VVO) algorithm, which also incorporates the Newton-Raphson load flow algorithm presented in chapter five. Similar to chapter five, several simulation case studies based on power systems ranging in size from 3-bus to 118-bus are used to analyse the performance of the developed PSO-VVO algorithm. A comparative analysis of the PDIPM-VVO and PSO-VVO algorithms is also presented in this chapter.

Chapter seven presents the conclusion, the deliverables of the thesis, and the recommendation for further research based on the outcomes of the current research.

1.11 Conclusion

This chapter has introduced the research presented in this thesis. Among the topics covered are the motivation for the research, the problem statement, the research aim and objectives, the hypotheses underpinning the research, the delimitation of the research, the assumptions that have been made in the process of conducting the research, the research methodology that has been followed, and the main research outputs. Subsequent chapters will develop the themes outlined in this chapter, which will culminate into the achievement of the aim and objectives stated for this research. The main result of this chapter has been to set the background for the research conducted and presented in this thesis.

To contextualize the current research and to lay the foundation for the developmental work carried out in the framework of this research, the following chapter presents a comprehensive literature review on the problem formulation and solution methods for the Volt/VAR optimization problem, covering both the classical/conventional as well as the heuristic/intelligent search-based optimization techniques.

CHAPTER TWO

LITERATURE REVIEW ON CLASSICAL AND HEURISTIC METHODS FOR VOLT/VAR OPTIMIZATION

2.1 Introduction

Optimization of power system operation as a subject of study has quite a long history, enriched over the years by advances in mathematical programming techniques and computational methods, but certainly predating the advent of digital computers which have revolutionized numerical optimization and computation in general. One of the most widely studied power system optimization problems is the Optimal Power Flow (OPF) problem, the first complete formulation of which is generally attributed to Carpentier (1962), (Cain *et al.*, 2012). The OPF problem seeks to optimize some aspect of power system operation (could be economical, technical, environmental, etc.), while satisfying the physical and operational constraints of the system (Frank & Rebennack, 2016).

Volt/VAR Optimization (VVO) can be considered to be a sub-problem of the OPF problem (or a variant formulation thereof) that is mainly concerned with the determination of the optimal coordinated dispatch of voltage-regulating devices and reactive power sources so as to maintain a secure voltage profile, while also optimizing some aspect of power system operation, subject to physical and operational system constraints (Feng & Peterson, 2010; Chebbo, 1990). Optimal reactive power dispatch plays a key role in the efficient transfer of real power, especially in the bulk power transmission system, and contributes significantly to the security, reliability, quality and economy of power system operation (Miller, 1982). The extensive research that has been (and continues to be) conducted in the area of Volt/VAR optimization gives evidence to the continued relevance of research in this aspect of power system operation, particularly in the wake of changes taking place in the electric power system, spurred on by such developments as electric power system deregulation, electric grid modernization under the paradigm of the smart grid, and the rapid growth of renewable and distributed power generation (Li & Zhong, 2021; Rahimi, 2020; Jha & Dubey, 2020; Golkar & Rajabzadeh, 2009; Puttgen et al., 2001; Lewis, 2000; Joos et al., 2000). Largely progressive as all these developments are, they nonetheless pose a significant challenge to the power system operator (Bekhouche, 2002), and hence the growing need for advancements in optimization techniques and computational methods that will support the secure, reliable, and economical operation of the 21st century power system and beyond (Myrda, 2013).

Volt/VAR optimization has a number of characteristics that make it a very challenging optimization problem, and much effort has been dedicated over the decades to the study of a variety of problem formulations, as well as the development of solution techniques for the various formulations. Key developments in the treatment of the OPF problem over the years have been presented in a number of review papers, some notable ones being (Alghamdi, 2022; Risi, B-G. *et al.*, 2022; Krishnamurthy & Tzoneva, 2012; Frank *et al.*, 2012a, 2012b; Pandya & Joshi, 2008; Momoh *et al.*, 1993; Huneault & Galiana, 1991; Alsac *et al.*, 1990; Happ, 1977). Aspects of interest that have been emphasized in these review papers have mainly been the problem formulation, as well solution techniques, considering both the classical/deterministic and the non-deterministic/artificial intelligence-based optimization methods.

A few review papers have focused on solution techniques for Volt/VAR optimization. A review of literature on reactive power planning has been presented by Zhang and Tolbert (2007). Taylor *et al* (2001) present a review of algorithmic and heuristic methods for Volt/VAR control. Lin *et al* (2003) focus in their review of Volt/VAR control on reactive power sources and their control devices, as well as discussing a number of solution methods for the Volt/VAR control problem.

Reactive power planning, a problem that is closely related to Volt/VAR optimization (or optimal reactive power dispatch), focuses on optimal investment in new reactive power sources to meet future reactive power compensation needs (Zhang & Tolbert, 2007). The relevance of effective reactive power planning has become even more pronounced in recent years, due to the need to account for the impact of the growing share of variable renewable generation such as wind and photovoltaic power generation on reactive power compensation. A multi-period, multi-scenario corrective security-constrained OPF has been explored by Savvopoulos *et al* (2019) as a way of dealing with increasing penetration of variable renewable generation. Ghodrati *et al* (2019) proposed a probabilistic multi-objective reactive power planning framework that considers large-scale wind generation integration. Li *et al* (2019) investigated the coordination of the reactive power control of large-scale renewable generation with the main grid as a way of enhancing the voltage stability of the entire system.

Recognition has continued to increase among utilities and researchers of the role to be played by smart inverters in various forms of grid support. As an example, California Rule 21, which regulates the integration of distributed generation (DG) to the power grid, has implemented an adjustment to the rule that requires the use of advanced (i.e. smart) inverters capable of performing a variety of grid support functions, such as Volt/VAR management (Cha *et al.*, 2020). Multi-agent deep reinforcement learning has been applied to the control of DGs via smart inverters as a

way of adapting to time-varying conditions, as well as the spatial and temporal uncertainties resulting from intermittent generation (Zhang *et al.*, 2021; Liu *et al.*, 2021). The overarching concept underlying many of these works is to exploit the capabilities of modern smart inverters to actively regulate inverter-based DG output so as to support network functions such as voltage regulation, network loss minimization, and electricity market-based day-ahead power dispatch, among others (Aldahmashi & Ma, 2022; Li & Zhong, 2021; Xu & Wu, 2020; Ding et al., 2020).

This chapter presents an up-to-date comprehensive survey of the main optimization methods that have been applied to the Volt/VAR optimization problem over the decades. Both classical/conventional and heuristic/intelligent search-based optimization methods are covered. Each optimization method is discussed in detail, its strengths and drawbacks are highlighted, and a thorough comparative analysis of the key characteristics of the classical and heuristic methods is presented. Figure (2.1) depicts a graph of the number of publications that have been reviewed, plotted against the year of publication, and Figure (2.2) indicates the number of each of the optimization algorithms that have been covered.

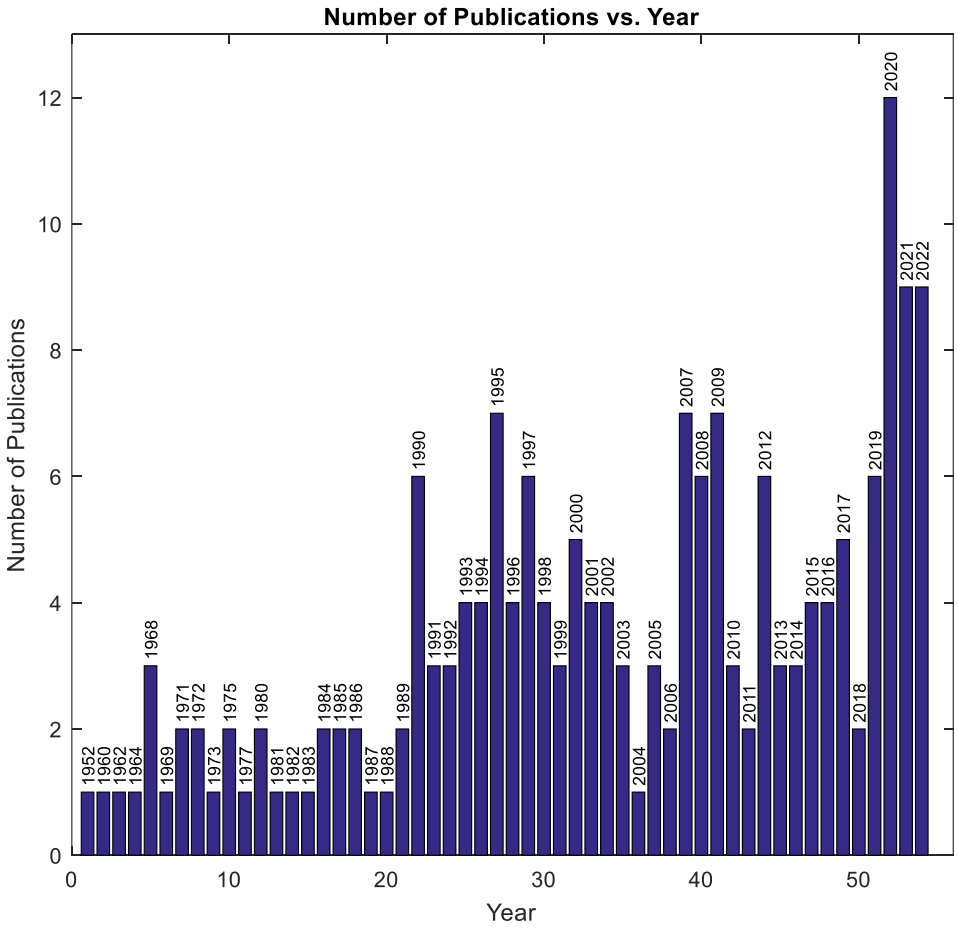

Fig. 2.1: Number of publications reviewed plotted against year of publication

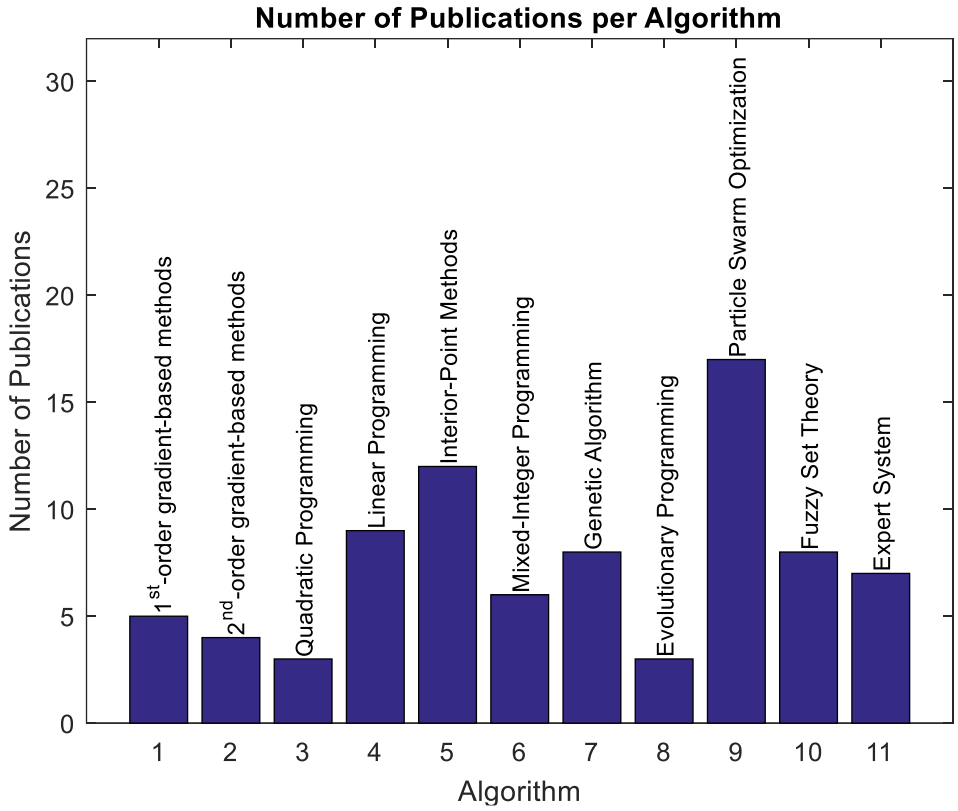

Fig. 2.2: Number of publications reviewed in terms of algorithm

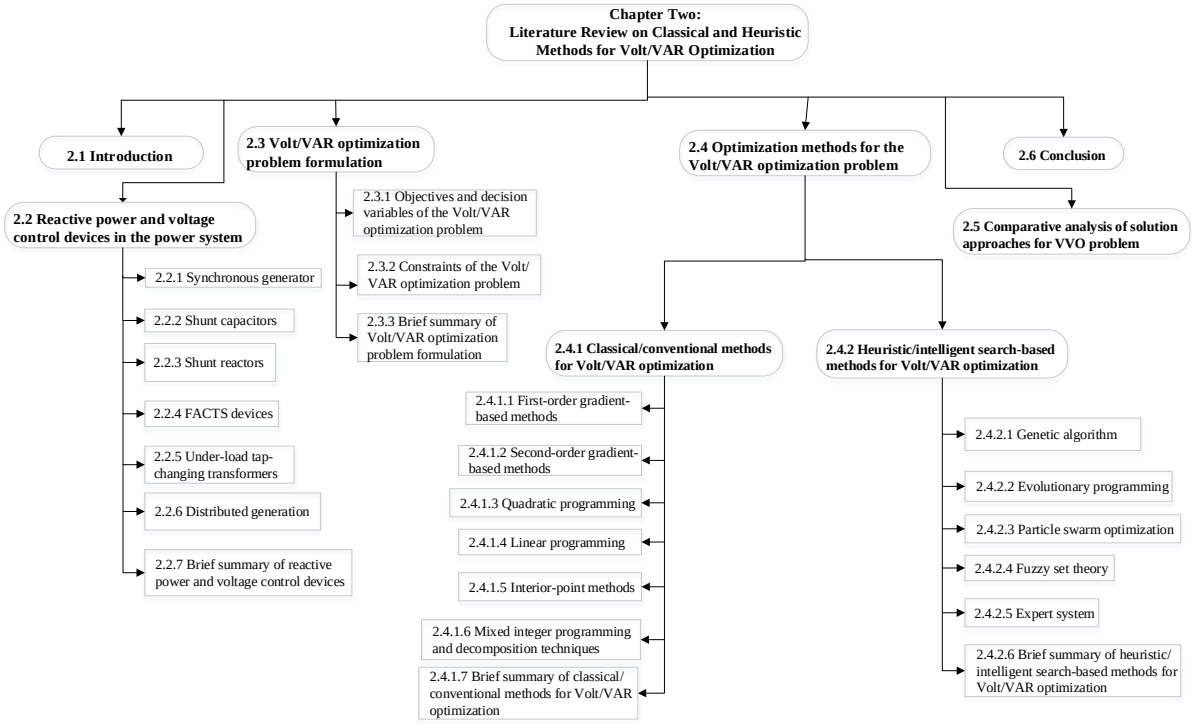

Fig. 2.3: pictorial representation of the content of this chapter

The rest of the chapter is organized as follows. Section 2.2 briefly discusses the main devices for reactive power and voltage control in the power system. The Volt/VAR optimization problem formulation is presented in section 2.3, to contextualize the discussion of the solution approaches presented in section 2.4, covering both

classical and heuristic methods. In section 2.5, a comparative analysis of the solution methods is presented, and the concluding remarks for the chapter are given in section 2.6. Figure (2.3) depicts a pictorial representation of the content covered in this chapter.

2.2 Reactive power and voltage control devices in the power system

As mentioned in the introductory section, reactive power and voltage control plays a pivotal role in the secure and economical operation of the power system. In the course of the operation of a power system, a variety of phenomena occur that need some form of intervention in order to maintain the system voltage, frequency and other vital system parameters within the nominal range. These phenomena may be classified as either steady-state or dynamic, depending on the speed of response required in addressing them. Table (2.1) lists (not in any order of precedence) some of the main phenomena, the addressing of which typically requires reactive power and voltage control of some form (EPRI, 1984). In the following sub-sections, the main power system devices that are typically employed in the provision of reactive power and voltage control are briefly discussed. The devices discussed are depicted in Figure (2.4).

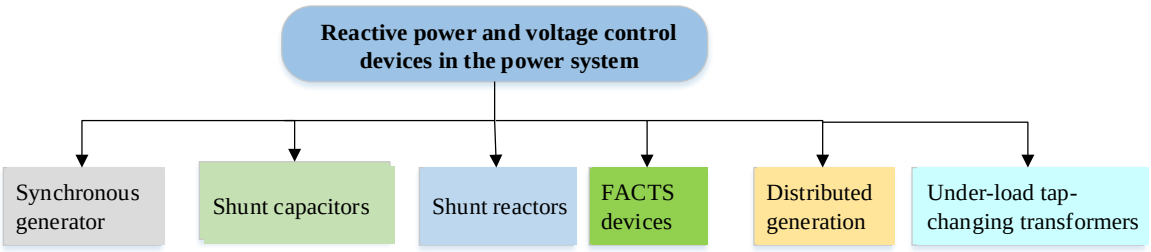

Fig. 2.4: Reactive power and voltage control devices discussed in this section

2.2.1. Synchronous Generator

Although the synchronous generator's main role in the power system is to supply active power demand, it is also principally used to regulate system reactive power, and has the ability to either generate (leading) or absorb (lagging) reactive power, depending on whether it is overexcited or under-excited. An automatic voltage regulator continually adjusts the generator's field excitation in response to system conditions, usually so as to maintain the terminal voltage or voltage at some other system bus at a desired level. The fast response characteristic of the synchronous generator's reactive power generation/absorption implies that it can be used to remedy dynamic system phenomena requiring Volt/VAR control. However, its reactive power supply/absorption capability is limited by the machine thermal and steady-state stability limits, and is a function of the real power output (Kundur, 1994).

When a synchronous generator is specially designed and operated so at to generate reactive power only (i.e. real power output set to be zero), it is referred to as a synchronous condenser. An example of a synchronous condenser integrated into a high-voltage direct-current (HVDC) system is depicted in Figure (2.5). As a device dedicated to reactive power supply/absorption, a synchronous condenser typically has automatic controls that enable fast dynamic response to system anomalies, and has a short-time overload capability that can be utilized in extreme situations. The main disadvantage of the synchronous condenser is its higher capital and maintenance costs compared to other solutions for reactive power supply and absorption (Zhou *et al.*, 2018).

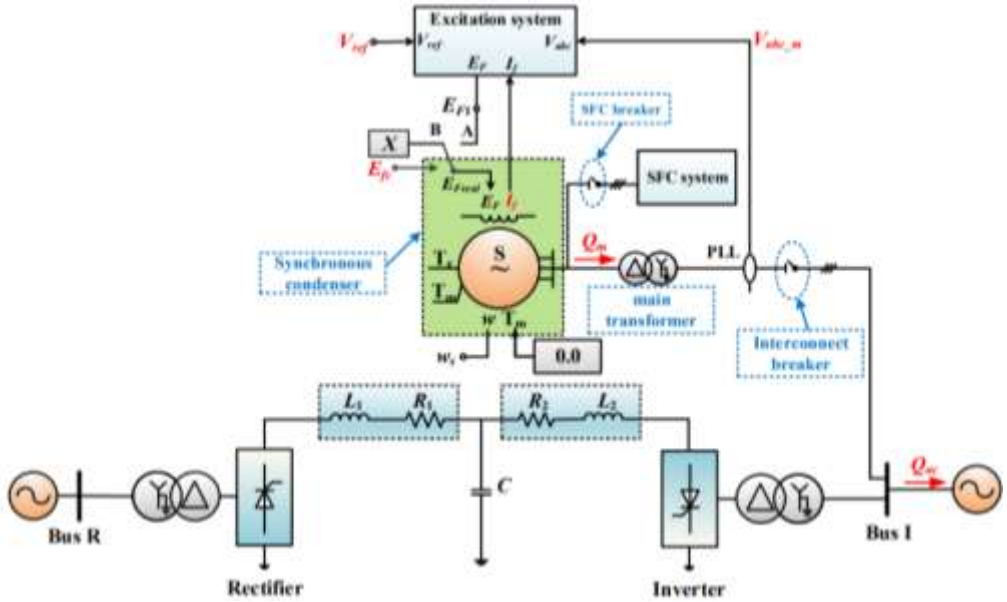

Fig. 2.5: Schematic of a synchronous condenser integrated into an HVDC system (adapted from Wang *et al.*, 2019)

Table 2.1: typical power system phenomena requiring Volt/VAR optimization

Steady-state phenomena (slow response)	Dynamic phenomena (fast response)
Low voltages	Fluctuating loads or impact loads
High voltages	Switching surges or load rejection overvoltages
Large voltage variability	Voltage instability (load voltage collapse)
Excessive reactive power flow (or losses)	Transient or dynamic instability
Normal requirements for HVDC converters	Instability due to subsynchronous resonance (SSR)
Steady-state stability	Variable system phase imbalances
	Dynamic reactive requirements at HVDC terminals
	Small-signal oscillations

2.2.2. Shunt capacitors

Shunt capacitors constitute a flexible and economical means of providing leading reactive power, which is typically required to boost system voltages during heavy loading periods, or to improve system power factor. Their flexibility stems from their

modular nature, large banks can be constructed from several small-size units, which in turn gives them the characteristics of greater control, expansion capability, transportability, and availability. Compared to synchronous generators, shunt capacitors, being static components, have lower maintenance costs, and are generally a cheaper source of reactive power. Their response characteristics, however, make them a lot less effective than synchronous generators in responding to dynamic system phenomena (FERC, 2005). Also, unlike synchronous generators, they supply discrete (rather than continuous) reactive power, which may affect their treatment in optimization problems, as the corresponding control variable will be discrete rather than continuous. A schematic of a shunt capacitor bank connected to the tertiary winding of a three-phase power transformer is depicted in Figure (2.6).

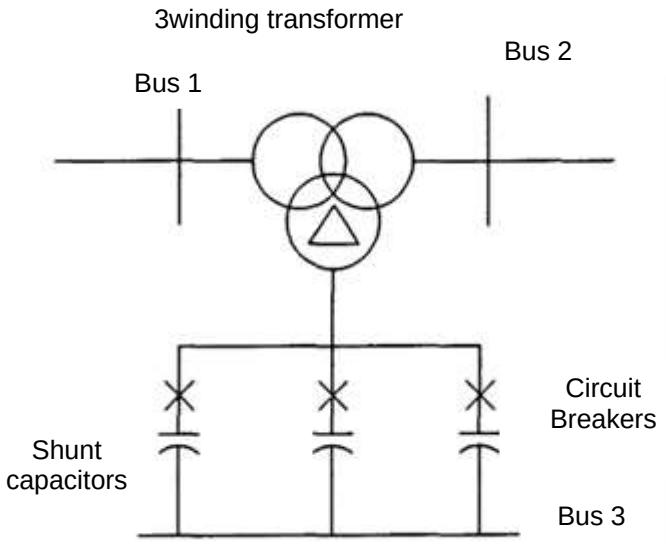

Fig. 2.6: Schematic of a shunt capacitor bank connected to the tertiary winding of a transmission system transformer (adapted from Kundur, 1994)

2.2.3. Shunt reactors

Shunt reactors are employed in the bulk transmission system to remedy abnormally high transmission voltages, often in lightly loaded conditions, when the capacitive line-charging effects of high-voltage transmission lines tend to lead to conditions exceeding design levels. They are typically required in extra high voltage lines longer than 200 km, where the effects of capacitive line charging can be quite pronounced (Kundur, 1994). A schematic of a tapped shunt reactor connected to the tertiary winding of a three-phase power transformer is depicted in Figure (2.7).

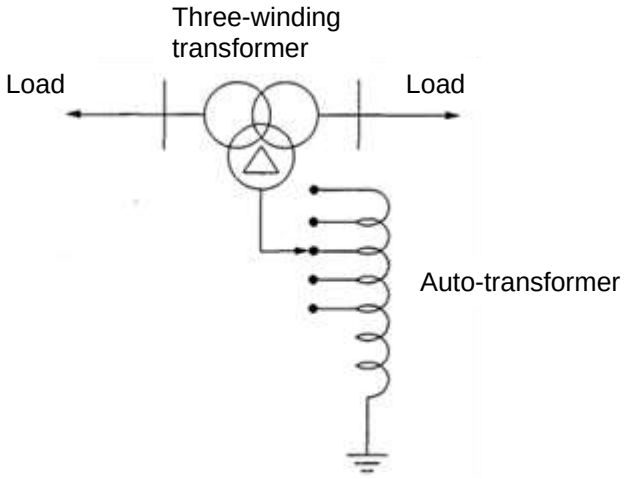

Fig. 2.7: Schematic of a tapped shunt reactor connected to the tertiary winding of a transmission system transformer (adapted from Kundur, 1994)

2.2.4. FACTS devices

Flexible AC Transmission System (FACTS) devices have in recent times emerged as a vital component in the efficient control of active/reactive power and voltage magnitude and frequency (Hongji *et al*., 2022). A static VAR compensator (SVC), an example of which is depicted in Figure (2.9), has ability to continuously vary inductive or capacitive reactive power injection into the system, making use of power electronic technologies. In terms of construction, an SVC can be thought of as being comprised of a controllable reactor and a fixed capacitor (as shown in Figure 2.8), both of which are controlled by means of power electronic switches in accordance with the required reactive power injection, the main purpose being to maintain bus voltage at some specified level.

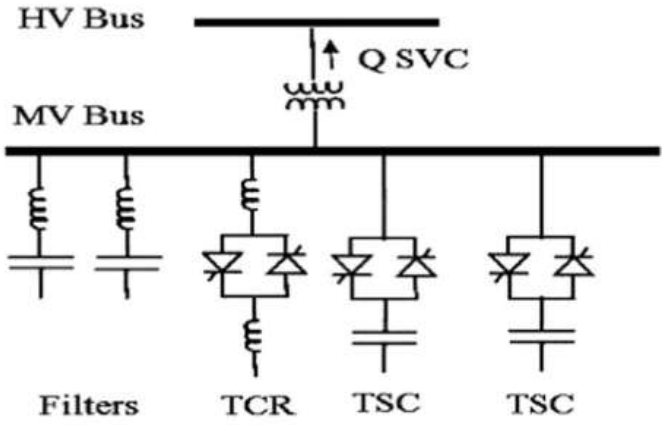

Fig. 2.8: Static VAR compensator (SVC), an example of a FACTS device (adapted from Gandoman *et al.*, 2018)

Use of power electronic switches gives FACTS devices ability to provide continuous, instantaneous reactive power, and are thus suitable for addressing many of the dynamic system phenomena associated with Volt/VAR control. Some drawbacks of

FACTS devices are their relatively higher cost, and possibly negative impact on system power quality due to harmonic generation by power electronic switches (Chebbo, 1990).

2.2.5. Under-load tap-changing transformer

A transformer equipped with a load tap-changing mechanism (LTC) can adjust the transformer turns ratio in response to system conditions so as to keep the system voltage within desired ranges. So unlike the devices discussed in the preceding sub-sections, the LTC is not a reactive power source, but rather a voltage-regulating device. Tap positions are discrete points on the windings of the transformer which can be varied so as to realize different transformer turns-ratios, and correspondingly different voltage transformations. The voltage can thus only be varied in discrete steps (rather than continuously). Figure (2.9) shows an example of an under-load tap-changing transformer, with the tap-changing mechanism located on the primary side of the transformer (Csany, 2014). Equipping a transformer with an LTC adds significantly to the cost, and thus requires the utility provided thereby to justify the added cost, which is typically the case in the bulk transmission system where effective voltage regulation is of paramount importance to the secure and efficient operation of the system (EPRI, 1984).

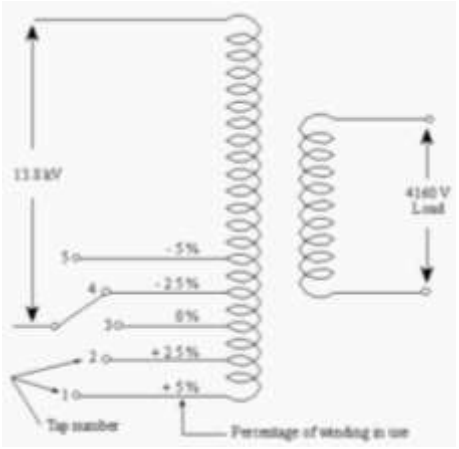

Fig. 2.9: Schematic of an under-load tap-changing transformer with the tap-changer located on the primary side of the transformer (adapted from Csany, 2014)

2.2.6. Distributed generation

The proliferation of diverse distributed generation technologies in the power system has been one of the most noteworthy developments in the electric power industry in recent years. A photovoltaic generation system is depicted in Figure (2.10) as a typical example of distributed generation systems. Along with their growth, the need for their contribution to the provision of grid ancillary services has been identified as key to their sustained growth and overall improvement in grid operation (Pecas Lopez

et al., 2007). Thus, the consideration of distributed generation in Volt/VAR optimization has become an active area of research (Dulau & Bica, 2022; Gupta *et al.*, 2021; Singh *et al.*, 2021; Zhang *et al.*, 2021; Ding *et al.*, 2020; Xu *et al.*, 2020). The diversity of the technologies (incorporating both conventional synchronous generators and newer technologies in the form of inverter-based generation systems) certainly presents an opportunity for exploiting this form of system resource in the meeting of the various steady-state and dynamic system requirements for the provision of reactive power and voltage control (Braun, 2007; FERC, 2005).

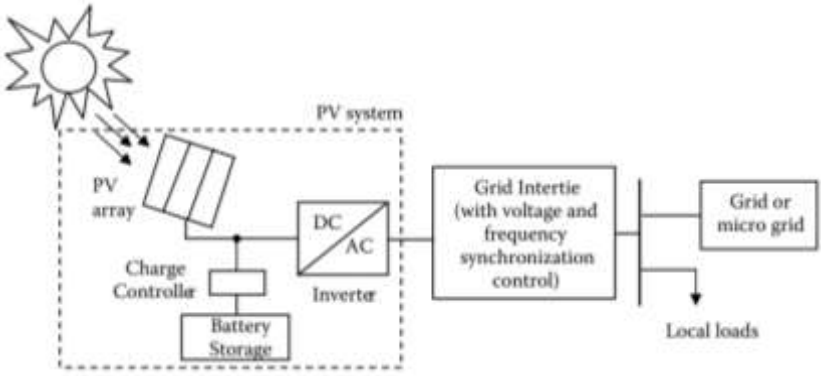

Fig. 2.10: Diagram of a photovoltaic (PV) generation system as an example of a distributed generation system (adapted from Momoh, 2007)

Table 2.2: Main characteristics of reactive power and voltage control devices

Volt/VAR Device	*Relative Cost Per MVA*	*Reactive Supplied*	*Continuous /Discrete*	*Dynamic response*	*Advantages*	*Disadvantages*	*Application (dynamic/steady-state)*
Synchronous Generator/Condenser	High	Lag/lead	Continuous	Fast	Fast response, flexible, strong stabilizing effect	High cost, complex controls	Dynamic
Shunt Capacitor	Moderate	Lead	Discrete	Slow	Flexible, modular, low maintenance requirement	Slow response, non-continuous (i.e. discrete)	Steady-state
Shunt Reactor	Moderate	Lag	Continuous	Slow	Simple, low maintenance requirement	Slow response, non-continuous (i.e. discrete)	Steady-state
FACTS	High	Lag/lead	Near-continuous	Fast	Fast response dynamics, flexible VAR supply/absorption	High cost, complex controls	Dynamic
ULTC Transformer	High	N/A	Discrete	Slow	Effective means of regulating system voltage	High cost, frequent operation may lead to high maintenance costs	Steady-state
Distributed Generation	Technology-dependent	Lag/lead	Continuous	Generally fast	Flexible, modular, can provide VAR support locally	VAR support may impact revenue from active demand supply	Dynamic

2.2.7. Brief summary of reactive power and voltage control devices

Reactive power and voltage control is a key component of the power system's energy management system. This section has reviewed the main power system devices that are typically employed in the provision of reactive power and voltage control,

highlighting their main characteristics, and how they contribute to the addressing of the static and dynamic reactive power and voltage control requirements of the power system. Table 2.2 summarizes the principal characteristics of the major devices for reactive power and voltage control that have been discussed in this section. It is the operation of these devices that has to be optimized in order to realize the secure, efficient and economical operation of the power system, as further discussed in section 2.4. The next section briefly discusses the problem formulation for the Volt/VAR optimization problem.

2.3 Volt/VAR optimization problem formulation

Volt/VAR optimization is a constrained optimization problem. The main components of the problem formulation are the objective function, the decision or control variables, and the constraints to be satisfied by the optimal solution to the problem. Mathematically, the objective and constraint functions can be either linear or nonlinear, the decision variables can be either continuous or discrete. Various combinations of these choices will lead to different formulations of the problem. The salient aspects of these components of the VVO problem formulation are briefly discussed in the following sub-sections. This helps to set the context for the discussion of solution methods in section 2.4.

2.3.1. Objectives and decision variables of the Volt/VAR optimization problem

There are multiple ways in which optimal reactive power dispatch contributes to the economical, secure and efficient operation of the power system. This can directly be related to the objectives of Volt/VAR optimization. Power loss minimization has featured as the main objective in many research works over the years, both in earlier publications (Billinton & Sachdeva, 1973; Narita & Hammam, 1971; Hano *et al.*, 1969; Peschon *et al.*, 1968), and in more recent ones (Vitor & Vieira, 2018; Ji *et al.*, 2017; Sivalingam *et al.*, 2017; Zheng *et al.*, 2017; Ahmadi *et al.*, 2015). Maintaining network voltages within the specified range of nominal values constitutes another key objective for Volt/VAR optimization (Padilha-Feltrin *et al.*, 2015; De Souza & De Almeida, 2010; Su & Lin, 1996). Then there is maximization of voltage security (De & Goswami, 2014; Katuri *et al.*, 2012; Venkatesh *et al.*, 2000), and minimization of the frequency of operation of the Volt/VAR control devices (Jin *et al.*, 2019; Rabiee & Parniani, 2013; Roytelman *et al.*, 1995). Each of these objectives enhances in one way or another the economics, security, power quality, and efficiency of power system operation.

As there are several objectives that can be considered, the VVO problem may be formulated as a single-objective optimization problem (the most prevalent formulation, based on the reviewed literature) or as a multi-objective optimization problem (for

example, Rabiee & Parniani, 2013; Ji *et al.*, 2017). A multi-objective formulation permits the simultaneous consideration of economic and security objectives, for example. A major concern in multi-objective optimization is how to formulate the problem in such a way that the obtained solution is optimal for all the considered (and potentially conflicting) objectives. The most common approach is to reduce the multiple objectives to a single objective function by a weighted summation of the individual objectives. This approach has the desirable characteristic of being simple to implement, but also has a number of drawbacks, such as the dependence of the obtained solution on the choice of the weighting vector, with considerable reliance on user expertise and experience. The subject of multi-objective optimization is discussed in detail by Deb (2001).

A key consideration regarding the objective function of the VVO problem is its dynamic characteristics, particularly in terms of whether it is linear or nonlinear. Taking the real power transmission losses as an example, the mathematical expression thereof can be stated as (Deeb & Shahidepour, 1990):

$$P_{Loss} = \sum_{k=1}^{N_L} G_k \left[V_i^2 + V_j^2 - 2V_i V_j \cos\theta_{ij} \right] \tag{2.1}$$

where the symbols are defined as follows:

G_k *series conductance of branch k*
N_L *number of branches in the network*
P_{Loss} *total power transmission losses*

V_i, V_j *voltage magnitude at buses i and j*
θ_{ij} *phase angle of* ij^{th} $Y-$*matrix component*

It can be deduced from Equation (2.1) that the expression for the real power transmission losses is both nonlinear and nonconvex, being quadratic in terms of the bus voltage magnitudes, in addition to having trigonometric function components. The inherent difficulty of evaluating a nonlinear objective function of this nature has motivated the devising of alternative (i.e. simpler) formulations of the objective function, chiefly by means of linearization. Thus, a number of linear objective functions for the loss minimization-based VVO problem have been proposed in the literature, for example (Mangoli *et al.*, 1993; Iyer *et al.*, 1983).

As for the decision or control variables for the VVO problem, these can be classified into those derived from voltage-regulating devices, and those derived from reactive power sources, as has been briefly presented in section 2.2. Voltage regulation is mainly through synchronous generator terminal voltage magnitude adjustments and

Under-Load Tap-Changing (ULTC) transformers. Reactive power injection/consumption devices are synchronous generators, synchronous condensers, shunt capacitors and reactors, Flexible AC Transmission System (FACTS) devices, and Distributed Generation to the extent that is dependent on the specific technology (Padullaparti *et al.*, 2016). Some of these devices generate continuous variables, others discrete variables. A complete and most accurate formulation of the VVO problem would thus be a Mixed Integer Nonlinear Programming (MINLP) problem formulation (Rabiee & Parniani, 2013).

2.3.2. Constraints of the Volt/VAR optimization problem

The constraints of the Volt/VAR optimization problem essentially consist of limits on the permissible range of values for the control variables (e.g. transformer tap limits, shunt capacitor range), operating limits on the power system state variables (e.g. generator real and reactive power outputs, voltage magnitudes and phase angles, line and transformer flow limits, etc.) (Momoh, 2001). The standard set of constraints considered in most formulations of the VVO problem can be stated as:

Network power balance equations:

$$P_i(V,\delta,\tau) - P_{Gi} + P_{Li} = 0 \tag{2.2}$$

$$Q_i(V,\delta,\tau) - Q_{Gi} \pm Q_{Si} + Q_{Li} = 0 \tag{2.3}$$

where (expressed in polar form):

$$P_i(V,\delta,\tau) = V_i \sum_{j=1}^{N_{Li}} V_j Y_{ij} \cos(\delta_{ij} - \theta_{ij})$$

$$Q_i(V,\delta,\tau) = V_i \sum_{j=1}^{N_{Li}} V_j Y_{ij} \sin(\delta_{ij} - \theta_{ij})$$

Control variable limits:

$$V_{Gi}^{\min} \le V_{Gi} \le V_{Gi}^{\max} \tag{2.4}$$

$$\tau_k^{\min} \le \tau_k \le \tau_k^{\max} \tag{2.5}$$

$$Q_{Si}^{\min} \le Q_{Si} \le Q_{Si}^{\max} \tag{2.6}$$

State variable constraints:

$$P_{G1}^{\min} \le P_{G1} \le P_{G1}^{\max} \tag{2.7}$$

$$Q_{Gi}^{\min} \le Q_{Gi} \le Q_{Gi}^{\max} \tag{2.8}$$

$$V_{Li}^{\min} \le V_{Li} \le V_{Li}^{\max} \tag{2.9}$$

$$|S_k|^2 \le \left(S_k^{\max}\right)^2 \tag{2.10}$$

The symbols in the above expressions have the following definitions:

$P_i(V,\delta,\tau)$ *active power injection at bus i*
P_{Gi} *generator active power output at bus i*
P_{G1} *generator active power output of slack bus*
P_{Li} *active power demand at bus i*
$Q_i(V,\delta,\tau)$ *reactive power injection at bus* i
Q_{Gi} *generator reactive power output at bus i*
Q_{Li} *reactive power demand at bus i*
Q_{Si} *reactive power source/sink magnitude at bus i*
V_{Gi} *generator terminal voltage magnitude at bus i*
V_{Li} *voltage magnitude at PQ bus i*
τ_k *tap position of ULTC connected in branch k*
S_k *apparent power flow in branch k*
Y_{ij} ij^{th} *component of admittance matrix*
δ *voltage phase angle*
N_{Li} *number of branches connected to bus i*

The set of constraints given by Equation (2.2) to (2.10) defines the feasible region for the VVO problem, and a solution for the problem (i.e. a set of control variables that minimizes Equation (2.1)) is admissible only if it is feasible with respect to the constraint set. It can be observed that this constraint set is nonlinear and non-convex, because the constraint Equations (2.2) and (2.3), for example, have trigonometric terms, and Equations (2.4), (2.7) and (2.9) are non-convex quadratic (Frank & Rebennack, 2016; Cain *et al.*, 2012). Moreover, some control variables (specifically ULTC tap positions and shunt reactive power sources, represented by Equations (2.5) and (2.6) respectively) can only take on discrete values. This gives the constraint set (indeed the overall problem formulation) for the VVO problem the characteristic of being highly nonlinear (Huneault & Galiana, 1991), and poses special challenges for any solution algorithm that may be applied to solve the problem.

2.3.3. Brief summary of Volt/VAR optimization problem formulation

Volt/VAR optimization problem formulation involves the specification of the objective function, the decision or control variables, and the constraints to be satisfied by the optimal solution to the problem. The problem formulation reflects the ways in which Volt/VAR optimization impacts the secure, economical and efficient operation of the power system. The literature review that has been conducted reveals that power loss minimization, voltage profile improvement, voltage security maximization, and control effort minimization are among the most important objectives of the Volt/VAR

optimization problem. The achievement of these objectives needs to take into account the functional and operational constraints that need to be maintained in the course of the operation of the power system. The modelling of the Volt/VAR optimization problem is further discussed in chapter 3 of this thesis.

In the following section, a detailed discussion of the variety of solution approaches that have been applied over the decades to the VVO problem is presented, based on the surveyed literature.

2.4 Optimization methods for the Volt/VAR optimization problem

The various approaches that have been proposed over the years for the solution of the VVO problem may be taken to fall into two main categories: classical/conventional methods, and heuristic/intelligent search-based techniques. The merit of any candidate solution approach can be gauged on the basis of its ability to address the performance characteristics relevant to the VVO problem, among them being (in no particular order of importance) (Frank & Rebennack, 2012a; Momoh, 2001):

- Accuracy requirement of problem formulation
- Computation time and memory requirements
- Possibility for real-time implementation
- Scalability of solution approach
- Global convergence characteristics
- Global optimality characteristics
- Reliability of solution
- Robustness of solution method
- Ability to handle both continuous and discrete decision variables
- Ability to (simultaneously) address multiple objectives
- Simplicity of solution method

Model accuracy is a very important consideration in an optimization problem, from the perspective of the accuracy (and usability) of the obtained solution, as well as the complexity of the optimization problem, which has a bearing on the choice of the solution algorithm for application to the problem (Wood *et al.*, 2014). Indeed, different solution algorithms require different levels of accuracy (or detail) of the problem formulation. With VVO being an operational optimization problem, speed of computation is also an important consideration, especially in the context of real-time implementation, where control decisions need to be generated quickly in response to dynamic system variations so as to maintain the reliability of system operation. Similar observations can be made about each of the other performance characteristic requirements of solution approaches for the VVO problem outlined above. A more

detailed discussion of desirable performance characteristics of optimization algorithms can be found in Nocedal and Wright (2006).

It is evidently hardly practical to find a single solution algorithm that effectively addresses all of the performance characteristics listed above, in part due to the inherent mutual conflict that they may exhibit. Commonly, the various solution algorithms are differentiated by how well they address some (and not necessarily all) of these requirements. In the following sub-sections, some of the solution algorithms that have been proposed in the literature are discussed under the two main categories as stated earlier (i.e. classical/conventional methods, and heuristic/intelligent search-based techniques).

2.4.1. Classical/conventional methods for Volt/VAR optimization

A wide variety of solution methods falling under the category of classical/conventional optimization techniques have been applied to the VVO problem, among them being first-order and second-order gradient-based methods, Quadratic Programming (QP), Linear Programming (LP), Interior-Point Methods (IPM), and Mixed-Integer Programming (MIP), along with decomposition techniques. The block diagram in Figure (2.11) depicts the classical optimization methods reviewed in this section.

Gradient-based methods are iterative optimization techniques that seek to extremize (i.e. minimize or maximize) a differentiable nonlinear function by generating a sequence of improving estimates of the decision vector, moving in such a direction as to achieve progressively lower values (in the case of minimization) of the objective function, until the sequence hopefully terminates at the solution (i.e. the minimum of the objective function to be optimized) (Nocedal & Wright, 2006). Some of the earliest efforts to algorithmically solve the VVO problem applied gradient-based methods, examples of which can be found in (Hano *et al.*, 1969; Peschon *et al.*, 1968; Dommel & Tinney, 1968).

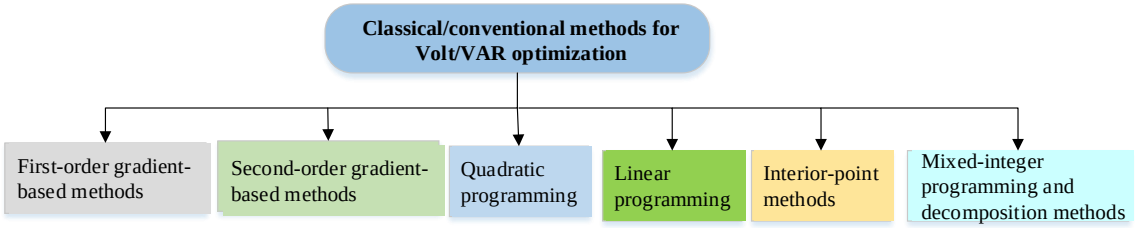

Fig. 2.11: Classical/conventional methods for Volt/VAR optimization reviewed in this section

2.4.1.1. First-Order Gradient-Based Methods

The principal first-order gradient-based methods that have been applied to the solution of the VVO problem are the Reduced Gradient (RG) (Bhatele *et al.*, 1985;

Fernandes *et al.*, 1980), Generalized Reduced Gradient (GRG) (Yu *et al.*, 1986), and Conjugate Gradient (CG) (Hano *et al.*, 1969) methods.

In the *Reduced Gradient* (RG) method, first applied to the OPF problem by Dommel and Tinney (1968), the functional and equality (i.e. power flow equation) constraints are handled by means of penalty terms and Lagrangian multipliers respectively, forming a linear combination with the objective function to construct the Lagrangian function, to which the Karush-Kuhn-Tucker (KKT) conditions are then applied to solve the minimization problem. The RG method provides a way to reduce the problem size, where the problem variables are divided into decision variables and state variables, the objective function expressed as a function of the decision variables, while the state variables are adjusted to maintain solution feasibility (Frank & Rebennack, 2012a).

The *Generalized Reduced Gradient* (GRG) method is an extension of the RG method that allows for the direct handling of nonlinear and inequality constraints. Inequality constraints are turned into equality constraints by the introduction of nonnegative slack variables, and the (nonlinear) constraints are then linearized about the operating point. The generalized reduced gradient is then defined as the gradient of the linear combination of the objective function and the linearized constraints (Rao, 1996). Each such linearization is treated as a subproblem, which can be solved by a gradient-based method such as the RG method, and a series of such subproblem solutions should lead to the solution of the original problem. The GRG method was applied by Yu *et al* (1986) to the solution of a variety of optimal power flow problems, chiefly power loss minimization and network voltage profile optimization. Some of the attractive features of the GRG method are the avoidance of penalty terms in dealing with the functional constraints, the convenient way it provides for transforming a nonlinear constrained optimization problem into an unconstrained one that can be solved by a gradient-based method, and the reduced dimensionality of the resulting problem (de Carvalho *et al.*, 2008).

The *Conjugate Gradient* (CG) method was proposed in the 1950s as an iterative method for solving linear systems with symmetric positive definite matrices (Hestenes & Stiefel, 1952), offering an alternative to existing methods such as Gaussian elimination, and especially well-suited to solving large-scale problems. Extension of the method to the application to nonlinear problems was developed in the 1960s (Fletcher & Reeves, 1964), and constituted one of the earliest known methods for solving large-scale nonlinear optimization problems (Nocedal & Wright, 2006). The nonlinear CG method was applied by Hano *et al* (1969) to the minimization of the node voltage magnitude deviations from their nominal values. The conjugate gradient

vector (which establishes the search direction in the CG method) is computed as a linear combination of successive previous search directions. This method of constructing the search direction ensures non-interference of consecutive search directions, consequently leading to greater advance of the algorithm towards the solution. Key features of the CG method are low storage requirements, and more rapid advance towards the solution relative to the steepest gradient method.

First-order gradient-based methods offer a reliable and fairly unsophisticated way to optimize a differentiable nonlinear function, without being computationally expensive. Their main drawback is the slow rate of convergence, as they rely solely on first-order information of the function to be optimized in advancing toward the solution. The second-order methods, discussed in the next sub-section, constitute an improvement in this aspect.

2.4.1.2. Second-Order Gradient-Based Methods

Second-order methods differ from first-order methods chiefly in the construction of the search direction for the optimization algorithm. Whereas first-order methods rely solely on the first-order (partial) derivatives of the objective and constraint functions, second-order methods additionally incorporate second-order information. The second-order partial derivatives carry the function's curvature information, and incorporation of this information leads to faster convergence of the algorithm to the solution. Newton's method, the representative second-order gradient-based method, applies a second-order Taylor series expansion to the objective function about the current iterate x_k, which leads to the search direction d_k at k being defined by $d_k = -\left(H(x_k)\right)^{-1}\nabla f(x_k)$, where $H(x_k)$ is the Hessian matrix (i.e. the matrix of second-order partial derivatives of the objective function), and $\nabla f(x_k)$ is the vector of first-order derivatives of the objective function at k. Examples of the application of Newton's method to the VVO problem can be found in (de Sousa *et al.*, 2003; da Costa, 1997; Bjelogrlic *et al.*, 1990).

The distinguishing feature of second-order methods is their quadratic rate of convergence, much faster than the convergence rate of first-order methods, although this comes at the expense of additionally having to compute the inverse of the Hessian matrix, which may be a cumbersome, error-prone, and computationally expensive process, especially in the case of problems with a dense Hessian matrix. An alternative is Quasi-Newton methods, which avoid the exact computation of the Hessian matrix by approximating it using information about the change in the first-order derivatives (Nocedal & Wright, 2006). Two other issues with second-order methods are the need for the Hessian matrix to be positive definite to ensure the

search direction is a descent direction, and the difficulty in dealing with the inequality constraints of the VVO problem (da Costa, 1997; Happ, 1977). The reliability of Newton's method particularly requires that the difference between the objective function and its 2nd-order approximation at the current iterate not be too large. Despite these issues, Newton's method is not only a classical method for nonlinear optimization, but also represents an important optimization approach, both efficient and robust for a large class of problems (Bjelogrlic *et al.*, 1990).

2.4.1.3. Quadratic Programming

Quadratic Programming (QP) is a special case of nonlinear programming in which the objective function is quadratic and the constraint set is linear. When applied to the VVO problem, a technique known as sequential quadratic programming (SQP) is employed, involving iteratively generating a quadratic approximation of the objective function, and linearizing the constraints about the current operating point. The solution of these QP subproblems should converge to the optimal solution of the original nonlinear problem (Bazaraa *et al.*, 2006). Quadratic programming is somewhat of a compromise between the general nonlinear programming problem and a linear programming formulation, trying to achieve some balance between the accuracy of the model representation and the computational complexity of the solution of the problem (Quintana & Santos-Nieto, 1989).

Depending on whether the QP model formulation is convex or nonconvex, a variety of solution techniques exist, among them being active set methods and interior point methods. Examples of the QP model formulation of the VVO problem can be found in (Grudinin, 1998; Quintana & Santos-Nieto, 1989; Nicholson & Sterling, 1972).

Quintana and Santos-Nieto (1989) developed a convex QP formulation for the real power loss minimization reactive power dispatch problem, and solved by it the active-set projection method. Nicholson and Sterling (1972) solved the real and reactive power dispatch problem by quadratic programming, considering a quadratic cost function for the generation and transmission line losses, and a linear approximation of the system constraints. An improvement in accuracy was obtained over the linear programming-based model, and faster solutions compared to the exact nonlinear model of the combined active/reactive power optimization problem. Grudinin (1998) also applied SQP to reactive power optimization, and developed a quadratic multi-objective optimization problem, combining economic and security objectives, which was also solved by the Newton-based active-set method.

The attractiveness of the quadratic programming solution technique lies in its providing a means to achieve a good balance between the requirements of a

reasonably accurate model of the VVO problem, and the computational expense associated with the exact nonlinear model formulation. The quadratic approximation of the nonlinear system power loss function is sufficiently accurate, and permits the application of efficient QP solution techniques to the problem (Grudinin, 1998; Quintana & Santos-Nieto, 1989).

2.4.1.4. Linear Programming

An optimization problem is classified as a linear programming (LP) problem when both the objective function and the constraint set are linear functions of the decision variables. Because the VVO problem is inherently nonlinear (as discussed in section 2.3), an LP formulation of the problem entails the linearization of both the objective function and the constraint set (section 2.3, Equations (2.1) and (2.2) to (2.10) respectively). As pointed out in section 2.3, linearization is typically performed around some desired operating point, and can be done on the basis of the first-order Taylor series expansion (i.e. taking the first-order partial derivatives of the nonlinear power loss function with respect to the control variables) (Deeb & Shahidepour, 1990), or on the basis of sensitivity relationships devised to relate changes in the state variables to changes in the control variables (Iyer *et al.*, 1983; Narita & Hamman, 1971; Hano *et al.*, 1969).

Linear programming has traditionally been a popular approach for the solution of the OPF problem, which includes economic dispatch and reactive power dispatch (Zhu, 2009). The approach has many desirable characteristics, such as reliability, very good convergence properties even for large-scale problems, faster computational speed, and availability of very efficient algorithms for solving the problem (Nocedal & Wright, 2006). The main solution techniques for the LP problem are several variants of the Simplex method, and Interior Point Methods (IPM).

Examples of LP formulations of the VVO problem are to be found in (Chebbo *et al.*, 1992; Mota-Palomino & Quintana, 1986; Mamandur & Chenoweth, 1981; Hobson, 1980; Kishore & Hill, 1971). Chebbo *et al* (1992) devised an LP formulation for the reactive power dispatch problem incorporating voltage stability, to minimize the risk of voltage collapse in the system, and solved it by the dual revised simplex method. The linearization and solution of the problem was done in an iterative manner, leading to what is commonly referred to as sequential linear programming (SLP). The desirable characteristic of this technique that has been highlighted is convergence of the solution that is independent of the problem size, whereas in the case of the original nonlinear problem formulation, depending on the solution algorithm, global convergence may not be guaranteed (Frank *et al.*, 2012a). An LP-based network-constrained reactive power control problem is presented by Hobson (1980), which is

found to be suitable for real-time application in large-scale power systems, with speed of solution and convergence characteristics that are difficult to achieve in the case of the classic nonlinear formulation of the problem.

With all the desirable characteristics of the LP approach to VVO, it must be borne in mind that this comes along with the compromise of the accurate representation of the otherwise highly nonlinear model of the VVO problem. Efforts to devise more efficient solution techniques for the original nonlinear problem formulation have thus continued to attract a lot of attention (Zhang et al., 2007).

2.4.1.5. Interior-Point Methods

Interior point methods (IPM) are a class of optimization techniques that were initially developed as an alternative to the Simplex method for solving linear programs (Forsgren *et al.*, 2002), with the introduction of Karmakar's method (Karmarkar, 1984), a polynomial-time linear programming method. Whereas the Simplex method exploits the convexity of the feasible region of the LP problem, searching along the vertices of the polytope that defines the feasible region for the optimal solution to the problem, IPMs take a different approach, attempting to confine the search path within the feasible region, and establishing and following a "central path" towards the optimal solution of the problem. Besides having pseudo-polynomial complexity, IPMs also exhibit some advantages relative to the Simplex method, such as being especially efficient for large-scale problems, and making more rapid advance towards the optimal point (Wright, 1997). The successes of IPMs in LP incited research efforts to extend them to general nonlinear problems, and these methods have attractive properties that make them especially suitable for nonlinearly constrained optimization, such as the efficient handling of inequality constraints (which is quite problematic for the classical Newton-based methods), rapid convergence, and not having to start from a strictly feasible initial solution (Capitanescu *et al.*, 2007).

Examples of the application of interior-point methods to reactive power optimization and voltage control can be found in (Ding *et al.*, 2000; Torres & Quintana, 1998; Granville, 1994). Granville (1994) applied the primal-dual logarithmic-barrier interior point algorithm (PDIPM) to the solution of the optimal reactive power dispatch (ORD) problem. It is highlighted in the paper that ORD is a large-scale highly nonlinear, nonconvex optimization problem, and the characteristics of the chosen IPM that make it suitable for application to this problem are the insensitivity of the problem complexity (i.e. number of iterations required to reach to solution) to the problem size, more efficient handling of the nonlinear inequality constraints, and numerical robustness, even ability to handle large-scale, ill-conditioned problems. Ding *et al* (2000) used a version of the PDIPM method known as predictor-corrector PDIPM

(PC-PDIPM) to solve the reactive power optimization/voltage control problem, a method which seeks to improve the search direction at each iteration (Mehrotra, 1992). The authors focused in the development of their algorithm on computational speed to be suitable for real-time application, reliability to converge even from an initially infeasible starting point, and the effective detection and handling of infeasibility. Comparison with quadratic programming and least squares-based infeasibility handling showed that the developed PDIPM method scaled better with the number of constraints (i.e. increase in number of constraints having less impact on computational speed), and the infeasibility detection and handling approach taken added much less computational burden to the overall optimization process. Interior point methods have thus been found to be very suitable for solving the large-scale, highly nonlinear constrained OPF problems, an example of which is the VVO problem (Torres & Quintana, 1998).

Due to the many appealing characteristics of interior-point methods as established from the conducted literature review, the primal-dual interior-point method (PDIPM) has been selected as the classical optimization algorithm that will form the basis for the design and implementation of the Volt/VAR optimization algorithm in this research. This work is presented in detail in chapters 4 and 5 of the thesis.

2.4.1.6. Mixed Integer Programming and Decomposition Techniques

In all the solution algorithms discussed thus far, only continuous control variables are considered in the formulation of the VVO problem. However, as pointed out in section 2.3, the presence of discrete control variables (e.g. transformer tap positions in ULTC transformers) makes the general formulation of the VVO problem a mixed integer programming problem, implying that integrality constraints have to be enforced on a subset of control variables. The motivation for considering a continuous approximate formulation of the problem has been the very high computational expense associated with the full mixed integer nonlinear programming (MINLP) problem, especially in large-scale systems with possibly thousands of mixed integer and continuous variables, and nonlinear objective and constraint functions (Feng & Peterson, 2010). The drawback of this approach is that achieving feasibility of the continuous solution by rounding off the values of the control variables required to be integral to the nearest integer values may in many cases be difficult. Moreover, the objective value of the rounded-off solution may deviate significantly from that of the continuous optimal solution (Rao, 1996). These considerations, along with the advances in computational capabilities of modern computers that have enhanced the tractability of this class of problems, have encouraged the search for effective algorithms to treat the full MINLP (Hemmecke *et al.*, 2009).

The most commonly used optimization algorithm for MINLP is the branch-and-bound (B&B) algorithm, developed by Land and Doig (1960) to solve integer linear programming problems, which was subsequently extended to the solution of MINLP problems. In the B&B method of solving MIP problems, an indirect approach is taken where firstly a continuous version of the problem is optimized by relaxing the integrality constraints (thus obtaining a continuous optimal solution), then each of the integrality constraints is progressively enforced until an integer optimal solution is found. The key components of the algorithm are branching, where for a given continuous optimal solution the associated integer feasible solutions are evaluated for optimality; and bounding, where the prevailing integer optimal value is used as an upper bound to eliminate from further consideration any alternatives that cannot possibly achieve a better optimal solution.

Examples of MINLP formulations of the VVO problem can be found in (Ahmadi *et al.*, 2015; Rabiee & Parniani, 2013; Saric & Stankovic, 2009; Mehra, 1994) [57], [112]-[114]. Due to the need to simultaneously treat both continuous and discrete variables, it is common to apply decomposition techniques in solving mixed-integer programming problems. The problem is reformulated into two separable optimization sub-problems, a continuous one and a discrete one. The two sub-problems are then solved alternately, and related together by some decomposition technique, such as Benders decomposition (Geoffrion, 1972), one of the most commonly applied decomposition methods to which the MINLP problem is very amenable. Such an approach has been used by Mehra (1994), for example, where Benders decomposition is used along with the B&B algorithm to solve the combined reactive power planning and real-time voltage control (or reactive power dispatch) problem formulated as a MINLP problem.

The main advantage of solving the VVO problem as a mixed-integer programming problem is the greater accuracy of problem formulation and resulting optimal solution that can be achieved, enabling the accurate modeling of all control devices involved in the problem, including discrete ones such as shunt capacitors, ULTC transformers, and a variety of FACTS devices (Mehra, 1994). This comes at the cost of greater computational complexity of the problem, however. Improvement of algorithms geared towards this class of problems and such paradigms as parallel computing can enhance results that are achievable using this approach of solving the VVO problem.

2.4.1.7. Brief summary of classical/conventional methods for Volt/VAR optimization

Classical/conventional optimization methods furnish a broad arsenal of techniques for solving a wide variety of optimization problems, and have been extensively applied to

the Volt/VAR optimization problem. The key characteristic they all share is their implementation of a mathematically rigorous and systematic iterative procedure in the search for the optimal solution to the optimization problem within the feasible space. Based on the reviewed literature, they do differ, however, in key performance metrics, such as accuracy, speed, reliability, convergence characteristics, and effectiveness of handling inequality constraints and discrete variables, among other criteria. A summary of these characteristics as they apply to each of the discussed methods is presented in Table (2.3).

Table 2.3: Summary of main characteristics of conventional optimization techniques

Technique	Operating principle/main characteristics	Main positive attributes	Main deficiencies
First-order gradient-based methods (RG, CG, GRG)	○ Iterative search based on 1st-order gradient of objective ○ Constraints handled by adding penalty terms to objective function to form Lagrangian ○ GRG uses successive constraint set linearization and slack variables to handle inequality constraints	○ Earliest approaches to algorithmically solve OPF ○ Unsophisticated, reliable, at moderate computational expense ○ Suitable for large-scale application (e.g. CG, due to low memory requirement)	○ Slow convergence rate; RG susceptible to zig-zag behaviour close to optimal point ○ Difficulty handling inequality constraints ○ Need for smoothness of objective function ○ Can only find local optima
Second-order gradient-based methods (Newton, Quasi-Newton)	○ Approach similar to 1st-order methods, with additional incorporation of 2nd-order derivative information of objective function ○ Newton's algorithm is the representative method under the category ○ Quasi-Newton methods approximate the 2nd-order derivative information to reduce computational expense	○ Very effective methods; addition of 2nd-order derivative information significantly improves convergence rate ○ Efficient and robust for a large class of problems, under some mild assumptions (e.g. sufficient accuracy of quadratic approximate model in vicinity of solution)	○ Similar issues as those of 1st-order methods, except having higher convergence rate ○ Convergence requires Hessian matrix to be positive semidefinite ○ Computationally more expensive than 1st-order methods
(Sequential) Quadratic Programming (SQP)	○ Special case of NLP with quadratic objective function and linear constraint set ○ Involves iterative quadratic and linear approximation of objective and constraint functions respectively about the current iterate	○ Achieves good balance between model accuracy and computational expense (accuracy of QP and speed of LP) ○ Efficient solution techniques available that can solve the QP formulation effectively and reliably	○ Has similar drawbacks as those outlined above of gradient-based methods ○ Convergence requires Hessian matrix of quadratic approximation to be positive semidefinite
Linear Programming (LP)	○ Both objective and constraint functions are linear ○ For VVO problem, this entails successive linearization of objective and constraint functions about the current operating point	○ Reliable, good convergence characteristics, even for large-scale problems ○ Fast computational speed ○ Availability of efficient solution techniques (typically Simplex and Interior-Point Methods)	○ Loss of accuracy due to linearization may lead to solution that's not only non-optimal, but perhaps even infeasible for original nonlinear problem ○ For successive linear formulation, solution is only locally optimal
Interior-Point Methods (IPM)	○ Make use of path-following techniques that confine the search path within the feasible region ○ Initially developed for application to LP as alternative to Simplex method; been extended to the treatment of NLP problems	○ Very effective and efficient, especially for large-scale problems, both linear and nonlinear ○ Rapid convergence, more effective constraint handling than classical gradient-based methods ○ Better handling of infeasibility	○ Reliability concerns for particularly difficult problems ○ Combines mathematical rigour with some heuristics; proper parameter selection (e.g. barrier parameters) can be challenging, affecting effectiveness and convergence properties
Mixed-Integer Programming (MIP)	○ Explicitly considers both continuous and discrete variables, thus a more accurate model for the VVO problem	○ A more accurate solution is achieved without the need to round off values of	○ Computationally more expensive, and requires sophisticated techniques to

	○ Decomposition techniques made use of to treat continuous and discrete portions as subproblems ○ Branch-and-bound the main solution technique for MINLP formulations	discrete variables so as to enforce integrality constraints, which may lead to non-optimality, even infeasibility relative to the exact formulation	simultaneously handle both continuous and discrete variables ○ Issues similar to classical gradient-based methods of difficulty of inequality constraint handling, only locally optimal

Collectively, the class of conventional optimization methods suffer from a number of significant deficiencies or drawbacks, notably the inherent difficulty of handling discrete variables, the requirement for the (nonlinear) objective and constraint functions to be smooth (i.e. for the gradient-based methods), and the difficulty of handling nonconvexity in nonlinear problems (meaning they can only find local optimal solutions) (Frank *et al.*, 2012a). Heuristic or intelligent search-based optimization methods have been extensively explored as an alternative approach to solving the VVO problem, as they possess characteristics that can be exploited particularly in addressing some of the deficiencies of the conventional optimization methods that have been highlighted above. The following section presents a review and discussion of this class of optimization methods.

2.4.2. Heuristic/intelligent search-based methods for Volt/VAR optimization

Heuristic/intelligent search-based optimization techniques employ a variety of optimum-seeking strategies that are distinctly different from the approaches taken in conventional optimization algorithms. The search strategies employed in these techniques are meant to overcome many of the deficiencies of the conventional optimization problems, such as the local (rather than global) nature of the search, the limited ability to handle combinatorial problems with discrete decision variables, and the requirement for smoothness of the objective and constraint functions for gradient-based methods, among other factors (Frank *et al.*, 2012b). Over the past few decades a wide variety of these heuristic optimum-seeking techniques have been developed. A representative sample of them are discussed in this section, as they have been applied to the VVO problem, particularly Genetic Algorithms (GA), Evolutionary Programming (EP), Particle Swarm Optimization (PSO), Fuzzy Set Theory, and Expert Systems (ES). The main distinctive characteristics of each technique are briefly discussed, and a sample of applications is also given. Figure (2.12) depicts the heuristic optimization methods reviewed in this section.

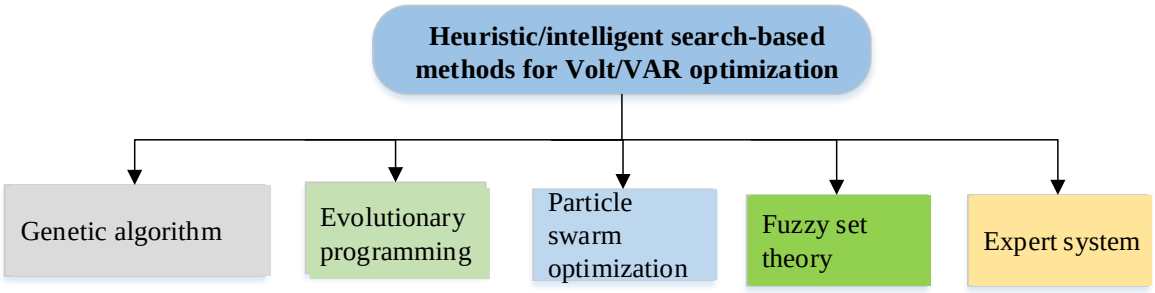

Fig. 2.12: Heuristic/intelligent search-based methods for Volt/VAR optimization reviewed in this section

2.4.2.1. Genetic Algorithm

Genetic algorithm (GA) is a population-based search algorithm that is modelled after the processes of natural selection and natural genetics, combining the features of survival of the fittest in a population of optimal solution candidates, efficient exploitation of historical information, and randomized information exchange among the population candidate solutions so as to evolve the population into a new generation of improved candidate solutions (Goldberg, 1989). The development of GAs was inspired by the robustness, efficiency and efficacy through adaptation observed in biological processes, and efforts were made to develop artificial software systems that could mimic and replicate the natural processes responsible for these characteristics, such as selection, crossover and mutation (De Jong *et al.*, 1997; Holland, 1975). GAs, though conceptually and computationally simple, constitute an efficient, effective and robust approach to search for optimal solutions to a variety of problems in diverse environments, with no reliance on such limiting assumptions of conventional optimization methods as continuity, existence of derivatives, and unimodality (Choden *et al.*, 2022). Once an initial population of candidate solutions is generated, either randomly or heuristically, the population is evolved through the sequential and iterative application of the selection, crossover and mutation operations, into a new generation of improved solution candidates (Alves da Silva & Falcao, 2008).

A number of works have applied GAs to the VVO problem, examples of which can be found in (Padilha-Feltrin *et al.*, 2015; Katuri *et al.*, 2012; Subbaraj & Rajnarayanan, 2009; Lee *et al.*, 1995; Iba, 1994; Haida & Akimoto, 1991). Haida and Akimoto (1991) emphasize the property of GAs of being domain-independent search mechanisms, which provide powerful search characteristics for large, complex search spaces without requiring full knowledge of the problem domain. Iba (1994) proposes an alternative crossover method and incorporation of stochastic “if-then” rules (akin to expert systems) into the GA applied to reactive power planning, so as to enhance the algorithm’s efficiency and effectiveness. A hybridized GA is considered by Lee *et al*

(1995) for the solution of the reactive power operation and planning problem, combining it with successive linear programming (SLP), and using a new population selection and generation method that makes use of Benders' cuts, in an effort to combine the positive characteristics of deterministic and non-deterministic optimization techniques.

Many appealing characteristics have been highlighted in the literature that make the genetic algorithm an effective search mechanism. The effectiveness, however, is also a function of the algorithm design, including choices regarding the selection, crossover and mutation operations, the encoding of the candidate solutions used, and the fitness function (Alves da Silva & Falcao, 2008).

2.4.2.2. Evolutionary Programming

Evolutionary Programming (EP) was conceived and developed by L.J. Fogel in the early 1960s as an alternative approach to realizing artificial intelligence (AI), utilizing the concepts of Darwinian evolution to iteratively generate increasingly appropriate solutions to a given optimization problem (Porto, 1997). It can be seen as an approach to optimization that makes use of simulated evolution to evolve a set of solutions (or organisms) which exhibit increasing intellect evidenced by ability to make correct predictions, to translate those predictions into suitable actions, and to adapt behaviour so as to meet specific goals in a range of environments (De Jong *et al.*, 1997). As an evolutionary algorithm, EP employs the key concept of selection-by-fitness, which entails the generation of a population of candidate solutions (to an optimization problem), devising a suitable fitness function with which to evaluate the worth of each candidate solution in light of stated objectives, and application of evolutionary operators such as mutation to evolve the population through generations of ever-improving candidate solutions (Miranda, 2008). The selection of candidate solutions to propagate through to the next generation can be either elitist (the best in each generation are selected to form the next one) or by stochastic tournament (probabilistic selection of next-generation candidate solutions).

EP has been used as the solution algorithm to the VVO problem in (Abido & Bakhashwain, 2003; Gomes & Saavedra, 1999; Wu & Ma, 1995). The global search characteristics of the EP algorithm, and the non-reliance on the smoothness and/or convexity properties of the objective and constraint functions for effective search, are highlighted by Wu and Ma (1995) as making it suitable for solving the reactive power optimization and voltage control problem, which is highly nonlinear and nonconvex. By maintaining a population of candidate solutions at each iteration, which are propagated through future generations using probabilistic transition rules as a function of their overall merit, with a Gaussian relationship between parents and

offspring, the EP algorithm is able to move over hills and valleys of the search space, and therefore arrive at the globally optimal solution. In (Gomes & Saavedra, 1999), enhanced evolutionary algorithms (evolutionary programming and evolutionary strategies), the enhancement consisting in use of alternative mutation strategies, have been applied to the solution of the reactive power dispatch problem, demonstrating that enhancements can be made to the standard algorithm to improve its efficiency and effectiveness. The effectiveness of population-based evolutionary algorithms in finding pareto-optimal solutions in multiobjective optimization has been pointed out by Abido & Bakhashwain (2003), who have developed a multiobjective evolutionary algorithm for the optimal reactive dispatch problem.

The main evolutionary operations used in EP are mutation, competition and reproduction. As with other evolutionary algorithms, parameter selection plays a key role in exploiting the various desirable attributes of the algorithm, and ensuring its efficiency and effectiveness.

2.4.2.3. Particle Swarm Optimization

Swarm intelligence is a stream of AI research that got established in the early 1990s, based on the study of the swarm behaviour of natural creatures, in terms of how decision making of the individual is influenced by both the individual's experience and the experiences of others (Colorni *et al.*, 1991; Reynolds, 1987; Boyd & Richerson, 1985). Particle swarm optimization (PSO), one variant of swarm intelligence techniques that has become prominent, was developed by Eberhart and Kennedy (Kennedy & Eberhart, 1995), and is based on the analogy of swarms of birds and fish schooling. The algorithm uses a population of particles exploring the search space in search of the optimal solution to an optimization problem. Associated with each particle is a position and a velocity in a two-dimensional search space, and the change in position of the particles as a function of the current best positions of the individual and of the overall population is what constitutes the population's evolution towards the optimal point. The use of a population of candidate solutions, incorporating randomness and memory, as well as diversification at the beginning, and intensification towards the end of the search, adds greatly to PSO's efficiency as a search mechanism (Fukuyama, 2008).

Having been originally developed to treat nonlinear optimization problems with continuous variables, a number of enhancements to the standard PSO algorithm have been proposed and developed, to improve the algorithm's efficiency, and to extend its applicability to other problems (e.g. combinatorial optimization, and mixed-integer nonlinear programming (MINLP) problems).

Examples of the application of PSO to the solution of the VVO problem can be found in (Vitor & Vieira, 2018; Ferreira *et al.*, 2016; Kaur *et al.*, 2016; Sahli *et al.*, 2014; Grant *et al.*, 2008; Cai *et al.*, 2007). Vitor and Vieira (2018) solve a multi-objective Volt/VAR control problem that considers robustness in addition to power loss minimization using the PSO algorithm, exploiting the ability to structure the algorithm so as to handle multiple objectives (Coello *et al.*, 2004). A modified PSO algorithm has been applied to optimal reactive power dispatch by Cai *et al* (2007), the modification consisting in adding mutation to the standard algorithm in order to improve its global search characteristics and prevent rapid convergence to local optima. Sahli *et al* (2014) follow a different approach to enhancing the global search characteristics of the PSO algorithm, which is to hybridize it with the Tabu Search algorithm, another stochastic search algorithm (Glover & Laguna, 1997). Grant *et al* (2008) solve the reactive power and voltage control problem using the differential evolution (DE) (an evolutionary computation algorithm) (Storn & Price, 1995) and PSO algorithms, and performance comparison of the two methods has been made, particularly in terms of solution quality and convergence characteristics. The authors found the PSO algorithm to slightly outperform the DE algorithm, although exhibiting relatively greater computational effort. Ferreira *et al* (2016) take advantage of the PSO's ability to better handle discrete control variables than the conventional optimization methods, and apply it to the solution of the optimal reactive power dispatch problem considering discrete variables. Kaur *et al*

(2016) focus on how PSO-based optimal reactive power dispatch can enhance system security, considering the impact of intermittent renewable generation such as wind power generation.

Besides being able to address diverse optimization objectives, as can be deduced from the surveyed literature, the PSO algorithm additionally has the desirable characteristic of being quite simple to implement, in the sense that simple rules governing individual agent behaviour can result in sophisticated swarm behaviour (Li & Coster, 2022). The model of each individual agent (or particle) is relatively simple, yet can lead to effective and efficient collective behaviour of the whole swarm in terms of searching for the optimal solution in a search space. Hybridization with other methods, and other enhancements to the standard algorithm, are often considered to improve the efficiency and effectiveness of the algorithm (Gad, 2022).

The PSO algorithm has been selected as the heuristic optimization algorithm to form the basis for the design and implementation of the Volt/VAR optimization algorithm in this research, due to its many desirable characteristics when compared with the other

heuristic techniques. The design and implementation of the PSO-based Volt/VAR optimization algorithm is presented in chapter 6 of this thesis.

2.4.2.4. Fuzzy Set Theory

Although conventional optimization problems are computed with the assumption of precise information, in reality most real-world data that serves as input to the optimization problems is embedded with uncertainty and imprecision. Power systems are especially prone to a significant amount of uncertainty in operational data, largely due to their large scale, being geographically widely distributed, complexity in operational dynamics, and susceptibility to unexpected events (Momoh & Tomsovic, 1995). Fuzzy set theory is a mathematical approach that can be used to capture this uncertainty and imprecision of information, the incorporation into the optimization problem of which can enhance the robustness of the obtained results. Fuzzy set theory enables objective and constraint functions to be represented as fuzzy sets, where the membership to these sets represents the degree of closeness to the optimum (for the objective function) and the degree of enforcement of the constraints (for constraint functions). The maximization of membership functions then implies the simultaneous optimization of the objective function and enforcement of the constraint set, all while taking uncertainties into account. This leads to a better compromised solution, more robust in the sense of being less sensitive to parameter variations (Zhang & Tolbert, 2007).

Fuzzy set theory is not actually an optimization technique, and so it is normally used in conjunction with optimization techniques, where it essentially serves as a tool for modeling uncertainty and imprecision in the problem formulation. Applications to reactive power optimization and voltage control have been many over the years, combining with a variety of optimization techniques, examples of which can be found in (de Souza & Almeida, 2010; Aucharimayet & Sirisumrannukul, 2009; Momoh *et al.*, 2009; Lu & Hsu, 1997; Su & Lin, 1996; Abdul-Rahman & Shahidepour, 1993; Yokoyama *et al.*, 1993; Tomsovic, 1992). De Souza and Almeida (2010) combine fuzzy set theory with a strength-pareto evolutionary algorithm (SPEA2) to solve the multi-objective reactive power/voltage control problem. Tomsovic (1992), Abdul-Rahman and Shahidepour (1993), and Yokoyama *et al.* (1993) formulate a fuzzy-linear programming-based reactive power/voltage control, combining the reliability and speed characteristics of linear programming with fuzzy set theory's ability to more efficiently depict the realistic system objective and constraint functions, leading to a more practical solution of the problem. Other examples of hybrid methods incorporating fuzzy set theory or fuzzy logic are the fuzzy-dynamic programming

approach presented by Lu and Hsu (1997), and the fuzzy-PSO multi-objective algorithm presented by Aucharimayet and Sirisumrannukul (2009).

The strength of fuzzy set theory that has been exploited in reactive power optimization and voltage control (among other power system applications) is the capability of handling ambiguity, conflicting objectives, and soft constraints in a flexible way that can moreover improve computational complexity of power system optimization problems (Momoh & Tomsovic, 1995). By providing the means to effectively model uncertainty and imprecision, and to incorporate the approximate reasoning and subjective judgment of expert operators into the mathematical model, fuzzy set-based modeling facilitates the realization of a better compromised solution, where both accuracy and robustness of the solution are taken into account (Yokoyama *et al.*, 1993).

2.4.2.5 Expert System

Expert systems (also known as knowledge-based systems) constituted one of the earliest approaches to building AI systems in the 1960s, and were among first successful commercial applications of the then nascent field of artificial intelligence (Russel & Norvig, 2010). An expert system (ES) can be defined as an intelligent computer-based system in which representations of human expert knowledge are stored, and it can apply inference procedures and heuristics to this knowledge base to solve complex problems in a manner that a human expert would do. An ES is fashioned after the model of human reasoning, which may be considered to be based on the creation of categories, application of specific (a priori) rules, use of heuristics (i.e. rules-of-thumb, representing conventional wisdom), as well as use of past experience (precedence-based reasoning). Most expert systems make use of rule-based reasoning, the main components of which are the knowledge/rule base containing much of the problem-solving knowledge, a database containing some data of interest to the system, an inference engine generating the decisions, and a user interface providing a means for user interaction with the system (Brown & O'Leary, 1995).

Expert systems are especially applicable to fields such as power system operation, where a wealth of system operational knowledge and expertise has been accumulated, and can be used to build intelligent decision support systems that can aid system operators in making decisions and taking quick action especially under anomalous conditions, where not only correct action, but also speed of execution can be critical in preventing major emergencies (Negnevitsky & Le, 1996).

A number of researchers have built expert systems for optimal reactive power dispatch and voltage control, examples of which are to be found in (Lomi & Limpraptono, 2017; Xiang-jun, 2011; Exposito *et al.*, 1993; Wagner *et al.*, 1990; Cheng *et al.*, 1988; Liu & Tomsovic, 1986). In many of these applications of expert systems to the reactive power/voltage control problem, the emphasis is placed on leveraging human expert knowledge and experience, and historical information to build a system that can quickly provide effective remedial action in emergency conditions, when human operator reaction may be too slow, and conventional optimization methods ineffective (Wagner *et al.*, 1990). Liu and Tomsovic (1986) built an ES that applies empirical rules to generate appropriate controls when slight voltage violations occur, whereas mathematical programming software is used to address more severe contingencies. Cheng et al (1988) developed an ES for reactive power and voltage control based on a sensitivity-tree approach, where the most effective control measures to alleviate abnormal voltage conditions are determined on the basis of the rule base coded into the ES. Scalability and possibility for real-time application are highlighted as the main characteristics of the proposed system. A similar sensitivity-based approach has been used in the ES developed by Exposito *et al* (1993) for reactive power control for voltage profile improvement. In Xiang-jun (2011) makes use of historical information and real-time data to develop an ES for substation voltage and reactive power control. The opportunity to leverage years of operating experience in developing the ES is highlighted as one of the main advantages of this approach. The ES developed by Limo and Limpraptono (2017) is focused on monitoring and improving power system voltage stability. The ES can use the empirical knowledge in the knowledge base to effectively identify the critical load buses most susceptible to excessive voltage violations, and recommend the most effective remedial actions, as an aid to the system operator.

Expert systems present many advantages as intelligent decision support systems where decisions have to be made to solve complex problems, as in the case of reactive power/voltage control under emergency conditions. Notable among these advantages are the opportunity to combine the knowledge and experience of several human experts, accumulated over a period of time, along with historical information, to build an efficient and effective decision-support system, little reliance on precise mathematical models of the system, thus especially effective under anomalous operating conditions, and others such as reproducibility, consistency, and lack of fatigue (which human operators are very susceptible to). Some obvious disadvantages of expert systems are that they lack the human capabilities of common sense, creativity, and learning. There is also the likelihood of gradual degradation of

the system, requiring periodical update of the rule base to remain up to date as the modeled system undergoes any changes (Brown & O'Leary, 1995).

2.4.2.6 Brief summary of heuristic/intelligent search-based methods for Volt/VAR optimization

Heuristic optimization techniques employ a variety of optimum-seeking strategies that differ conceptually from those employed in conventional optimization methods. By and large, these techniques make use of a population of candidate search points, which, coupled with their stochastic nature, generally gives them global search characteristics (that is, the ability to globally converge to a solution where one exists, independently of the initial point, and to find the globally optimal solution, despite nonconvexity of the objective function and the feasible region). They do suffer some drawbacks, however, when compared with the conventional methods, such as lacking mathematical rigor (by virtue of their heuristic nature), being relatively computationally more expensive, and their effectiveness being very dependent on the judicious choice of the algorithm parameters. A summary of the main characteristics of each of the heuristic methods discussed in this section is presented in Table (2.4). The following section presents a brief comparative analysis of the two main classes of optimization techniques that have been presented in this section.

Table 2.4: Summary of main characteristics of nonconventional/heuristic optimization techniques

Technique	Operating principle/main characteristics	Main positive attributes	Main deficiencies
Genetic algorithm (GA)	○ Population-based evolutionary stochastic search algorithm, modeled after mechanics of natural genetics, incorporating crossover, mutation and selection ○ Adaptation through use of genetic operators, enabling use of historical information, and randomized information exchange among population candidates, is what characterizes the search for the optimal point	○ Conceptually and computationally simple, yet efficient, effective and robust search mechanism applicable to various problem classes ○ Not limited by the properties of convexity, smoothness, unimodality of objective function, requirements typical of classical techniques ○ Can achieve global convergence and global optimality	○ Computation time can be long, thus limited scalability (although parallelization is possible to improve computational efficiency) ○ Effectiveness a function of design aspects like encoding, choice of fitness function, and other parameters of the algorithm
Evolutionary Programming (EP)	○ Evolutionary stochastic search algorithm using simulated evolution to evolve a population of candidate solutions of "increasing intellect" in search of optimal point ○ Stresses mutation (rather than crossover, opposed to GA)	○ Capable of global convergence and global optimality, by judicious choice of mutation and selection-by-fitness mechanisms (which can be either elitist or by stochastic tournament)	○ Computationally quite expensive for OPF problems, which typically have thousands of variables and constraints
Particle Swarm Optimization (PSO)	○ Modeled after swarm behavior of natural creatures, where an individual makes decisions based on best own experience and best group experience ○ Population of particles evolved towards the optimal point by modifying each particle's position as a function of its current best and group's best position	○ Simple both conceptually and in terms of implementation; simple rules governing individual particle behavior can result in sophisticated swarm behaviour ○ Many enhancements to the standard algorithm are possible, to enable	○ Convergence properties are highly influenced by parameter selection for the algorithm

		application to a wide variety of problems	
Fuzzy Set Theory-based methods	○ Fuzzy set theory used as a tool for modeling uncertainty present in objective, constraint functions, and system parameters ○ Objective, constraint functions are represented as fuzzy sets; membership to these sets represents the degree of closeness to optimum point and degree of enforcement of constraints	○ Provides a better compromised solution, balancing robustness (i.e. insensitivity to parameter variations) and optimality ○ Furnishes capability to handle ambiguity, conflicting objectives, and soft constraints in a flexible way that can improve computational complexity	○ Not actually an optimization problem, only a way to handle uncertainty and imprecision; thus largely needs to be combined with an optimization technique; this may affect overall complexity and effectiveness, depending on problem formulation and optimization algorithm used
Expert System (ES)	○ Uses a computer-based representation of human expert knowledge, in conjunction with an information database and an inference engine to solve complex problems requiring human expertise and experience ○ Main components are the knowledge base, database, inference engine, and interfaces to the user and to other programs needed to execute the system's functionalities	○ Accumulated wealth of knowledge in the field can be exploited to build an intelligent decision support system to assist system operators respond quickly and effectively especially under anomalous conditions ○ Has desirable attributes of efficiency, reproducibility, consistency, and opportunity for expertise consolidation	○ Lacks many natural strengths of a human expert operator, such as common sense, creativity, adaptability, learning ability

2.5 Comparative analysis of solution approaches for the VVO problem

It is quite evident that the two classes of optimization techniques discussed in the preceding section exhibit diverse characteristics, both in terms of operating principle, as well as strengths and drawbacks when gauged against the desired performance characteristics outlined at the beginning of the section.

The key characteristic of classical/conventional optimization methods is their implementation of a mathematically rigorous and systematic iterative procedure in the search for the optimal solution to an optimization problem within the feasible space. They do differ, however, in key performance metrics, such as accuracy, speed, reliability, convergence characteristics, and effectiveness of handling inequality constraints and discrete variables, among other criteria. Collectively, the class of conventional optimization methods suffer from a number of significant deficiencies or drawbacks, notably the inherent difficulty of handling discrete variables, the requirement for the (nonlinear) objective and constraint functions to be smooth (i.e. for the gradient-based methods), and the difficulty of handling nonconvexity in nonlinear problems (meaning they can only find locally optimal solutions) (Frank et al., 2012a).

Heuristic optimization techniques employ a variety of optimum-seeking strategies that differ conceptually from those employed in conventional optimization methods. By and large, these techniques make use of a population of candidate search points, which, coupled with their stochastic nature, generally gives them global search characteristics (that is, the ability to globally converge to a solution where one exists,

independently of the initial point, and to find the globally optimal solution, despite nonconvexity of the objective function and the feasible region). They do suffer some drawbacks, however, when compared with the conventional methods, such as lacking mathematical rigor (by virtue of their heuristic nature), being relatively computationally more expensive, and their effectiveness being very dependent on judicious choice of the algorithm parameters.

Table (2.5) presents a high-level comparative analysis of the conventional and heuristic techniques. The comparison is made on the basis of some of the key performance characteristics for an optimization technique that have been outlined in section (2.4), such as computational speed, reliability, robustness, convergence and global optimality properties, among others. The table (along with Tables 2.3 and 2.4) provides a general overview of the relative strengths and shortcomings of the two classes of methods, which should prove to be informative to researchers and other practitioners in the field of engineering optimization.

Table 2.5: Comparison of conventional with nonconventional/heuristic optimization techniques

Characteristic	Classical/conventional optimization techniques	Nonconventional/heuristic optimization techniques
Computational speed	Varies among the various techniques, but *generally faster than heuristic techniques*	*Generally slower than conventional techniques*, largely due to dependence on heuristic search, thus a function of parameter selection
Reliability/quality of solution	A function of problem formulation; e.g. LP is generally reliable (as regards convergence), while Newton's algorithm requires sufficient accuracy of quadratic approximate model relative to original model at each iterate	Mainly a function of parameter selection; no theoretical guarantees can be made generally
Accuracy of model/solution	Somehow a trade-off between accuracy and complexity; e.g. LP formulation (of VVO problem) is fast, but only approximate; MINLP is accurate but also computationally expensive	Generally more versatile, able to handle problems of varying detail and accuracy, often not requiring a precise mathematical model; not dependent on such properties as smoothness of functions, etc.
Robustness	Exhibit sensitivity to initial starting point (e.g. Newton's algorithm), degree of nonlinearity, ill-conditioned nature of problem, and other problem parameters	Many techniques employ heuristics that make them robust, i.e. able to handle problems of diverse characteristics and parameters
Complexity of solution technique	Well-grounded theoretically, well-understood and quite straightforward to implement algorithmically	Heuristic nature of the methods generally requires domain expertise; parameter tuning often requires experience and good understanding of the problem
Handling of discrete variables	Not well-suited to handle discrete variables	Many have ability to handle discrete, even mixed-integer problems quite naturally
Convergence properties	Generally a function of problem definition and such factors as initial starting point and system parameters (e.g. nature of Hessian matrix in case of 2nd-order methods)	Can generally achieve global convergence, independent of problem formulation, although computational expense may be a limiting factor
Global optimality properties	All local solvers, global optimality only achievable in the case of convex problems (which VVO problem is not)	Many can achieve global optimality, although computational expense may be high, if not prohibitive

2.6 Conclusion

Volt/VAR optimization is one of the key operational tools needed by electric power system operators, and has a significant impact on the security, economy, technical viability and efficiency of system operation. It is also one of the most complex optimization problems to solve, being nonlinear, nonconvex and involving both

continuous and discrete variables. The challenge of efficiently and effectively solving the VVO problem is reflected in the diversity of the solution techniques that have been applied to the problem, which exhibit varying characteristics, both in operating principle and how effectively they address the key performance characteristics of the optimization problem. Conventional optimization methods have proven to be efficient, reliable, fast and quite straightforward to algorithmically implement, but suffer from significant drawbacks when applied to the VVO problem, as discussed in section 2.4.1. Particularly, shortcomings exist in their convergence and global optimality properties, and the difficulty in the handling of inequality constraints and discrete variables. The nonconventional/heuristic optimization techniques present some advantages exactly where the conventional techniques fall short, such as superior global search characteristics, thus having the ability to achieve global convergence and global optimality independently of the problem formulation, and the natural ability to handle discrete variables. Their main drawbacks are that their heuristic nature implies that parameter selection weighs heavily on their efficiency and effectiveness, and they incur relatively greater computational expense. The main results of this chapter have been to present a comprehensive and critical literature review that contextualizes the research conducted and presented in this thesis. The results will be applied throughout the remainder of the thesis in model development, and algorithm design and implementation.

The main objective of the research presented in this thesis is to design and implement Volt/VAR optimization strategies that exhibit the desirable characteristics of superior solution quality, high computational efficiency, and scalability (among others), characteristics which are vital for the efficient and optimal operation of modern complex power systems. The following chapter presents two formulations of the Volt/VAR optimization problem, and briefly compares the two models in terms of their computational characteristics. One of the models is then used in subsequent chapters as the basis for the implementation of the Volt/VAR optimization algorithms developed in this research.

CHAPTER THREE

FORMULATION OF THE VOLT/VAR OPTIMIZATION PROBLEM

3.1 Introduction

The electrical power system is arguably one of the most complex engineering systems in existence. For the majority of the world population, reliable electric power supply has become an indispensable daily commodity, with prolonged unavailability thereof causing enough disruption to essential public (and private) services and normal daily activities to be considered practically intolerable. To be able to deliver electric power with the required high reliability and security, while being economical, planning and operational strategies have been developed over the decades by means of which the system can be operated optimally as far as practicable. These strategies are collectively referred to as Optimal Power Flow (**OPF**). In the course of the operation of the power system, changes in load demand and network configurations may cause the system to operate outside of the nominal range, which could threaten the quality and even security or reliability of supply. To prevent or correct anomalous operating conditions, the system operator continually implements controls to regulate the production, absorption and flow of power at all levels in the system. Some of the controlled variables include generator active and reactive power outputs, reactive power (VAR) compensation device outputs, transformer tap settings, phase shifter settings, etc. **OPF** is a sophisticated computational tool used in determining the optimal dispatch of all the system control variables so as to ensure the economical and secure operation of the system, while respecting many functional and operational constraints of the system.

Mathematically, **OPF** has the characteristic of being a very large-scale nonconvex, nonlinear programming problem, with a large number of constraints and a mixture of continuous and discrete variables. Given its great importance to the efficient operation of the power system, the **OPF** problem has attracted extensive research interest over the decades, with particular attention being paid to improvement in problem formulation as well as solution techniques (Momoh, 2001). Optimal reactive power dispatch (**ORPD**) or Volt/VAR optimization (**VVO**) is the variant of **OPF** concerned with the maintenance of system voltage quality to increase system security, at the same time leading to improved system economy (Chebbo, 1990). This chapter is mainly concerned with the presentation of the mathematical formulation of the **VVO** problem, which will then be solved in chapter 5 by the *Interior Point Method* (**IPM**), one of the most efficient classical methods for large-scale nonlinear optimization presently available.

The chapter is organized as follows. After briefly presenting some general definitions and discussing the elements of the problem formulation, two variants of the mathematical formulation of the Volt/VAR optimization problem are developed in section 3.2, one based on the rectangular representation of the system bus voltages, the other based on the polar representation of the voltages. A discussion is then presented in section 3.3 particularly highlighting the distinct characteristics of the two variants of the problem formulation. Concluding remarks for the chapter are then presented in section 3.4. A pictorial representation of the content of this chapter is depicted in Figure (3.1).

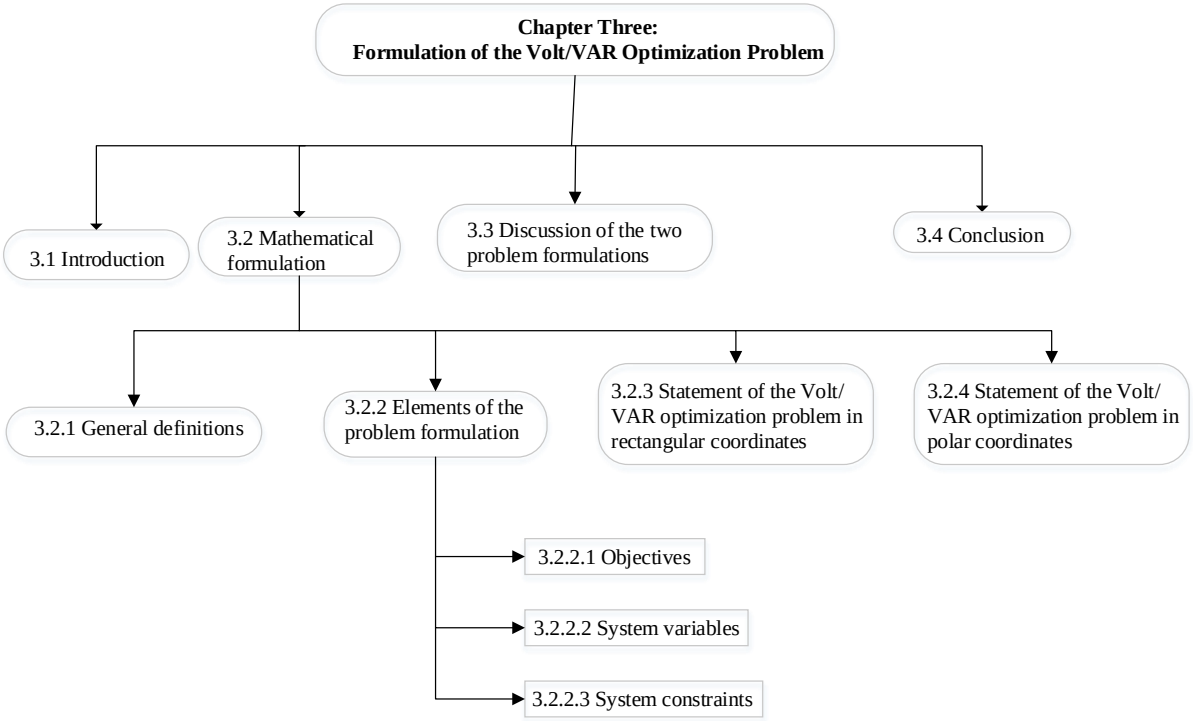

Fig. 3.1: Summary of the content covered in this chapter

3.2 Mathematical formulation

Volt/VAR optimization (**VVO**) is mainly concerned with the determination of the optimal coordinated dispatch of voltage-regulating devices and reactive power sources so as to maintain a secure voltage profile, subject to system functional and operational constraints, while optimizing some aspect of system operation, such as minimization of active power transmission losses (Mataifa *et al.*, 2022). Mathematically, it is formulated as a constrained nonlinear optimization problem, intended to minimize a scalar objective function subject to equality and inequality constraints. This can be stated as:

$$\min \quad f(x,u) \qquad (3.1a)$$

$$subject\ to: \quad g(x,u)=0 \qquad (3.1b)$$

$$u^{\min} \le u \le u^{\max} \qquad (3.1c)$$

$$x^{\min} \le x \le x^{\max} \qquad (3.1d)$$

$$h^{\min} \le h(x,u) \le h^{\max} \qquad (3.1e)$$

Where $f(x,u)$ represents the objective function to be minimized, as a scalar function of the state variable vector, x, and the control or decision variable vector, u, subject to the physical and operational constraints according to (3.1b) – (3.1e). $g(x,u)$ represents functional constraints that need to be satisfied at every operating point (power flow equations in the case of the **VVO** problem), (3.1c) and (3.1d) represent bounds on the values of the control and state variables respectively, and $h(x,u)$ is a generic function of the state and control variables representing other system operational parameters that may need to be constrained within specified limits (e.g. generator reactive power output limits, and branch flow limits, among others).

In the following sub-sections, further details are given pertaining to the problem formulation for the **VVO** problem, based on the generic model (3.1).

3.2.1. General definitions

Since Volt/VAR optimization is essentially an optimal power flow problem, it effectively consists in finding the solution to the power flow problem while minimizing a specified objective function subject to constraints. Therefore, the power flow balance equations play a central role in the problem definition.

For a given network, we can make the following definitions:

- $\mathcal{N}$: *set of all buses in the network*
- $\mathcal{G}$: *set of generators*
- $\mathcal{D}$: *set of consumers (loads)*
- $\mathcal{B}$: *set of branches in the network*
- $\mathcal{L}$: *set of transmission lines*
- $\mathcal{T}$: *set of transformers (with OLTC)*
- $\mathcal{C}$: *set of shunt compensation elements*

Voltage at bus *i* is a complex quantity, and can be defined either in rectangular or polar form as:

$$V_i = e_i + jf_i, \quad \forall i \in \mathcal{N} \qquad (3.2)$$

$$V_i = V_i e^{j\theta_i}, \quad \forall i \in \mathcal{N} \qquad (3.3)$$

Where e_i and f_i are the real and imaginary components of the complex voltage respectively, $V_i = \sqrt{e_i^2 + f_i^2}$, and $\theta_i = \arctan\left(f_i / e_i\right)$ are the magnitude and phase angle of the voltage at bus *i* respectively. The two definitions of the bus voltage lead to two variants of the power flow equations. The rectangular form of the of the active and reactive power injections at bus *i* can be expressed as (Torres, 1998):

$$P_i(e,f,t,q_C) = G_{ii}\left(e_i^2 + f_i^2\right) + e_i \sum_{j\in\mathcal{N}_i}(G_{ij}e_j - B_{ij}f_j) + f_i \sum_{j\in\mathcal{N}_i}(G_{ij}f_j + B_{ij}e_j) \qquad (3.4)$$

$$Q_i(e,f,t,q_C) = -B_{ii}\left(e_i^2 + f_i^2\right) + f_i \sum_{j\in\mathcal{N}_i}(G_{ij}e_j - B_{ij}f_j) - e_i \sum_{j\in\mathcal{N}_i}(G_{ij}f_j + B_{ij}e_j) \qquad (3.5)$$

Where G_{ij} and B_{ij} are the real and imaginary components of the ij[th] element of the bus admittance matrix, Y_{ij} , and t is the vector of (on-load) transformer tap settings, which are implicit in some of the elements of Y .

The polar form of the complex bus voltage representation is commonly used in power system studies, and leads to the following expressions for the power flow equations:

$$P_i(v,\theta,t,q_C) = G_{ii}V_i^2 + V_i \sum_{j\in\mathcal{N}_i} V_j\left[G_{ij}cos\theta_{ij} + B_{ij}sin\theta_{ij}\right] \qquad (3.6)$$

$$Q_i(v,\theta,t,q_C) = -B_{ii}V_i^2 + V_i \sum_{j\in\mathcal{N}_i} V_j\left[G_{ij}sin\theta_{ij} - B_{ij}cos\theta_{ij}\right] \qquad (3.7)$$

Where v and θ are the vectors of the bus voltage magnitude and phase angle respectively, and θ_{ij} is the phase angle difference between buses i and j . The active power losses in the transmission system can be expressed either in rectangular or polar form according to Equations (3.8) and (3.9) respectively (Capitanescu *et al*., 2005, Torres, 1998).

$$P_{Loss}(e,f,t) = \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} G_{ij}\left[(e_i - e_j)^2 + (f_i - f_j)^2\right] \qquad (3.8)$$

$$P_{Loss}(v,\theta,t) = \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} G_{ij}\left[V_i^2 + V_j^2 - 2V_iV_jcos\theta_{ij}\right] \qquad (3.9)$$

3.2.2. Elements of the problem formulation

Formulation of the **VVO** problem consists in defining the system (state and control) variables, the objective(s) of optimization, and the constraints, as reflected in the

generic problem formulation (3.1). Each of these components is further elaborated on in the following sub-sections.

3.2.2.1 Objectives

The main objective of Volt/VAR optimization is to ensure a network voltage profile that meets nominal requirements for bus voltage magnitudes, at the same time optimizing network reactive power dispatch which has a significant impact on the economical operation of the power system (Martinez Ramos *et al.*, 2005). Key objectives of Volt/VAR optimization can thus be stated as:

- Active power loss minimization (Equation (3.8) or (3.9)).
- Reactive power loss minimization (see, for example, Torres, 1998).
- Voltage profile improvement (e.g. minimization of bus voltage deviation from nominal values; see, for example, Momoh, 2001).
- Voltage stability maximization (see, for example, Cai *et al.*, 2007).
- Minimization of control effort to achieve a desired system operating state (see, for example, Cai *et al.*, 2007).

The problem may be formulated to have a single objective or multiple objectives. Traditionally, the main objective considered is active power loss minimization, and has been adopted in this study, in the implementation of the Volt/VAR optimization discussed in chapters 5 and 6.

3.2.2.2 System variables

System variables can be classified into two types: *state* (dependent) variables and *control* (independent) variables. State variables include:

- Load bus voltage magnitudes.
- Load and generator bus phase angles.
- Slack bus real power output.
- Generator reactive power outputs.
- Line flows.

Load and generator bus voltage magnitudes and phase angles are represented in Equation (3.10e), for example, which expresses the system voltages in rectangular coordinates. The slack-bus real power is represented in Equation (3.10b), generator reactive power outputs are represented in Equation (3.10c), and line flows are represented in Equation (3.10d).

Control variables can in turn be classified into those derived from voltage-regulating devices, and those derived from reactive power sources, and include:

- Generator terminal voltage magnitudes.
- Under-Load Tap-Changing (**ULTC**) transformer settings.
- Shunt capacitors and reactors.
- Flexible AC Transmission System (**FACTS**) devices.
- Distributed Generation (**DG**).

Some of these variables are continuous, others are discrete. A complete and most accurate formulation of the **VVO** problem would thus be a Mixed Integer Nonlinear Programming (**MINLP**) problem formulation (Rabie & Parniani, 2013). Although having the desirable characteristic of being accurate, it is also computationally intensive, an important consideration for practical applications, and as far as choice of solution method is concerned.

Generator terminal voltage magnitudes are considered as the control variables in this study in the implementation of the Volt/VAR optimization algorithms in chapters 5 and 6. Control of generator voltage magnitudes implicitly translates into the control of generator reactive power outputs. This constitutes a very efficient way of simultaneously achieving both system voltage regulation and reactive power optimization to minimize system losses (Martinez Ramos et al., 2005).

3.2.2.3 System constraints

The Volt/VAR optimization problem is solved subject to both equality and inequality constraints, which are all generally nonlinear, and encompass operational and functional-type constraints. The main equality constraints are the bus active and reactive power balance equations, but may also include such constraints as voltage magnitude and/or phase angle imposed or required to be of a specified value at a given bus.

Inequality constraints are of two types: operational constraints that apply to the power system state variables, needed to ensure the secure operation of the system, and physical limits on the operating range of values for the control variables. Limits in the form of inequality constraints are typically imposed on each of the following:

- Generator reactive power outputs (Equation 3.10f).
- Bus voltage magnitudes (Equation 3.10e).
- Shunt reactive power compensation device outputs (Equation 3.10g).
- Load tap changing transformer tap settings (Equation 3.10h).
- Line flows (in terms of either active/reactive power or current) (Equation 3.10d).

All these constraints are considered in this study, other than the transformer tap positions (Equation 3.10h) and shunt reactive compensation (Equation 3.10g).

3.2.3. Statement of the Volt/VAR optimization problem in rectangular coordinates

Combining the general definitions and the elements of the problem formulation presented in sections 3.2.1 and 3.2.2 respectively, the Volt/VAR optimization problem formulation (**P_1**) in the rectangular form of the nodal voltages can be stated as (Capitanescu *et al.*, 2005):

$$\min P_{Loss}(e,f,t) \qquad (3.10a)$$

s.t.

$$P_i(e,f,t) + P_{di} - P_{gi} = 0 \qquad i \in \mathcal{N} \qquad (3.10b)$$

$$Q_i(e,f,t) + Q_{di} - q_{ci} - Q_{gi} = 0 \qquad i \in \mathcal{N} \qquad (3.10c)$$

$$\left(G_{ij}^2 + B_{ij}^2\right)\left[\left(e_i - e_j\right)^2 + \left(f_i - f_j\right)^2\right] \le \left(I_{ij}^{max}\right)^2 \qquad i,j \in \mathcal{N} \qquad (3.10d)$$

$$\left(V_i^{min}\right)^2 \le e_i^2 + f_i^2 \le (V_i^{max})^2 \qquad i \in \mathcal{N} \qquad (3.10e)$$

$$Q_{gi}^{min} \le Q_{gi} \le Q_{gi}^{max} \qquad i \in \mathcal{G} \qquad (3.10f)$$

$$q_{ci}^{min} \le q_{ci} \le q_{ci}^{max} \qquad i \in \mathcal{C} \qquad (3.10g)$$

$$\tau_i^{min} \le \tau_i \le \tau_i^{max} \qquad i \in \mathcal{T} \qquad (3.10h)$$

The variables in model **P_1** can be defined as:

$P_{loss}(e,f,t)$: *system active power losses (given by equation* 3.8)

$P_i(e,f,t,q_C)$: *active power injection at bus i (given by equation* 3.4)

$Q_i(e,f,t,q_C)$: *reactive power injection at bus i (given by equation* 3.5)

P_{di} : *active power consumption of load i*

Q_{gi} : *reactive power output of generator i*

q_{ci} : *reactive power output of shunt compensation device i*

Q_{di} : *reactive power consumption of load i*

τ_i : *tap setting of load tap changing (LTC) transformer i*

$Q_{gi}^{min}, Q_{gi}^{max}$: *lower and upper bounds on reactive power output of generator i*

$q_{ci}^{min}, q_{ci}^{max}$: *lower and upper bounds on shunt compensation device i's output*

V_i^{min}, V_i^{max} : *lower and upper bounds on bus i's voltage magnitude*

$\tau_i^{min}, \tau_i^{max}$: *lower and upper bounds on LTC transformer i's tap setting*

I_{ij}^{max} : *maximum permissible current flow in line ij*

3.2.4. Statement of the Volt/VAR optimization problem in polar coordinates

Defining the Volt/VAR optimization problem in the polar form of the nodal voltages (**P₂**) follows the same procedure as that outlined in section 3.2.3 for the rectangular formulation, with the main differences being in the form taken by the objective function, the power flow balance equations, and the bus voltage bound constraints, and can be stated as (Torres, 1998):

$$\min P_{Loss}(V,\theta,t) \qquad (3.11a)$$

s.t.

$$P_i(V,\theta,t,q_C) + P_{di} - P_{gi} = 0 \qquad i \in \mathcal{N} \qquad (3.11b)$$

$$Q_i(V,\theta,t,q_C) + Q_{di} - q_{ci} - Q_{gi} = 0 \qquad i \in \mathcal{N} \qquad (3.11c)$$

$$\left(G_{ij}^2 + B_{ij}^2\right)\left[V_i^2 + V_j^2 - 2V_iV_j\cos\theta_{ij}\right] \le \left(I_{ij}^{max}\right)^2 \qquad i,j \in \mathcal{N} \qquad (3.11d)$$

$$V_i^{min} \le V_i \le V_i^{max} \qquad i \in \mathcal{N} \qquad (3.11e)$$

$$Q_{gi}^{min} \le Q_{gi} \le Q_{gi}^{max} \qquad i \in \mathcal{G} \qquad (3.11f)$$

$$q_{ci}^{min} \le q_{ci} \le q_{ci}^{max} \qquad i \in \mathcal{C} \qquad (3.11g)$$

$$\tau_i^{min} \le \tau_i \le \tau_i^{max} \qquad i \in \mathcal{T} \qquad (3.11h)$$

The variables in model **P₂** that are different from those in model **P₁** are defined as follows:

$P_{loss}(V,\theta,t)$: *system active power losses (given by equation* 3.9)

$P_i\left(V,\theta,t,q_C\right)$: *active power injection at bus i (given by equation* 3.6)

$Q_i\left(V,\theta,t,q_C\right)$: *reactive power injection at bus i (given by equation* 3.7)

3.3 Discussion of the two problem formulations

Both the objective function and the main constraints (the power balance equations) of the Volt/VAR optimization problem are nonlinear and nonconvex. Computational effort to evaluate these functions and their derivatives (in the case of gradient-based solution techniques) is an important consideration, and in this respect, a comparison can be made between the rectangular and polar formulations as presented in sections 3.2.3 and 3.2.4 respectively. The main distinction to be observed is that in the case of the rectangular formulation (model **P₁**), both the objective function (equation 3.8) and power balance equality constraints (equations 3.4, 3.5, 3.10b and 3.10c) are quadratic functions of the complex voltage components (e_i, f_i), whereas this is not the case for the polar formulation due to the presence of trigonometric terms in the corresponding expressions (Equations 3.6, 3.7, 3.9, 3.11b and 3.11c). This is an important distinction which has significant implications for the relative computational efficiency of the two formulations. Particularly, the advantages of the

rectangular formulation resulting in quadratic objective and constraint functions consist in the fact that (Capitanescu *et al.*, 2005):

- The second-order derivatives (Hessian matrices) of the objective and constraint functions (including the power balance equations, branch flow limits and bus voltage bound constraints) are constant
- The Taylor-series expansion of a quadratic function terminates at the second-order term without any truncation error, and is thus relatively more accurate

The properties stated above make the rectangular formulation computationally efficient in an optimization technique that requires availability of second-order derivative information of the objective and constraint functions, such as the Newton method employed in interior-point methods (as outlined in the next chapter). The (slight) disadvantage of the rectangular formulation lies in the fact that bus voltage bound constraints (Equation 3.10e) need to be treated as functional constraints, whereas in the polar formulation, they are handled as simple bound constraints (Equation 3.11e).

For the reasons discussed above, the rectangular formulation (model $\mathbf{P_1}$) has been adopted in this study, and is used in the implementation of the Volt/VAR optimization in chapters 5 and 6.

3.4 Conclusion

Volt/VAR optimization (**VVO**) is the variant of the optimal power flow (**OPF**) problem concerned with the optimal dispatch of voltage-regulating and reactive power control devices so as to minimize system losses and improve the network voltage profile, and assumes that an economic dispatch that minimizes the fuel cost of meeting a given demand has already been computed. This chapter has presented the problem formulation for the **VVO** problem, thoroughly discussing the main elements of the problem formulation (i.e. the system variables comprising the state and control variables, the objective functions, and the constraints). The problem formulation has been presented in both the rectangular and polar forms of the complex bus voltage, and relative merits and demerits of the two formulations have been briefly discussed.

The main result of this chapter, which is the development of the model used in the design of the algorithm for the VVO problem, will be used in chapters 5 and 6, and contributes to one of the main deliverables of this thesis (*i.e.*, problem formulation). In the next chapter, the solution of the **VVO** problem based on the interior-point method, one of the most efficient classical optimization techniques for large-scale nonlinear optimization, will be presented.

CHAPTER FOUR

DESIGN AND IMPLEMENTATION OF THE PRIMAL-DUAL INTERIOR POINT METHOD AND APPLICATION TO A NONLINEAR PROBLEM

4.1 Introduction

The general optimal power flow (**OPF**) problem was first formulated in the early 1960s (Carpentier, 1962), and has since then attracted a lot of research interest, with the primary focus being on the development of efficient problem formulations, as well as effective solution methods. With the characteristic of being a very large-scale nonlinear, nonconvex optimization problem with typically thousands of variables and constraints, as well as a mix of continuous and discrete variables, solving the **OPF** problem efficiently poses special challenges, and much effort has been dedicated to the on-going search for efficient solution techniques over the decades. Gradient-based techniques constituted the first approaches applied to the algorithmic solution of the **OPF** problem (Dommel & Tinney, 1968; Peschon *et al.*, 1968; Hano *et al.*, 1969). Over the years, a variety of solution methods have been investigated and developed, broadly classified as conventional/classical or deterministic methods, and non-conventional/heuristic or non-deterministic methods. The principle classical optimization methods include a variety of gradient-based methods (e.g. reduced-gradient, generalized reduced-gradient, conjugate-gradient, Newton and quasi-Newton methods), (successive) linear programming, (successive) quadratic programming, and interior-point methods. Heuristic optimization methods encompass genetic algorithms, evolutionary programming, particle swarm optimization, fuzzy set theory, and expert systems, among others (Mataifa et al., 2022).

Interior-point methods (IPM) stand out among traditional optimization methods due to a number of characteristics that make them very suitable for application to large-scale nonlinear programming problems. Having been initially developed as an alternative to the Simplex method for solving linear programs (LP), their success in this class of optimization problems elicited the attention of the operations research community, and substantial efforts were made quite early in their development to extend their application to nonlinear programming problems (Capitanescu *et al*., 2007). With these initial efforts bearing fruitful results, their popularity continued to grow, and until today they constitute one of the most widely researched and most commonly applied classical optimization methods to the OPF problem (Frank & Rebennack, 2012).

This chapter presents the design and implementation of the primal-dual interior-point method (PDIPM), a principal variant of interior-point methods, which then forms the basis for the development of an efficient Volt/VAR optimization (VVO) algorithm in the following chapter. Following this introductory section, the details of the development

of the algorithm are presented in section 4.2. This is followed by an illustrative example in section 4.3 that demonstrates the implementation of the developed algorithm, thoroughly covering all the aspects of the algorithm discussed in section 4.2. Section 4.4 concludes the chapter with a brief summary of the key results from this chapter. Figure (4.1) depicts a pictorial representation of the content of this chapter.

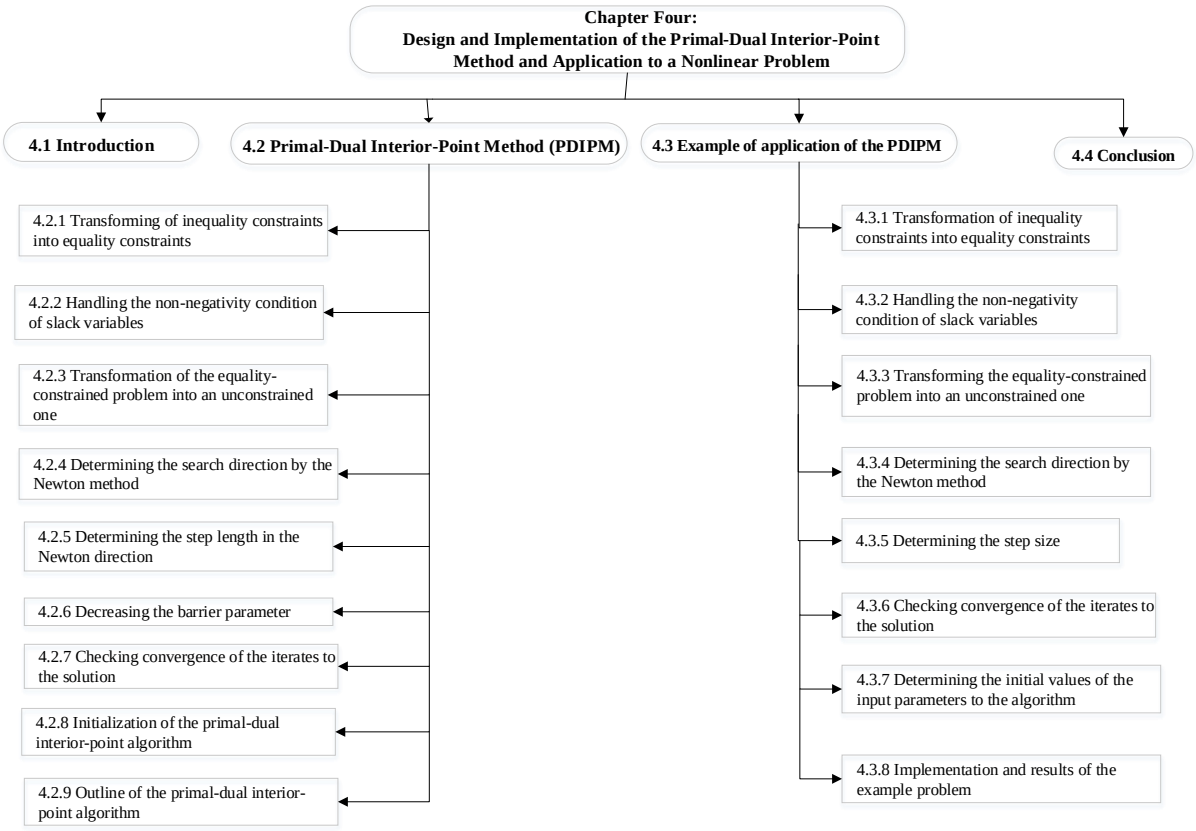

Fig. 4.1: Summary of the content covered in this chapter

4.2 Primal-Dual Interior Point-Method (PDIPM)

The primal-dual interior-point method (PDIPM) effectively combines three concepts to provide an approach for solving constrained nonlinear optimization problems: (i) handling of inequality constraints by means of a logarithmic barrier function, (ii) application of Lagrangian theory of optimization to the solution of an equality-constrained optimization problem, and (iii) application of the Newton method to solve the resulting system. The main steps of the technique can be outlined as:

1. Transforming all inequality constraints into equality constraints by adding a nonnegative slack variable to each inequality constraint.
2. Implicit handling of the non-negativity condition of slack variables by appending each of them to the objective function using a logarithmic barrier function.

3. Transforming the resulting equality-constrained optimization problem into an unconstrained one using the Lagrangian approach.
4. Solving the resulting unconstrained optimization problem using the Newton method.

To form the basis for the development of the primal-dual interior-point algorithm (**PDIPA**) to be applied to the solution of the **VVO** problem, the general nonlinear programming problem (initially stated in equations (3.1) – (3.5)) can be restated here as:

$$\min f(x) \qquad (4.1a)$$

$$s.t. \quad g(x) = 0 \qquad (4.1b)$$

$$h(x) \leq 0 \qquad (4.1c)$$

where $f(x)$ represents the objective function as a scalar-valued function of the combined state and control variable vector $x \in R^n$, $g(x) \in R^m$ represents the active and reactive power balance equations, and $h(x) \in R^p$ combines the operational and functional inequality constraints of the **VVO** problem (as stated in problem $\mathbf{P_1}$ or $\mathbf{P_2}$ in chapter 3). In the following subsections, the steps of the **PDIPM** outlined earlier are applied to the problem (4.1) in order to develop the **PDIPA**.

4.2.1 Transformation of inequality constraints into equality constraints

By adding slack variables to the inequality constraints given by equation (4.1c), the general nonlinear program (4.1) is transformed into:

$$\min f(x) \qquad (4.2a)$$

$$s.t. \quad g(x) = 0 \qquad (4.2b)$$

$$h(x) + s = 0 \qquad (4.2c)$$

$$s \geq 0 \qquad (4.2d)$$

4.2.2 Handling the non-negativity condition of slack variables

The non-negativity condition of the slack variables (equation 4.2d) is handled by appending them to the objective function by means of the logarithmic barrier function, which has the singularity property at the origin. With this modification, the problem (4.2) becomes:

$$\min f(x) - \mu \sum_{i=1}^{p} \ln(s_i) \qquad (4.3a)$$

$$s.t. \quad g(x) = 0 \qquad (4.3b)$$

$$h(x) + s = 0 \qquad (4.3c)$$

where μ is a positive scalar, referred to as the barrier parameter, which is progressively decreased to zero as the iteration progresses. It has been shown by Fiacco and McCormick (1968) that as μ tends to zero, the solution of the problem (4.3), $x(\mu)$, approaches the optimizer of problem (4.1), x^*. In problem (4.3), the non-negativity condition on the slack variables is handled implicitly through the logarithmic barrier functions appended to the objective function, so that equation (4.2d) is no longer required, and so does not appear in (4.3).

4.2.3 Transformation of the equality-constrained problem into an unconstrained one

The next step in the development of the algorithm is to formulate the Lagrangian function of the problem by forming a linear combination of the objective function (equation (4.3a)) and the equality constraints (equations (4.3b), (4.3c)) with the help of Lagrangian multipliers, thus transforming the equality-constrained problem into an unconstrained problem (Capitanescu *et al.*, 2005):

$$\mathcal{L}_\mu = f(x) - \mu \sum_{i=1}^{p} \ln(s_i) + \lambda_E^T g(x) + \lambda_I^T (h(x) + s) \qquad (4.4)$$

where λ_E and λ_I are the Lagrange multipliers for the equality and inequality constraints respectively. The first-order optimality conditions of the problem (known as the *Karush-Kuhn-Tucker* or **KKT** conditions) are derived by taking the first partial derivatives of the Lagrangian function (equation (4.4)) with respect to both the primal and dual variables (i.e. the vector $X = [x, s, \lambda_E, \lambda_I]^T$), and equating each of them to zero, which yields:

$$\begin{aligned}
\nabla_x \mathcal{L}_\mu &= \nabla f(x) + \nabla g^T(x)\lambda_E + \nabla h^T(x)\lambda_I = 0 && (4.5a)\\
\nabla_s \mathcal{L}_\mu &= -\mu S^{-1}\boldsymbol{e} + \lambda_I = 0 && (4.5b)\\
\nabla_{\lambda_E} \mathcal{L}_\mu &= g(x) = 0 && (4.5c)\\
\nabla_{\lambda_I} \mathcal{L}_\mu &= h(x) + s = 0 && (4.5d)
\end{aligned}$$

where e is a vector of ones of appropriate length (i.e. $e = [1, 1, ..., 1]^T$), S is a diagonal matrix with the slack variables on the diagonal (i.e. $S = diag(s_1, s_2, ..., s_p)$).

The **KKT** conditions (4.5) can be written in a compact form as:

$$F(X) = \begin{bmatrix} \nabla_x \mathcal{L}_\mu \\ \nabla_s \mathcal{L}_\mu \\ \nabla_{\lambda_E} \mathcal{L}_\mu \\ \nabla_{\lambda_I} \mathcal{L}_\mu \end{bmatrix} = \begin{bmatrix} \nabla f(x) + \nabla g^T(x)\lambda_E + \nabla h^T(x)\lambda_I \\ S\lambda_I - \mu \boldsymbol{e} \\ g(x) \\ h(x) + s \end{bmatrix} \qquad (4.6)$$

Equation (4.6) has been derived from (4.5) by multiplying equations (4.5b) by S, which has the advantage (particularly for the Newton method) of decreasing the relative nonlinearity of the system (4.5) near the solution as $s \to 0$ (Nocedal & Wright, 2006).

Determining the solution of the general nonlinear programming problem (4.1) typically takes the form of an iterative process that seeks the point $X^* = [x^*, s^*, \lambda_E^*, \lambda_I^*]$ which satisfies the **KKT** conditions (4.6), as well as the non-negativity conditions on the slack and dual variables. For the primal-dual interior-point method, the main components of this iterative process include:

- Determining the search direction (by the Newton method).
- Determining the step size to be taken in the already computed search direction, then updating the primal and dual variables.
- Updating the barrier parameter, which should monotonically be decreased to zero as the iterative process proceeds.
- Checking the stopping criteria that indicate the algorithm's convergence to the solution of the problem.

These aspects of the algorithm are discussed in detail in the following sub-sections.

4.2.4 Determining the search direction by the Newton method

The Newton method is an iterative procedure for finding the solution to a nonlinear problem of the form (Zhu, 2009):

$$F(X) = 0 \qquad (4.7)$$

which involves generating the Taylor series expansion of $F(X)$ about an initial estimated solution X^0, subjected to a small increment ΔX^0:

$$F(X^0 + \Delta X^0) = F(X^0) + F'(X^0)\Delta X^0 + F''(X^0)\frac{(\Delta X^0)^2}{2!} + F^n(X^0)\frac{(\Delta X^0)^n}{n!} + \cdots = 0 \qquad (4.8)$$

where $F'(X^0), \ldots, F^n(X^0)$ are the derivatives of the function $F(X)$, evaluated at X^0. Assuming that the initial point X^0 is close to the solution for problem (4.7), such that the change ΔX^0 can be considered to be relatively small, a first-order model can be

derived from equation (4.8) by neglecting the higher-order terms, leading to the approximate model:

$$F(X^0 + \Delta X^0) \cong F(X^0) + F'(X^0)\Delta X^0 = 0 \tag{4.9}$$

From equation (4.9), the Newton iteration is derived as:

$$\Delta X^k = -\left(J(X^k)\right)^{-1} F\left(X^k\right) \tag{4.10}$$

where $J(X^k)$ is the matrix of first partial derivatives of $F(X^k)$, $F'(X^k)$ (known as the Jacobian of $F(X^k)$). The increment ΔX^k is successively added to the current solution at each iteration until the approximate solution reaches a sufficient level of accuracy. k denotes the iteration number, and may be omitted in subsequent expressions (for example, in Equation 4.11) for the sake of simplicity.

To determine the Newton-based search direction for the **KKT** system, equation (4.10) is applied to equation (4.6), resulting in the system:

$$J(X)\Delta X = -F(X) \tag{4.11}$$

where:

$$J(X) = \begin{bmatrix} \nabla^2_{xx}\mathcal{L}_\mu & \nabla^2_{xs}\mathcal{L}_\mu & \nabla^2_{x\lambda_E}\mathcal{L}_\mu & \nabla^2_{x\lambda_I}\mathcal{L}_\mu \\ \nabla^2_{sx}\mathcal{L}_\mu & \nabla^2_{ss}\mathcal{L}_\mu & \nabla^2_{s\lambda_E}\mathcal{L}_\mu & \nabla^2_{s\lambda_I}\mathcal{L}_\mu \\ \nabla^2_{\lambda_E x}\mathcal{L}_\mu & \nabla^2_{\lambda_E s}\mathcal{L}_\mu & \nabla^2_{\lambda_E\lambda_E}\mathcal{L}_\mu & \nabla^2_{\lambda_E\lambda_I}\mathcal{L}_\mu \\ \nabla^2_{\lambda_I x}\mathcal{L}_\mu & \nabla^2_{\lambda_I s}\mathcal{L}_\mu & \nabla^2_{\lambda_I\lambda_E}\mathcal{L}_\mu & \nabla^2_{\lambda_I\lambda_I}\mathcal{L}_\mu \end{bmatrix} = \begin{bmatrix} \nabla^2_{xx}\mathcal{L}_\mu & 0 & \nabla g^T(x) & \nabla h^T(x) \\ 0 & \Lambda_I & 0 & S \\ \nabla g(x) & 0 & 0 & 0 \\ \nabla h(x) & I & 0 & 0 \end{bmatrix} \tag{4.12}$$

$$\Delta X = \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta\lambda_E \\ \Delta\lambda_I \end{bmatrix} \tag{4.13}$$

$$\nabla^2_{xx}\mathcal{L}_\mu = \nabla^2_{xx} f(x) + \nabla^2_{xx} g^T(x)\lambda_E + \nabla^2_{xx} h^T(x)\lambda_I \tag{4.14}$$

and $F(X)$ is given by equation (4.6). Combining equations (4.6), (4.12) and (4.13), we obtain the following (primal-dual) system:

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}_\mu & 0 & \nabla g^T(x) & \nabla h^T(x) \\ 0 & \Lambda_I & 0 & S \\ \nabla g(x) & 0 & 0 & 0 \\ \nabla h(x) & I & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta\lambda_E \\ \Delta\lambda_I \end{bmatrix} = - \begin{bmatrix} \nabla f(x) + \nabla g^T(x)\lambda_E + \nabla h^T(x)\lambda_I \\ S\lambda_I - \mu \boldsymbol{e} \\ g(x) \\ h(x) + s \end{bmatrix} \tag{4.15}$$

where Λ_I is a diagonal matrix with the Lagrange multiplier vector for the inequality constraints, λ_I, on the diagonal (i.e. $\Lambda_I = diag(\lambda_I)$). By expressing Δs in terms of Δx and $\Delta\lambda_I$ in terms of Δs, a reduced-order system can be derived from system (4.15), as detailed below. Considering the second and fourth rows of equation (4.15), the following expressions in terms of $\Delta\lambda_I$ and Δs are derived.

$$\nabla h(x)\Delta x + \Delta s = -h(x) - s$$

$$\Longrightarrow \quad \Delta s = -h(x) - s - \nabla h(x)\Delta x \tag{4.16}$$

$$\begin{aligned}\Delta\lambda_I &= S^{-1}(-S\lambda_I + \mu \boldsymbol{e} - \Lambda_I \Delta s)\\ &= S^{-1}(-S\lambda_I + \mu \boldsymbol{e} - \Lambda_I(-h(x) - s - \nabla h(x)\Delta x))\\ &= -\lambda_I + \mu S^{-1}\boldsymbol{e} + S^{-1}\Lambda_I(h(x) + s + \nabla h(x)\Delta x)\end{aligned} \tag{4.17}$$

Then substituting equation (4.17) into the first row of (4.15) leads to:

$$\begin{aligned}&\nabla^2_{xx}\mathcal{L}_\mu \Delta x + \nabla g^T(x)\Delta\lambda_E + \nabla h^T(x)\Delta\lambda_I = -\nabla_x \mathcal{L}_\mu\\ &\nabla^2_{xx}\mathcal{L}_\mu \Delta x + \nabla g^T(x)\Delta\lambda_E + \nabla h^T(x)\left(-\lambda_I + \mu S^{-1}\boldsymbol{e} + S^{-1}\Lambda_I(h(x) + S + \nabla h(x)\Delta x)\right) = -\nabla_x \mathcal{L}_\mu\\ &\left(\nabla^2_{xx}\mathcal{L}_\mu + \nabla h^T(x)S^{-1}\Lambda_I \nabla h(x)\right)\Delta x + \nabla g^T(x)\Delta\lambda_E\\ &\qquad = -\left(\nabla_x \mathcal{L}_\mu + \nabla h^T(x)\left(-\lambda_I + \mu S^{-1}\boldsymbol{e} + S^{-1}\Lambda_I(h(x) + s)\right)\right)\end{aligned} \tag{4.18}$$

Taking equation (4.18) and the third row of equation (4.15), the reduced-order primal-dual system can be written as:

$$\begin{bmatrix} A & \nabla g^T(x) \\ \nabla g(x) & 0 \end{bmatrix}\begin{bmatrix} \Delta x \\ \Delta\lambda_E \end{bmatrix} = -\begin{bmatrix} B \\ g(x) \end{bmatrix} \tag{4.19}$$

where:

$$A = \nabla^2_{xx}\mathcal{L}_\mu + \nabla h^T(x)S^{-1}\Lambda_I \nabla h(x) \tag{4.20}$$

$$B = \nabla_x \mathcal{L}_\mu + \nabla h^T(x)S^{-1}\left(\mu \boldsymbol{e} + \Lambda_I h(x)\right) \tag{4.21}$$

Thus, to determine the Newton direction for the primal-dual system (4.15), we can solve the reduced-order system (4.19) for Δx and $\Delta\lambda_E$, then determine Δs and $\Delta\lambda_I$ using equations (4.16) and (4.17) respectively.

4.2.5 Determining the step length to take in the Newton direction

Once the Newton direction has been computed as in the previous sub-section, the primal and dual variables ($X = [x, s, \lambda_E, \lambda_I]$) are updated according to:

$$x^{k+1} = x^k + \alpha_p^k \Delta x^k \tag{4.22a}$$

$$s^{k+1} = s^k + \alpha_p^k \Delta s^k \tag{4.22b}$$

$$\lambda_E^{k+1} = \lambda_E^k + \alpha_d^k \Delta \lambda_E^k \tag{4.22c}$$

$$\lambda_I^{k+1} = \lambda_I^k + \alpha_d^k \Delta \lambda_I^k \tag{4.22d}$$

$\alpha_p \in (0,1]$ and $\alpha_d \in (0,1]$ are the step lengths taken in the Newton direction for the primal and dual spaces respectively. Incorporating a step length adjustment into the Newton step computation has the dual objective of maintaining the strict positivity condition of the slack variables and their corresponding dual variables, while providing for a sufficient advance towards the (local) minimizer of the problem. The following expressions are commonly used in the computation of the primal and dual step lengths (Capitanescu et al., 2005; Torres, 1998):

$$\alpha_p = \min\left(1,\ \zeta \min_{\Delta s_i < 0}\left(-\frac{s_i}{\Delta s_i}\right)\right) \tag{4.23a}$$

$$\alpha_d = \min\left(1,\ \zeta \min_{\Delta \lambda_{Ii} < 0}\left(-\frac{\lambda_{Ii}}{\Delta \lambda_{Ii}}\right)\right) \tag{4.23b}$$

where $\zeta \in (0,1)$ is a scalar parameter slightly less than one, referred to as the *safety factor*, and is intended to ensure strict positivity of the slack variables and their corresponding dual variables. It is commonly set to be as close to one as possible, to enable as large a step in the Newton direction as possible. A usual value of the parameter is $\zeta = 0.99995$.

Close coupling between the primal and dual variables in the case of interior-point methods for general nonlinear programming (as reflected in the dual feasibility condition in equation (4.5b)), gives rise to the consideration of a common step length adjustment for the primal and dual variables, in which case it can be derived from equation (4.23) as:

$$\alpha_p = \alpha_d = \min(\alpha_p, \alpha_d) \tag{4.24}$$

The exceptional case when such use of a common step length might not be effective is when there is a large difference in magnitude between the primal and dual step lengths, usually an indication of a poorly centred iteration, in which case it may be preferable to use separate step length adjustments (Capitanescu *et al.*, 2005).

4.2.6 Decreasing the barrier parameter

For the solution of problem (4.3) (having the logarithmic barrier function) to coincide with that of the original problem (4.1), a scheme is required as part of the iterative process to monotonically decrease the sequence of barrier parameters $\{\mu^k\}$ until it

converges to zero at the solution. The rate of decrease of the barrier parameter does affect the rate of progress of the iterative process. If decreased too slowly, the number of iterations required for the interior-point algorithm becomes large. If decreased too quickly, some of the slack or dual variables may approach zero prematurely, again slowing down the rate of progress of the iterations (Nocedal & Wright, 2006). Most (modern) implementations of the interior-point method use an adaptive strategy for updating the barrier parameter, varying it at every iteration as a function of the progress of the algorithm, based on the complementarity gap, that is, the residue of the complementarity constraints:

$$\rho = s^T \lambda_I \tag{4.25}$$

The barrier parameter is adjusted proportionately to the complementarity gap (equation (4.25)) according to:

$$\mu^{k+1} = \sigma^k \frac{\rho^k}{2(m+p)} \tag{4.26}$$

where m and p are the numbers of equality and inequality constraints respectively, k is the iteration number, σ^k is referred to as the *centering parameter*, for which there exist several heuristics for setting its value (Nocedal & Wright, 2006). Its value is essentially a compromise between achieving optimality (i.e. making substantial advance in the Newton direction) and feasibility (i.e. improving centrality of the iterate). A value in the range $\sigma \in [0.1, 0.2]$ is often used. Choice of the initial barrier parameter (μ^0) is also an important consideration for the convergence characteristics of the algorithm, and is to some extent problem-dependent. Its choice is governed by the need to avoid the premature forcing of the inequality constraints to become active, which may adversely affect the convergence of the algorithm ((Capitanescu *et al.*, 2005; Torres, 1998).

4.2.7 Checking convergence of the iterates to the solution

The interior-point algorithm is considered to have converged when an approximate local minimum has been obtained, which is signified by the iterate satisfying specified tolerances for the primal feasibility, (scaled) dual feasibility, (scaled) complementarity gap and scaled objective function variation. Additionally, the barrier parameter is often required to decrease to a specified tolerance (as opposed to the requirement of becoming null at the solution). The termination conditions are given in Equation (4.27), and have been adopted in the implementation of the PDIPM algorithm in this study (Capitanescu et al., 2005).

$$Primal\ feasibility = \max(\|g(x)\|_\infty, \max h(x)) \leq \varepsilon_1 \quad (4.27a)$$

$$Gradient\ condition = \frac{\|\nabla f(x) + \nabla g^T(x)\lambda_E + \nabla h^T(x)\lambda_I\|_\infty}{1 + \|x\|_2 + \|\lambda_E\|_2 + \|\lambda_I\|_2} \leq \varepsilon_1 \quad (4.27b)$$

$$Complementarity\ condition = \frac{\rho}{1 + \|x\|_2} \leq \varepsilon_1 \quad (4.27c)$$

$$Objective\ condition = \frac{|f(x^k) - f(x^{k-1})|}{1 + |f(x^k)|} \leq \varepsilon_2 \quad (4.27d)$$

$$Barrier\ parameter\ condition = \mu^k \leq \varepsilon_\mu \quad (4.27e)$$

Typical values for the tolerances are $\varepsilon_1 = 10^{-4}, \varepsilon_2 = 10^{-6}, \varepsilon_\mu = 10^{-6}$ (Martinez Ramos *et al.*, 2005). A point X^k that satisfies the above conditions is said to be a **KKT** point of accuracy ε_1. Besides satisfying optimality conditions according to equation (4.27), the algorithm may also terminate unsuccessfully, either due to numerical infeasibility (e.g. when the primal/dual step lengths become so small that no further progress can be made either towards reaching optimality or decreasing the barrier parameter), or the predetermined maximum number of iterations being reached.

4.2.8 Initialization of the primal-dual interior-point algorithm

The primal-dual interior-point algorithm is referred to as an infeasible interior-point method, in the sense that it need not start from a feasible initial point, the only requirement being the satisfaction of the strict positivity condition on the slack variables and their corresponding dual variables (Torres & Quintana, 1998). In spite of this fact, the initial point may have a significant impact on the convergence characteristics of the algorithm, and thus problem-specific heuristics may be applied to come up with a 'good' initial point. Such a 'good' initial point should ideally be well-centred (such that values of complementarity products $\left(s^0\right)^k \left(\lambda_I^0\right)^k$ are comparable for every iteration index k), and should not be 'too infeasible' (as measured by the complementarity gap).

For the optimal power flow problem, it is recommended to initialize the decision vector x^0 with the solution of a load flow computation, if available. Otherwise, the values of the variables may be set to be in the middle of the range determined by their lower and upper limits. The Lagrange multiplier for the equality constraints (λ_E) may be set to zero, while the slack variable vector and its corresponding dual variable (i.e. the Lagrange multiplier for the inequality constraints, λ_I) may be set as follows (Torres, 1998):

$$s^0 = \min(\max(\gamma(h^M - h^m), h(x^0) - h^m), (1-\gamma)(h^M - h^m)) \quad (4.28a)$$

$$\lambda_I^0 = \mu^0 (S^0)^{-1} \boldsymbol{e} \qquad (4.28b)$$

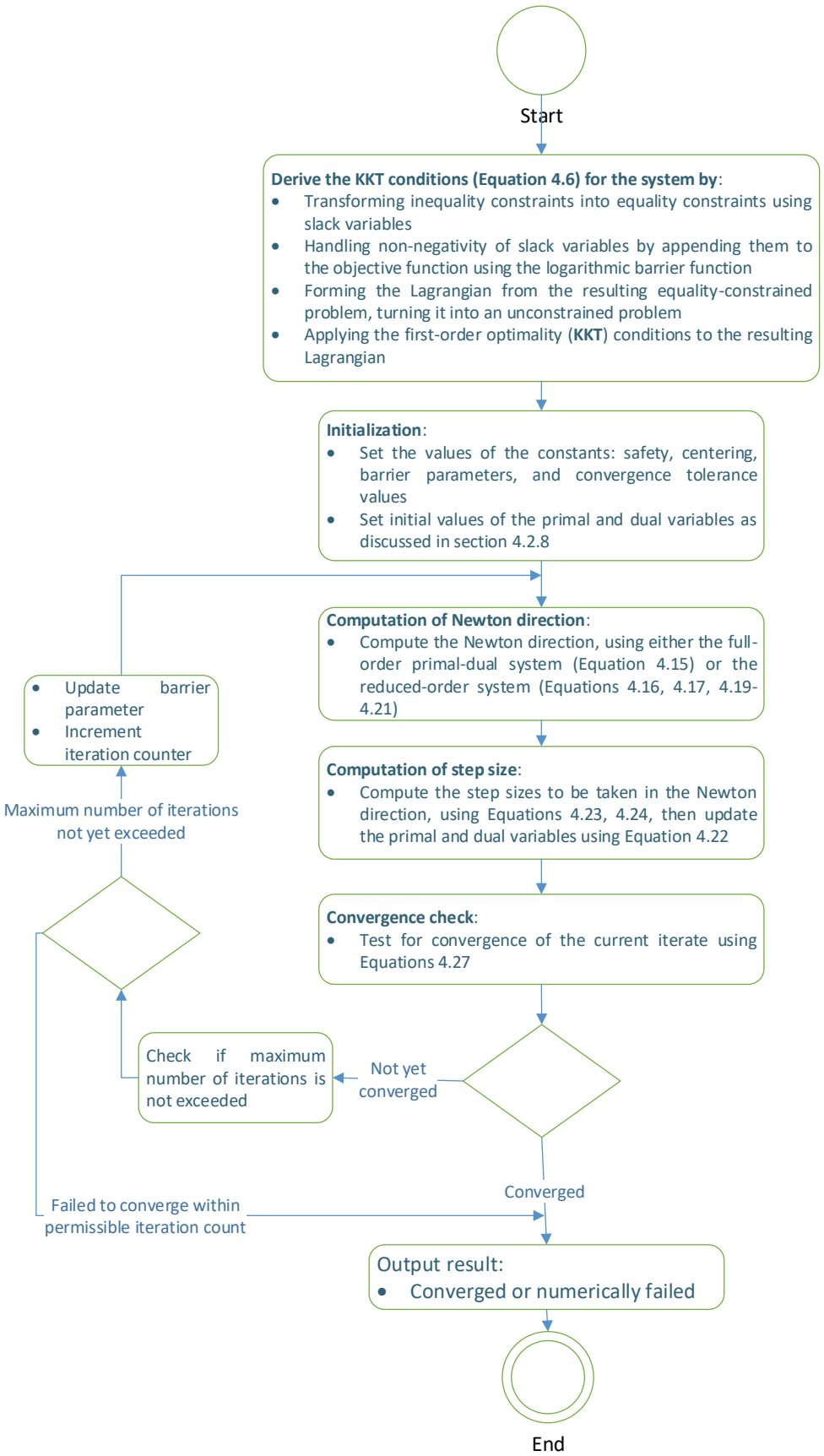

Fig. 4.2: Flowchart of the Primal-Dual Interior-Point Algorithm

where h^m and h^M are the lower and upper limits on the inequality constraints $h(x)$, the scalar parameter γ is chosen to be in the range $\gamma \in [0.1, 0.3]$. Equation (4.28a) ensures that the maximum inequality constraint violation is minimized.

4.2.9 Outline of the primal-dual interior-point algorithm

The flowchart in Figure (4.2) below summarizes the steps of the primal-dual interior-point algorithm discussed in the preceding sub-sections.

4.3 Example of application of the PDIPM

To illustrate the implementation of the primal-dual interior-point algorithm developed in the previous section, it is applied to the following simple problem, taken from Momoh (2001), stated as:

$$\min f(x) = x_1 x_2 \qquad (4.29a)$$

s.t.

$$1 \leq x_1 - x_2 \leq 2 \qquad (4.29b)$$

This is a (fairly simple) inequality-constrained quadratic programming problem (with a quadratic objective function and linear inequality constraints). It can be expressed in standard form as:

$$\min f(x) = x_1 x_2 \qquad (4.30a)$$

s.t.

$$h_1(x) = x_1 - x_2 - 2 \leq 0 \qquad (4.30b)$$

$$h_2(x) = -x_1 + x_2 + 1 \leq 0 \qquad (4.30c)$$

In the following sub-sections, the steps outlined in section (4.2) will be applied to problem (4.30) so as to solve the system using the primal-dual interior-point method.

4.3.1 Transformation of inequality constraints into equality constraints

By adding slack variables to the two inequality constraints, the problem is transformed into an equality-constrained problem as follows:

$$\min f(x) = x_1 x_2 \qquad (4.31a)$$

s.t.

$$x_1 - x_2 - 2 + s_1 = 0 \qquad (4.31b)$$

$$-x_1 + x_2 + 1 + s_2 = 0 \qquad (4.31c)$$

$$s_1, s_2 \geq 0 \qquad (4.31d)$$

4.3.2 Handling the non-negativity condition of slack variables

The next step is to form the logarithmic barrier function to handle the non-negativity condition of the slack variables, resulting in the problem taking the following form:

$$\min f(x) - \mu \sum_{i=1}^{2} \ln(s_i) = x_1 x_2 - \mu(lns_1 + lns_2) \qquad (4.32a)$$

$$s.t. \quad x_1 - x_2 - 2 + s_1 = 0 \qquad (4.32b)$$

$$-x_1 + x_2 + 1 + s_2 = 0 \qquad (4.32c)$$

4.3.3 Transforming the equality-constrained problem into an unconstrained one

The logarithmic barrier function and the equality constraints as presented in problem (4.32) are then combined to form the Lagrangian function of the problem as follows:

$$\mathcal{L}_\mu = f(x) - \mu \sum_{i=1}^{2} \ln(s_i) + \lambda_E^T g(x) + \lambda_I^T (h(x) + s)$$

$$= x_1 x_2 - \mu(lns_1 + lns_2) + \lambda_{I1}(x_1 - x_2 + s_1 - 2) + \lambda_{I2}(-x_1 + x_2 + s_2 + 1) \qquad (4.33)$$

Note that the original problem has no equality constraints, and so the term $\lambda_E^T g(x)$ does not appear in the Lagrangian function (4.33) of the problem. The first-order optimality (i.e. **KKT**) conditions for the problem are derived by taking the partial derivatives of the Lagrangian function with respect to the primal and dual variables as follows:

$$\nabla_x \mathcal{L}_\mu = \begin{bmatrix} \nabla_{x_1} \mathcal{L} \\ \nabla_{x_2} \mathcal{L} \end{bmatrix} = \begin{bmatrix} x_2 + \lambda_{I1} - \lambda_{I2} \\ x_1 - \lambda_{I1} + \lambda_{I2} \end{bmatrix} = 0 \qquad (4.34a)$$

$$\nabla_s \mathcal{L}_\mu = -\mu S^{-1} \boldsymbol{e} + \lambda_I = \begin{bmatrix} -\mu s_1^{-1} + \lambda_{I1} \\ -\mu s_2^{-1} + \lambda_{I2} \end{bmatrix} = 0 \qquad (4.34b)$$

$$\nabla_{\lambda_E} \mathcal{L}_\mu = g(x) = 0 \qquad (4.34c)$$

$$\nabla_{\lambda_I} \mathcal{L}_\mu = h(x) + s = \begin{bmatrix} x_1 - x_2 - 2 \\ -x_1 + x_2 + 1 \end{bmatrix} = 0 \qquad (4.34d)$$

The **KKT** conditions can be written in compact form as:

$$F(x) = \begin{bmatrix} \nabla_x \mathcal{L}_\mu \\ \nabla_s \mathcal{L}_\mu \\ \nabla_{\lambda_E} \mathcal{L}_\mu \\ \nabla_{\lambda_I} \mathcal{L}_\mu \end{bmatrix} = \begin{bmatrix} x_2 + \lambda_{I1} - \lambda_{I2} \\ x_1 - \lambda_{I1} + \lambda_{I2} \\ -\mu s_1^{-1} + \lambda_{I1} \\ -\mu s_2^{-1} + \lambda_{I2} \\ 0 \\ x_1 - x_2 - 2 \\ -x_1 + x_2 + 1 \end{bmatrix} = 0 \qquad (4.35)$$

The solution of system (4.35) is determined by iteratively applying the Newton method to it to compute the Newton direction, then determining the step size to be taken in the Newton direction. The algorithm will also involve determining the initial values of all the parameters needed to implement the algorithm, updating the barrier parameter μ, and testing the convergence conditions.

4.3.4 Determining the search direction by the Newton method

The search direction can be computed on the basis of either the full (Newton-based) primal-dual system according to Equation (4.15), or the reduced-order system according to Equations (4.16), (4.17), (4.19) – (4.21). The reduced-order system has the advantage of requiring less computation time, and will be used to solve the example problem.

Since the problem (4.29) does not have equality constraints, the system (4.19) reduces to:

$$A\Delta x = -B \quad \Longrightarrow \quad \Delta x = -A^{-1}B \tag{4.36}$$

where **A** and **B** are given by equations (4.20) and (4.21) respectively, restated below for convenience.

$$A = \nabla^2_{xx}\mathcal{L}_\mu + \nabla h^T(x) S^{-1}\Lambda_I \nabla h(x) \tag{4.37a}$$

$$B = \nabla_x \mathcal{L}_\mu + \nabla h^T(x) S^{-1}\left(\mu \boldsymbol{e} + \Lambda_I h(x)\right) \tag{4.37b}$$

Once the increment Δx has been computed using equation (4.36), the increments Δs and $\Delta\lambda_I$ can be determined using the equations (4.16) and (4.17) respectively, also restated below for convenience:

$$\Delta s = -h(x) - s - \nabla h(x)\Delta x \tag{4.38a}$$

$$\Delta\lambda_I = S^{-1}(-S\lambda_I + \mu \boldsymbol{e} - \Lambda_I \Delta s) \tag{4.38b}$$

The components required to compute the values of **A** and **B** for the example problem can be determined as follows:

$$\nabla_x \mathcal{L}_\mu = \begin{bmatrix} x_2 + \lambda_{I1} - \lambda_{I2} \\ x_1 - \lambda_{I1} + \lambda_{I2} \end{bmatrix} \tag{4.39a}$$

$$\nabla^2_{xx}\mathcal{L}_\mu = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{4.39b}$$

$$h(x) = \begin{bmatrix} x_1 - x_2 - 2 \\ -x_1 + x_2 + 1 \end{bmatrix} \tag{4.39c}$$

$$\nabla h(x) = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{4.39d}$$

$$S = diag(s) = \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \tag{4.39e}$$

$$\Lambda_I = diag(\lambda_I) = \begin{bmatrix} \lambda_{I1} & 0 \\ 0 & \lambda_{I2} \end{bmatrix} \tag{4.39f}$$

By using equations (4.36) – (4.39), the (Newton-based) search direction for the interior-point algorithm is determined for problem (4.29).

4.3.5 Determining the step size

The step size to be taken in the Newton direction can be computed by means of Equation (4.23), as discussed in section (4.2.5), and Equation (4.22) can be used to update the primal and dual variables (for problem (4.29), these are $X = (x, s, \lambda_I)$).

4.3.6 Checking the convergence of the algorithm

Next, the convergence conditions according to equation (4.27) are computed at the current iterate to check whether the algorithm has converged to the solution of the problem (4.29).

If the algorithm has not yet converged and the predetermined maximum number of iterations has not yet been exceeded, a new Newton direction is computed and a new iterate generated, as depicted in the algorithm in Figure (4.1). At each new iteration, the barrier parameter also has to be updated, using Equation (4.26).

4.3.7 Determining the initial values of the input parameters to the algorithm

The primal-dual interior-point method has quite a number of parameters whose initial values need to be determined judiciously as they significantly impact the convergence performance of the algorithm. Following the guidelines discussed in section (4.2.8) the following initial values have been used for problem (4.29):

$$x^0 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \tag{4.40a}$$

$$s^0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{4.40b}$$

$$\mu^0 = 10 \tag{4.40c}$$

$$\lambda_I^0 = \mu S^{-1} e = 10 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \end{bmatrix} \tag{4.40d}$$

With the initial parameter values specified as above, the parameters needed to implement the algorithm are available. Other input parameters (mainly constants) are determined as discussed in section (4.2).

4.3.8 Implementation and results of the example problem

After applying the steps outlined in sections (4.3.1) – (4.3.7) to problem (4.29), the following results are obtained. As an illustration, computation of the first two iterations is detailed below, the rest of the results are given in Table 4.1.

Using the initial values in Equation (4.40), the components needed for the computation of the Newton direction according to equation (4.39) are determined as follows:

$$\nabla_{x^0}\mathcal{L}_\mu = \begin{bmatrix} x_2^0 + \lambda_{I1}^0 - \lambda_{I2}^0 \\ x_1^0 - \lambda_{I1}^0 + \lambda_{I2}^0 \end{bmatrix} = \begin{bmatrix} 1 + 10 - 10 \\ 2 - 10 + 10 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\nabla^2_{x^0 x^0}\mathcal{L}_\mu = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$h(x^0) = \begin{bmatrix} x_1^0 - x_2^0 - 2 \\ -x_1^0 + x_2^0 + 1 \end{bmatrix} = \begin{bmatrix} 2 - 1 - 2 \\ -2 + 1 + 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

$$\nabla h(x^0) = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

$$S^0 = diag(s^0) = \begin{bmatrix} s_1^0 & 0 \\ 0 & s_2^0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\Lambda_I^0 = diag(\lambda_I^0) = \begin{bmatrix} \lambda_{I1}^0 & 0 \\ 0 & \lambda_{I2}^0 \end{bmatrix} = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$$

1st iteration

Firstly, the *Newton direction* is determined as follows:

$$A^0 = \nabla^2_{x^0 x^0}\mathcal{L}_\mu + \nabla h^T(x^0)(S^0)^{-1}\Lambda_I^0 \nabla h(x^0)$$

$$= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 20 & -19 \\ -19 & 20 \end{bmatrix}$$

$$B^0 = \nabla_{x^0}\mathcal{L}_\mu + \nabla h^T(x^0)(S^0)^{-1}\left(\mu^0 \boldsymbol{e} + \Lambda_I^0 h(x^0)\right)$$

$$= \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \left(10 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\right) = \begin{bmatrix} -9 \\ 12 \end{bmatrix}$$

$$\Delta x^0 = -(A^0)^{-1}B^0 = -\begin{bmatrix} 20 & -19 \\ -19 & 20 \end{bmatrix}^{-1} \begin{bmatrix} -9 \\ 12 \end{bmatrix} = \begin{bmatrix} -1.2308 \\ -1.7692 \end{bmatrix}$$

$$\Delta s^0 = -h(x^0) - s^0 - \nabla h(x^0)\Delta x^0 = -\begin{bmatrix} -1 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} -1.2308 \\ -1.7692 \end{bmatrix}$$

$$= \begin{bmatrix} -0.5385 \\ -0.4615 \end{bmatrix}$$

$$\Delta\lambda_I^0 = (S^0)^{-1}(-S^0\lambda_I^0 + \mu^0 \boldsymbol{e} - \Lambda_I^0 \Delta s^0)$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \left(-\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \end{bmatrix} + 10 \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \begin{bmatrix} -0.5385 \\ -0.4615 \end{bmatrix}\right) = \begin{bmatrix} 5.3846 \\ 4.6154 \end{bmatrix}$$

Next, the *step length* to be taken in the Newton direction is determined as follows:

$$\alpha_p^0 = \min\left(1,\ \zeta \min_{\Delta s_i < 0}\left(-\frac{s_i}{\Delta s_i}\right)\right)$$

$$= \min\left(1,\ 0.9995 \min\left(-\frac{1}{-0.5385}, -\frac{1}{-0.4615}\right)\right) = 1$$

$$\alpha_d^0 = \min\left(1,\ \zeta \min_{\Delta\lambda_{Ii}<0}\left(-\frac{\lambda_{Ii}}{\Delta\lambda_{Ii}}\right)\right) = 1$$

$$\alpha_p^0 = \alpha_d^0 = \min\left(\alpha_p^0, \alpha_d^0\right) = \min(1,1) = 1$$

Then the primal and dual variables are updated as follows:

$$x^1 = x^0 + \alpha_p^0 \Delta x^0 = \begin{bmatrix}2\\1\end{bmatrix} - 1\begin{bmatrix}1.2308\\1.7692\end{bmatrix} = \begin{bmatrix}0.7692\\-0.7692\end{bmatrix}$$

$$s^1 = s^0 + \alpha_p^0 \Delta s^0 = \begin{bmatrix}1\\1\end{bmatrix} - 1\begin{bmatrix}0.5385\\0.4615\end{bmatrix} = \begin{bmatrix}0.4615\\0.5385\end{bmatrix}$$

$$\lambda_I^1 = \lambda_I^0 + \alpha_d^0 \Delta\lambda_I^0 = \begin{bmatrix}10\\10\end{bmatrix} + 1\begin{bmatrix}5.3846\\4.6154\end{bmatrix} = \begin{bmatrix}15.3846\\14.6154\end{bmatrix}$$

The next step is to check the convergence conditions, using equation (4.27), as follows:

$$Primal\ feasibility = \max(\ h(x^1)) = \max\left(\begin{bmatrix}0.7692 + 0.7692 - 2\\-0.7692 - 0.7692 + 1\end{bmatrix}\right) = -0.4616 \leq 0$$

$$Gradient\ condition = \frac{\|\nabla f(x^1) + \nabla h^T(x^1)\lambda_I^1\|_\infty}{1 + \|x^1\|_2 + \|\lambda_I^1\|_2} = \frac{\left\|\begin{bmatrix}-0.7692\\0.7692\end{bmatrix} + \begin{bmatrix}1 & -1\\-1 & 1\end{bmatrix}\begin{bmatrix}15.3846\\14.6154\end{bmatrix}\right\|_\infty}{1 + \left\|\begin{bmatrix}0.7692\\-0.7692\end{bmatrix}\right\|_2 + \left\|\begin{bmatrix}15.3846\\14.6154\end{bmatrix}\right\|_2}$$

$$= 1.5242e-16 \leq \varepsilon_1 = 1e-4$$

$$Complementarity\ condition = \frac{(s^1)^T\lambda_I^1}{1 + \|x\|_2} = \frac{\begin{bmatrix}0.4615 & 0.5385\end{bmatrix}\begin{bmatrix}15.3846\\14.6154\end{bmatrix}}{1 + \left\|\begin{bmatrix}0.7692\\-0.7692\end{bmatrix}\right\|_2} = 7.17 > \varepsilon_1 = 1e-4$$

$$Objective\ condition = \frac{|f(x^1) - f(x^0)|}{1 + |f(x^1)|} = \frac{|0.7692 \cdot -0.7692 - 2 \cdot 1|}{1 + |0.7692 \cdot -0.7692|} = 1.6283 > \varepsilon_2 = 1e-6$$

$$Barrier\ parameter\ condition = \mu^0 = 10 > \varepsilon_\mu = 1e-6$$

It can be observed from the results above that the *complementarity, objective function*, and *barrier parameter* conditions (i.e. tolerances) are not satisfied by the current iterate. Thus, another iterate needs to be computed, after updating the barrier parameter according to equations (4.25), (4.26) (here using $\sigma = 0.15$).

$$\mu^1 = \sigma\frac{(s^1)^T\lambda_I^1}{2p} = 0.15\frac{\begin{bmatrix}0.4615 & 0.5385\end{bmatrix}\begin{bmatrix}15.3846\\14.6154\end{bmatrix}}{4} = 0.5614$$

2nd iteration

Newton direction:

$$\nabla_{x^1}\mathcal{L}_\mu = \begin{bmatrix}x_2^1 + \lambda_{I1}^1 - \lambda_{I2}^1\\x_1^1 - \lambda_{I1}^1 + \lambda_{I2}^1\end{bmatrix} = \begin{bmatrix}-0.7692 + 15.3846 - 14.6154\\0.7692 - 15.3846 + 14.6154\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$$

$$\nabla^2_{x^1x^1}\mathcal{L}_\mu = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$$

$$h(x^1) = \begin{bmatrix} x_1^1 - x_2^1 - 2 \\ -x_1^1 + x_2^1 + 1 \end{bmatrix} = \begin{bmatrix} 0.7692 + 0.7692 - 2 \\ -0.7692 - 0.7692 + 1 \end{bmatrix} = \begin{bmatrix} -0.4616 \\ -0.5384 \end{bmatrix}$$

$$\nabla h(x^1) = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

$$S^1 = diag(s^1) = \begin{bmatrix} s_1^1 & 0 \\ 0 & s_2^1 \end{bmatrix} = \begin{bmatrix} 0.4615 & 0 \\ 0 & 0.5385 \end{bmatrix}$$

$$\Lambda_I^1 = diag(\lambda_I^1) = \begin{bmatrix} \lambda_{I1}^1 & 0 \\ 0 & \lambda_{I2}^1 \end{bmatrix} = \begin{bmatrix} 15.3846 & 0 \\ 0 & 14.6154 \end{bmatrix}$$

$$A^1 = \nabla^2_{x^1 x^1} \mathcal{L}_\mu + \nabla h^T(x^1)(S^1)^{-1}\Lambda_I^0 \nabla h(x^1)$$

$$= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}^T \begin{bmatrix} 0.4615 & 0 \\ 0 & 0.5385 \end{bmatrix}^{-1} \begin{bmatrix} 15.3846 & 0 \\ 0 & 14.6154 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 60.4762 & -59.4762 \\ -59.4762 & 60.4762 \end{bmatrix}$$

$$B^1 = \nabla_{x^1} \mathcal{L}_\mu + \nabla h^T(x^1)(S^1)^{-1}\left(\mu^1 \boldsymbol{e} + \Lambda_I^1 h(x^1)\right)$$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}^T \begin{bmatrix} 0.4615 & 0 \\ 0 & 0.5385 \end{bmatrix}^{-1} \left(0.5614 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 15.3846 & 0 \\ 0 & 14.6154 \end{bmatrix} \begin{bmatrix} -0.4616 \\ -0.5384 \end{bmatrix}\right)$$

$$= \begin{bmatrix} -0.5955 \\ 0.5955 \end{bmatrix}$$

$$\Delta x^1 = -(A^1)^{-1} B^1 = -\begin{bmatrix} 60.4762 & -59.4762 \\ -59.4762 & 60.4762 \end{bmatrix}^{-1} \begin{bmatrix} -0.5955 \\ 0.5955 \end{bmatrix}$$

$$= \begin{bmatrix} 0.005 \\ -0.005 \end{bmatrix}$$

$$\Delta s^1 = -h(x^1) - s^1 - \nabla h(x^1)\Delta x^1 = -\begin{bmatrix} -0.4616 \\ -0.5384 \end{bmatrix} - \begin{bmatrix} 0.4615 \\ 0.5385 \end{bmatrix} - \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 0.005 \\ -0.005 \end{bmatrix}$$

$$= \begin{bmatrix} -0.0099 \\ 0.0099 \end{bmatrix}$$

$$\Delta \lambda_I^1 = (S^1)^{-1}(-S^1 \lambda_I^1 + \mu^1 \boldsymbol{e} - \Lambda_I^1 \Delta s^1)$$

$$\begin{bmatrix} 0.4615 & 0 \\ 0 & 0.5385 \end{bmatrix}^{-1} \left(-\begin{bmatrix} 0.4615 & 0 \\ 0 & 0.5385 \end{bmatrix} \begin{bmatrix} 15.3846 \\ 14.6154 \end{bmatrix} + 0.5614 \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 15.3846 & 0 \\ 0 & 14.6154 \end{bmatrix} \begin{bmatrix} -0.0099 \\ 0.0099 \end{bmatrix}\right)$$

$$= \begin{bmatrix} -13.8373 \\ -13.8423 \end{bmatrix}$$

Step length determination:

$$\alpha_p^1 = \min\left(1,\ \zeta \min_{\Delta s_i < 0}\left(-\frac{s_i}{\Delta s_i}\right)\right)$$

$$= \min\left(1,\ 0.9995 \cdot \frac{0.4615}{0.0099}\right) = 1$$

$$\alpha_d^1 = \min\left(1,\ \zeta \min_{\Delta\lambda_{Ii}<0}\left(-\frac{\lambda_{Ii}}{\Delta\lambda_{Ii}}\right)\right) = 1$$

$$= \min\left(1,\quad 0.9995 \min\left(\frac{15.3846}{13.8373}, \frac{14.6154}{13.8423}\right)\right) = 1$$

$$\alpha_p^1 = \alpha_d^1 = \min(\alpha_p^1, \alpha_d^1) = \min(1,1) = 1$$

Update of primal-dual variables:

$$x^2 = x^1 + \alpha_p^1 \Delta x^1 = \begin{bmatrix} 0.7692 \\ -0.7692 \end{bmatrix} + 1 \begin{bmatrix} 0.005 \\ -0.005 \end{bmatrix} = \begin{bmatrix} 0.7742 \\ -0.7742 \end{bmatrix}$$

$$s^2 = s^1 + \alpha_p^1 \Delta s^1 = \begin{bmatrix} 0.4615 \\ 0.5385 \end{bmatrix} + 1 \begin{bmatrix} -0.0099 \\ 0.0099 \end{bmatrix} = \begin{bmatrix} 0.4516 \\ 0.5484 \end{bmatrix}$$

$$\lambda_I^2 = \lambda_I^1 + \alpha_d^1 \Delta \lambda_I^1 = \begin{bmatrix} 15.3846 \\ 14.6154 \end{bmatrix} - 1 \begin{bmatrix} 13.8373 \\ 13.8423 \end{bmatrix} = \begin{bmatrix} 1.5473 \\ 0.7731 \end{bmatrix}$$

Test of convergence:

$$Primal\ feasibility = \max(\ h(x^2)) = \max\left(\begin{bmatrix} 0.7742 + 0.7742 - 2 \\ -0.7742 - 0.7742 + 1 \end{bmatrix}\right) = -0.4516 \leq 0$$

$$Gradient\ condition = \frac{\left\|\nabla f(x^2) + \nabla h^T(x^2)\lambda_I^2\right\|_\infty}{1 + \|x^2\|_2 + \|\lambda_I^2\|_2} = \frac{\left\|\begin{bmatrix} -0.7742 \\ 0.7742 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 1.5473 \\ 0.7731 \end{bmatrix}\right\|_\infty}{1 + \left\|\begin{bmatrix} 0.7742 \\ -0.7742 \end{bmatrix}\right\|_2 + \left\|\begin{bmatrix} 1.5473 \\ 0.7731 \end{bmatrix}\right\|_2}$$

$$= 8.7e-17 \leq \varepsilon_1 = 1e-4$$

$$Complementarity\ condition = \frac{(s^2)^T\lambda_I^2}{1 + \|x^2\|_2} = \frac{\begin{bmatrix} 0.4516 & 0.5484 \end{bmatrix}\begin{bmatrix} 1.5473 \\ 0.7731 \end{bmatrix}}{1 + \left\|\begin{bmatrix} 0.7742 \\ -0.7742 \end{bmatrix}\right\|_2} = 0.5359 > \varepsilon_1$$

$$= 1e-4$$

$$Objective\ condition = \frac{|f(x^2) - f(x^1)|}{1 + |f(x^2)|} = \frac{|0.7742 \cdot -0.7742 - (0.7692 \cdot -0.7692)|}{1 + |0.7742 \cdot -0.7742|}$$

$$= 4.825e-3 > \varepsilon_2 = 1e-6$$

$$Barrier\ parameter\ condition = \mu^1 = 0.5614 > \varepsilon_\mu = 1e-6$$

Table 4.1: Simulation results of the **PDIPA** applied to problem (4.29)

Iteration	x_1	x_2	s_1	s_2	λ_{I1}	λ_{I2}	μ	$\|\nabla_x \mathcal{L}\|$	f(x)
0	2	1	1	1	10	10	10	2	2
1	0.7692	-0.7692	0.4615	0.5385	15.3846	14.6154	0.5614	2	-0.5917
2	0.7742	-0.7742	0.4516	0.5484	1.5473	0.7731	0.0421	3.5527e-15	-0.5994
3	0.8459	-0.8459	0.3083	0.6917	0.8459	0	0.0098	3.3307e-16	-0.7155
4	1	-1	0	1	1.0118	0.0118	4.43e-4	2.22e-16	-1
5	0.9998	-0.9998	0.0004	0.9996	1.0002	0.0004	3.32e-5	5.5511e-17	-0.9996
6	1	-1	0	1	1	0	2.5e-6	4.55e-15	-1

The convergence test shows that the *complementarity, objective function*, and *barrier parameter* conditions are still not satisfied by the second iterate. Thus, another iterate should be computed, after updating the barrier parameter. The algorithm converges

after about 6 iterations. The results for the rest of the iterations are displayed in Table 4.1. Figure (4.3) depicts the trajectories of the solution (x_1 , x_2) over the iterations of the algorithm, from which it can be observed that the variable x_1 settles at the optimal value of 1, and the variable x_2 settles at the optimal value of -1, as can be read from Table (4.1) as well. Figure (4.4) depicts the trajectories of the norm of the gradient of the Lagrangian function as well as the barrier parameter, and shows that both of them are driven to zero at the optimal solution, as is required by the algorithm. The MATLAB program for the PDIPA algorithm applied to problem (4.29) is presented in Appendix A.

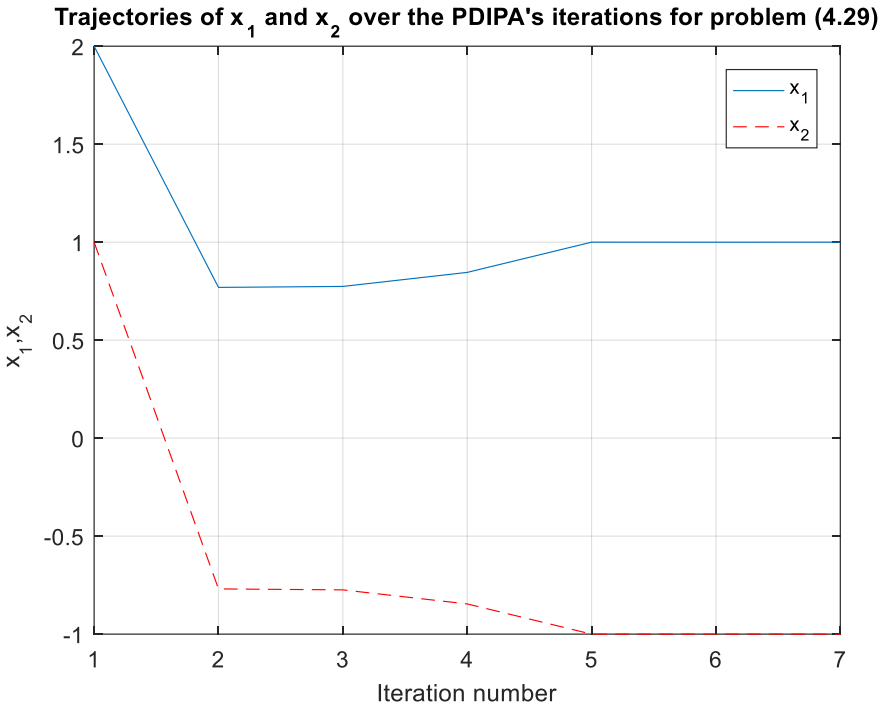

Fig. 4.3: Evolution of the variables x_1 and x_2 over the iterations of the PDIPA for problem (4.29)

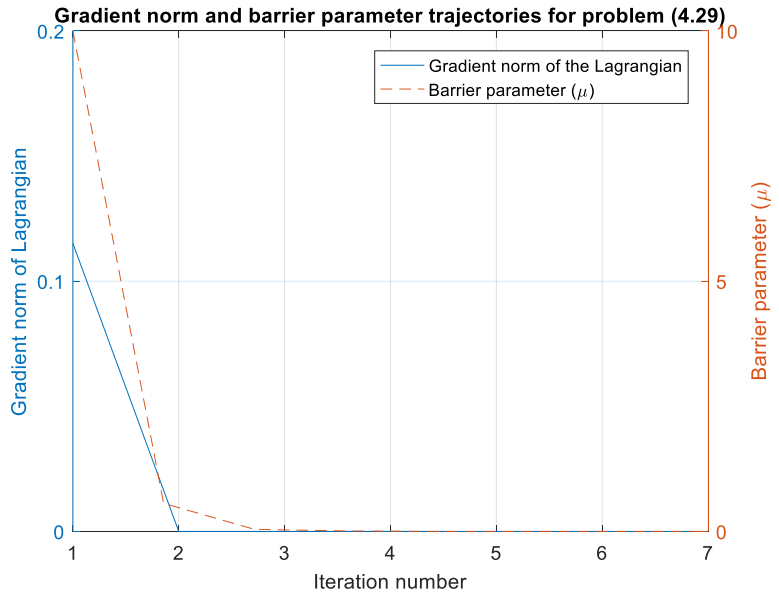

Fig. 4.4: Evolution of the norm of the gradient of the Lagrangian and of the barrier parameter for problem (4.29)

4.4 Conclusion

The primal-dual interior-point method (PDIPM) combines effective inequality constraint handling by the logarithmic barrier function and an efficient iterative search technique by the Newton method to provide one of the most efficient classical methods for large-scale constrained nonlinear optimization. The main result of this chapter has been to present a thorough, step-by-step process for the design and implementation of this algorithm. All the pertinent aspects related to the implementation of the algorithm have been discussed in great detail, encompassing the derivation of the first-order optimality (KKT) conditions and their solution by the Newton method, as well as the many parameter selection and tuning considerations that are integral to the effective implementation of the algorithm. The developed algorithm has been applied to a general nonlinear programming problem in order to demonstrate the key practical implementation aspects that have been presented in the chapter. Although the example used to illustrate the implementation of the algorithm is fairly simple, the results obtained nonetheless demonstrate the effectiveness and efficiency of the developed algorithm. The results of this chapter are used as the basis for the development and implementation of an efficient optimization strategy for the Volt/VAR optimization problem in the following chapter, namely, the primal-dual interior-point algorithm. This also constitutes one of the main deliverables of the thesis (i.e., theoretical algorithm development and design).

CHAPTER FIVE

SOLUTION OF THE VOLT/VAR OPTIMIZATION PROBLEM BY THE PRIMAL-DUAL INTERIOR POINT METHOD

5.1 Introduction

The significance of the optimal power flow (OPF) as the principal tool used by the power system operator for all aspects of power system planning and operation has been underscored in chapter 3. Volt/VAR optimization (VVO), an important variant of the OPF, is primarily concerned with the optimal coordinated dispatch of voltage-regulating devices and reactive power sources so as to maintain a secure voltage profile, and plays a key role in ensuring system security, and improving system economy by minimizing system losses (Chebbo *et al.*, 1992). Optimal reactive power dispatch (as it is otherwise referred to) plays a key role in the efficient transfer of real power, especially in the bulk power transmission system, and contributes significantly to the security, reliability, quality and economy of power system operation (Miller, 1982). In fact, reactive power-related bottlenecks are often cited as underlying the system operator's inability to economically dispatch active power, which underscores the great significance of efficiently utilizing available reactive power resources. It is also worth pointing out that the security aspect of system operation normally takes precedence over the economic aspect. Thus, in the presence of limit violations (e.g. voltages or branch flows exceeding predetermined limits), the main objective becomes the elimination of the violations (or minimizing them in case they cannot be eliminated entirely), and a minimal set of controls is sought that can be dispatched to achieve that objective. The economic aspect (i.e. loss minimization) can then be considered once the security of system operation is ensured (Martinez Ramos *et al.*, 2005).

This chapter presents the design and implementation of an efficient primal-dual interior-point method-based Volt/VAR optimization (PDIPM-VVO) algorithm, and builds on the work presented in chapter 4. The algorithm makes use of the rectangular formulation of the VVO problem (presented in chapter 3), and incorporates a Newton-Raphson-based load flow computation, which is also formulated in rectangular coordinates. The content of this chapter is organized as follows. Section 5.2 outlines in great detail the adaptation of the primal-dual interior point algorithm (PDIPA) presented in chapter 4 to the requirements of the VVO problem. The development and implementation of the Newton-Raphson load flow algorithm in rectangular coordinates is also presented in this section, prior to its incorporation into the PDIPM-VVO algorithm. Section 5.3 presents five case studies that facilitate the analysis of the newly developed PDIPM-VVO algorithm. Extensive simulations and discussion of the results in this section are presented to demonstrate

the effectiveness and efficiency of the developed algorithm. Section 5.4 concludes the chapter with a brief summary of the main outcomes of the work presented in the chapter. A pictorial representation of the content of this chapter is depicted in Figure (5.1).

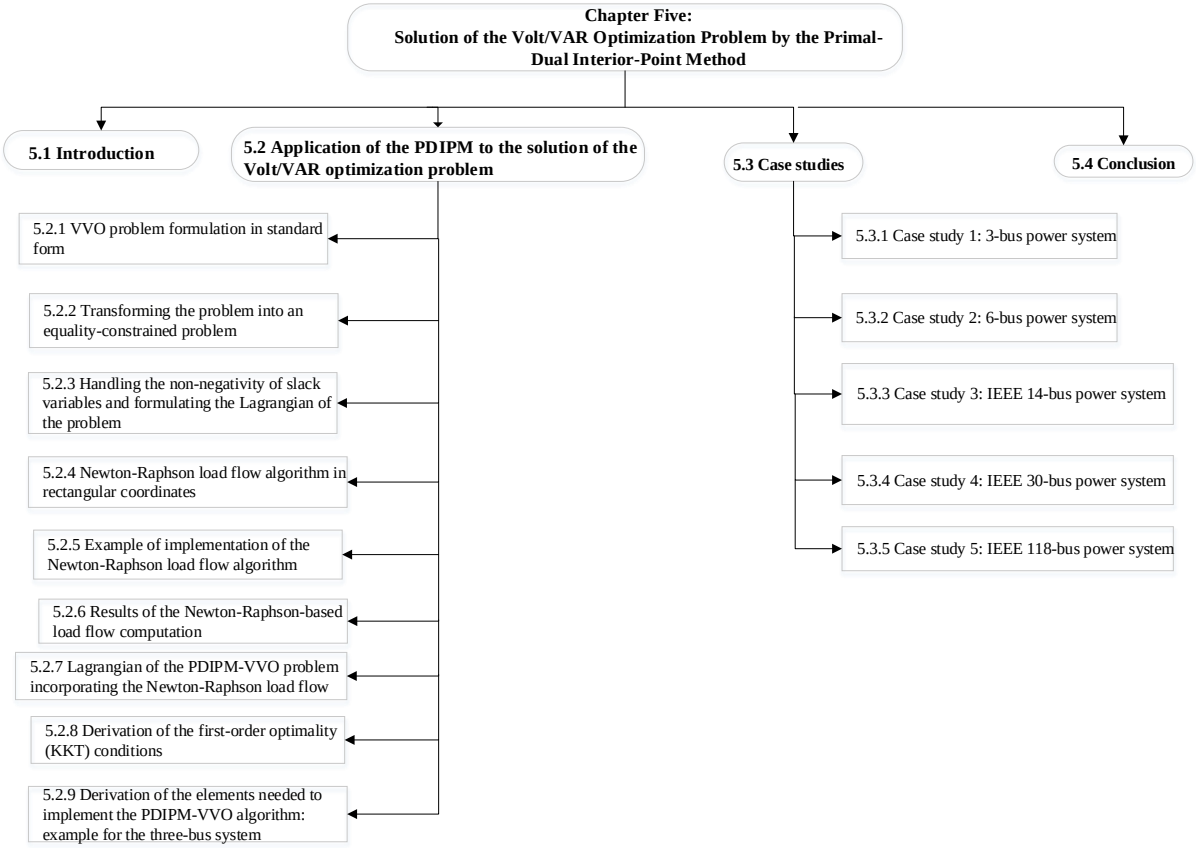

Fig. 5.1: Summary of the content covered in this chapter

The key contributions of this research as presented in this chapter are:

- Development and implementation of an efficient Newton-Raphson load flow algorithm in the rectangular coordinate representation of the system voltages
- Development and implementation of a novel efficient primal-dual interior-point algorithm for Volt/VAR optimization (PDIPM-VVO), formulated in rectangular coordinates, which incorporates the rectangular-coordinate Newton-Raphson load flow computation
- Comprehensive performance analysis of the developed PDIPM-VVO algorithm, focusing on the quality of the solution (in terms of the magnitude of real power loss percentage reduction and the voltage profile improvement) and the computational efficiency of the algorithm (in terms of the required number of iterations and runtime)
- Demonstrating the scalability of the developed algorithm by analysing its performance for test systems ranging in size from 3-bus to 118-bus system

5.2 Application of the PDIPM to the solution of the Volt/VAR optimization problem

Following the procedure illustrated in section 4.3 in the preceding chapter, development of the PDIPM-based solution algorithm for the **VVO** problem is presented in this section, making use of the rectangular form of the **VVO** problem formulation as detailed in section 3.2.3. As demonstrated in section 4.3, the procedure involves (i) expressing the optimization problem in standard form, (ii) adding slack variables to the inequality constraints so as to transform the problem into an equality-constrained problem, (iii) handling the non-negativity of slack variables by means of the logarithmic barrier function, (iv) formulating the Lagrangian of the problem, then (v) deriving the **KKT** system and subsequently (vi) solving it by the Newton method. Each of these steps is applied sequentially to the **VVO** problem in the following sub-sections.

5.2.1 **VVO** problem formulation in standard form

In standard form, the rectangular representation of the **VVO** problem can be expressed as:

$$\min P_{Loss}(e,f,t) = \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} G_{ij}\left[(e_i-e_j)^2+(f_i-f_j)^2\right] = f(x) \qquad (5.1a)$$

s.t.

$$\left.\begin{aligned} g_{Pi} &= P_i(e,f,t) + P_{di} - P_{gi} \\ g_{Qi} &= Q_i(e,f,t) + Q_{di} - q_{ci} - Q_{gi} \end{aligned}\right\} = g(x) = 0 \qquad (5.1b)$$

$$\left.\begin{aligned} h_{Vim} &= -e_i^2 - f_i^2 - \left(V_i^{min}\right)^2 \\ h_{ViM} &= e_i^2 + f_i^2 - (V_i^{max})^2 \\ h_{Qgim} &= -Q_{gi} + Q_{gi}^{min} \\ h_{QgiM} &= Q_{gi} - Q_{gi}^{max} \\ h_{qcim} &= -q_{ci} + q_{ci}^{min} \\ h_{qciM} &= q_{ci} - q_{ci}^{max} \\ h_{tim} &= -t_i + t_i^{min} \\ h_{tiM} &= t_i - t_i^{max} \end{aligned}\right\} = h(x) \le 0 \qquad (5.1c)$$

In Equation (5.1c), subscripts im and iM represent the inequality constraints corresponding to the lower and upper bounds of the i^{th} element respectively. Thus, for example, h_{Vim} is the inequality constraint corresponding to the lower bound on the voltage magnitude of the i^{th} bus. The rest of the symbols have been defined in chapter 3, section 3.2.3.

5.2.2 Transforming the problem into an equality-constrained problem

By adding slack variables to each of the inequality constraints (Equation 5.1c), the problem (5.1) is transformed into the following:

$$\min P_{Loss}(e,f,t) = \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} G_{ij}\left[(e_i-e_j)^2+(f_i-f_j)^2\right] = f(x) \qquad (5.2a)$$

s.t.

$$\left.\begin{array}{rcl} g_{Pi} & = & P_i(e,f,t)+P_{di}-P_{gi} \\ g_{Qi} & = & Q_i(e,f,t)+Q_{di}-q_{ci}-Q_{gi} \end{array}\right\} = g(x) = 0 \qquad (5.2b)$$

$$\left.\begin{array}{rcl} h_{Vim} & = & -e_i^2-f_i^2+\left(V_i^{min}\right)^2+s_{Vim} \\ h_{ViM} & = & e_i^2+f_i^2-(V_i^{max})^2+s_{ViM} \\ h_{Qgim} & = & -Q_{gi}+Q_{gi}^{min}+s_{Qgim} \\ h_{QgiM} & = & Q_{gi}-Q_{gi}^{max}+s_{QgiM} \\ h_{qcim} & = & -q_{ci}+q_{ci}^{min}+s_{qcim} \\ h_{qciM} & = & q_{ci}-q_{ci}^{max}+s_{qciM} \\ h_{tim} & = & -t_i+t_i^{min}+s_{tim} \\ h_{tiM} & = & t_i-t_i^{max}+s_{tiM} \end{array}\right\} = h(x)+s = 0 \qquad (5.2c)$$

$$s = \left[s_{Vim}, s_{ViM}, s_{Qgim}, s_{QgiM}, s_{qcim}, s_{qciM}, s_{tim}, s_{tiM}\right]^T \geq 0 \qquad (5.2d)$$

5.2.3 Handling the non-negativity of slack variables and formulating the Lagrangian of the problem

The non-negativity condition on the slack variables (Equation 5.2d) is then handled by means of a logarithmic barrier function augmented to the objective function, leading to:

$$\min f(x) - \mu\sum_{i\in\mathcal{J}} ln(s_i) \qquad (5.3a)$$

s.t.

$$\left.\begin{array}{rcl} g_{Pi} & = & P_i(e,f,t)+P_{di}-P_{gi} \\ g_{Qi} & = & Q_i(e,f,t)+Q_{di}-q_{ci}-Q_{gi} \end{array}\right\} = g(x) = 0 \qquad (5.3b)$$

$$\left.\begin{array}{rcl} h_{Vim} & = & -e_i^2-f_i^2+\left(V_i^{min}\right)^2+s_{Vim} \\ h_{ViM} & = & e_i^2+f_i^2-(V_i^{max})^2+s_{ViM} \\ h_{Qgim} & = & -Q_{gi}+Q_{gi}^{min}+s_{Qgim} \\ h_{QgiM} & = & Q_{gi}-Q_{gi}^{max}+s_{QgiM} \\ h_{qcim} & = & -q_{ci}+q_{ci}^{min}+s_{qcim} \\ h_{qciM} & = & q_{ci}-q_{ci}^{max}+s_{qciM} \\ h_{tim} & = & -t_i+t_i^{min}+s_{tim} \\ h_{tiM} & = & t_i-t_i^{max}+s_{tiM} \end{array}\right\} = h(x)+s = 0 \qquad (5.3c)$$

Where $\mathcal{J} = \{Vim, ViM, Qgim, QgiM, qcim, qciM, tim, tiM\}$, defines the set of indices for the slack variables and Lagrange multipliers associated with the inequality constraints.

At this point, the implementation of the PDIPM-based VVO solution digresses slightly from that discussed and implemented in section 4.3 for a general nonlinear programming problem, particularly in the handling of the (original) equality and inequality constraints. The reason for the digression is that after each iteration of the Newton method, the VVO algorithm needs to run a load flow algorithm in order to compute the system bus voltages (following the generator voltage set-point adjustments made by the VVO algorithm). The equality constraints (i.e. the real and reactive power balance equations, Equation 5.3b) are consequently handled by the load flow algorithm, while the PDIPM-based VVO algorithm handles the inequality constraints (Equation 5.3c). This scheme ensures the convergence of the VVO algorithm. Since the VVO algorithm makes use of the rectangular form of the problem formulation, the load flow algorithm is developed on the basis of the rectangular form of the system voltages as well, which happens to exhibit characteristics of fast convergence and high efficiency. In the following sub-section, the developed rectangular form of the Newton-Raphson load flow algorithm is presented. Other methods that are commonly applied to the load flow computation problem include Gauss-Seidel, fast-decoupled, and DC load flow methods, all of which are extensively discussed in power system analysis textbooks (see, for example, Glover & Sarma, 2002).

5.2.4 Newton-Raphson load flow algorithm in rectangular coordinates

The Newton method of solving a general nonlinear problem was discussed in section 4.2.4. The same algorithm forms the basis for the development of the Newton-Raphson load flow algorithm. The objective of a load flow computation for a power system is to determine the system bus voltages (magnitudes and phase angles) for a given generation, load and network condition, while satisfying active and reactive power balance equations (i.e. sum of active and reactive power injections at each bus, each treated separately, must equal zero). Other than handling the active and reactive power balance equations, the load flow algorithm does not enforce the satisfaction of any other system constraints (such as limits on bus voltage magnitudes), which thus needs to be taken care of by the VVO algorithm. Once the system voltages have been determined, other system quantities such as line power flows and system losses can be computed in turn as part of the load flow solution of the system.

A load flow computation requires classifying each system bus on the basis of the known and unknown variables at the bus, as detailed in Table 5.1. All system buses essentially fall broadly into two main categories, depending on whether there is generation at the bus or not. Non-generator buses are referred to as load (or PQ)

buses, and the rest are referred to as generator (or PV or regulated) buses. Among generator buses, one bus (possibly more) is selected to be the reference bus, which is responsible for setting the reference voltage phase angle for the system, as well as compensating for the mismatch between load demand (plus system losses) and scheduled generation. Hence, it is also referred to as the slack bus or swing bus.

Table 5.1: classification of system buses based on specified and unknown variables

Bus type	Voltage ($\|V\|\angle\delta$)		Real power (P)		Reactive power (Q)	
	Magnitude	Angle	Generation	Load	Generation	Load
Reference/slack/swing	Specified	Specified	Unknown	-	Unknown	-
Generator/PV/regulated	Specified	Unknown	Specified	-	Unknown	-
Load/PQ	Unknown	Unknown	-	Specified	-	Specified

As can be deduced from Table 5.1, both the voltage magnitude and voltage phase angle are specified at the reference bus, the voltage magnitude is specified at each generator bus, whereas neither voltage magnitude nor phase angle is specified at load buses. The load flow solution is thus needed to compute voltage magnitudes and phase angles for all load buses, as well as voltage phase angles for generator buses. The load flow algorithm is derived on the basis of the active and reactive power balance equations for the system, Equations (5.3a), (5.3b), (3.4) and (3.5), which can be expressed as follows for the presently considered application:

$$P_{i\Delta} = P_i + P_{di} - P_{gi}$$
$$= G_{ii}(e_i^2 + f_i^2) + e_i \sum_{j\in\mathcal{N}_i} (G_{ij}e_j - B_{ij}f_j) + f_i \sum_{j\in\mathcal{N}_i} (G_{ij}f_j + B_{ij}e_j) + P_{di} - P_{gi} \qquad (5.4a)$$

$$Q_{i\Delta} = Q_i + Q_{di} - q_{ci} - Q_{gi}$$
$$= -B_{ii}(e_i^2 + f_i^2) + f_i \sum_{j\in\mathcal{N}_i} (G_{ij}e_j - B_{ij}f_j) - e_i \sum_{j\in\mathcal{N}_i} (G_{ij}f_j + B_{ij}e_j) + Q_{di} - q_{ci} - Q_{gi} \qquad (5.4b)$$

$$V_{i\Delta}^2 = V_i^2 - (e_i^2 + f_i^2) \qquad (5.4c)$$

Equation (5.4a) is the active power mismatch (or active power balance) equation, and needs to be computed for each bus other than the slack bus. Equation (5.4b) is the reactive power mismatch equation, and needs to be computed for each load bus. Equation (5.4c) needs to be computed for each generator bus except for the slack bus, to ensure maintenance of the voltage magnitude set-point at the voltage-regulated (i.e. PV) buses. For a system with n buses, a total of $2(n-1)$ equations are formulated in order to solve for the load-bus voltage magnitudes and phase angles, as well as PV-bus voltage phase angles. The mismatch equations (5.4) have

been expressed in rectangular form, as the developed load flow algorithm is based on the rectangular representation of system bus voltages.

Similar to the computation of the Newton-based search direction for the PDIPM algorithm in section 4.2.4, the Newton-Raphson load flow algorithm is based on equation (4.11), rewritten here for ease of reference:

$$J(X)\Delta X = -F(X) \tag{5.5a}$$

where $F(X)$ is comprised of equations (4.44), $J(X)$ is the Jacobian of $F(X)$ (i.e. the first-order partial derivatives of $F(X)$ with respect to the system bus voltages), and ΔX is the correction to be applied to the variable X (the system bus voltages) in order to drive it towards the load flow solution, according to equation (4.45b).

$$X^{k+1} = X^k + \Delta X^k \tag{5.5b}$$

For each i[th] PQ bus, the mismatch vector (F_{PQi}) and the corresponding Jacobian (J_{PQij}) are given by:

$$F_{PQi} = \begin{bmatrix} P_{i\Delta} \\ Q_{i\Delta} \end{bmatrix} \tag{5.6a}$$

$$J_{PQij} = \begin{bmatrix} \frac{\partial P_{i\Delta}}{\partial e_j} & \frac{\partial P_{i\Delta}}{\partial f_j} \\ \frac{\partial Q_{i\Delta}}{\partial e_j} & \frac{\partial Q_{i\Delta}}{\partial f_j} \end{bmatrix} \tag{5.6b}$$

And for each *i[th]* PV bus, the mismatch vector (F_{PVi}) and the corresponding Jacobian (J_{PVij}) are given by:

$$F_{PVi} = \begin{bmatrix} P_{i\Delta} \\ V_{i\Delta}^2 \end{bmatrix} \tag{5.7a}$$

$$J_{PVij} = \begin{bmatrix} \frac{\partial P_{i\Delta}}{\partial e_j} & \frac{\partial P_{i\Delta}}{\partial f_j} \\ \frac{\partial V_{i\Delta}^2}{\partial e_j} & \frac{\partial V_{i\Delta}^2}{\partial f_j} \end{bmatrix} \tag{5.7b}$$

And the vector ΔX_i is given by:

$$\Delta X_i = \begin{bmatrix} \Delta e_i \\ \Delta f_i \end{bmatrix} \tag{5.8}$$

for every bus other than the slack bus.

The flowchart in Figure (5.2) outlines the Newton-Raphson load flow algorithm.

5.2.5 Example of implementation of the Newton-Raphson load flow algorithm

To illustrate the implementation of the Newton-Raphson-based load flow algorithm, a simple three-bus power system is used, whose network diagram is depicted in Figure (5.3) (adapted from Albadi, 2019).

The example three-bus system has one slack bus (bus one), one PQ bus (bus two), and one PV bus (bus three). Thus, according to equations (5.6) – (5.8), the components of equation (5.4a) can be defined as follows:

$$\Delta X = \begin{bmatrix} \Delta e_2 \\ \Delta f_2 \\ \Delta e_3 \\ \Delta f_3 \end{bmatrix} \tag{5.9a}$$

$$F(X) = \begin{bmatrix} P_{2\Delta} \\ Q_{2\Delta} \\ P_{3\Delta} \\ V^2_{3\Delta} \end{bmatrix} \tag{5.9b}$$

$$J(X) = \begin{bmatrix} \frac{\partial P_{2\Delta}}{\partial e_2} & \frac{\partial P_{2\Delta}}{\partial f_2} & \frac{\partial P_{2\Delta}}{\partial e_3} & \frac{\partial P_{2\Delta}}{\partial f_3} \\ \frac{\partial Q_{2\Delta}}{\partial e_2} & \frac{\partial Q_{2\Delta}}{\partial f_2} & \frac{\partial Q_{2\Delta}}{\partial e_3} & \frac{\partial Q_{2\Delta}}{\partial f_3} \\ \frac{\partial P_{3\Delta}}{\partial e_2} & \frac{\partial P_{3\Delta}}{\partial f_2} & \frac{\partial P_{3\Delta}}{\partial e_3} & \frac{\partial P_{3\Delta}}{\partial f_3} \\ \frac{\partial V^2_{3\Delta}}{\partial e_2} & \frac{\partial V^2_{3\Delta}}{\partial f_2} & \frac{\partial V^2_{3\Delta}}{\partial e_3} & \frac{\partial V^2_{3\Delta}}{\partial f_3} \end{bmatrix} \tag{5.9c}$$

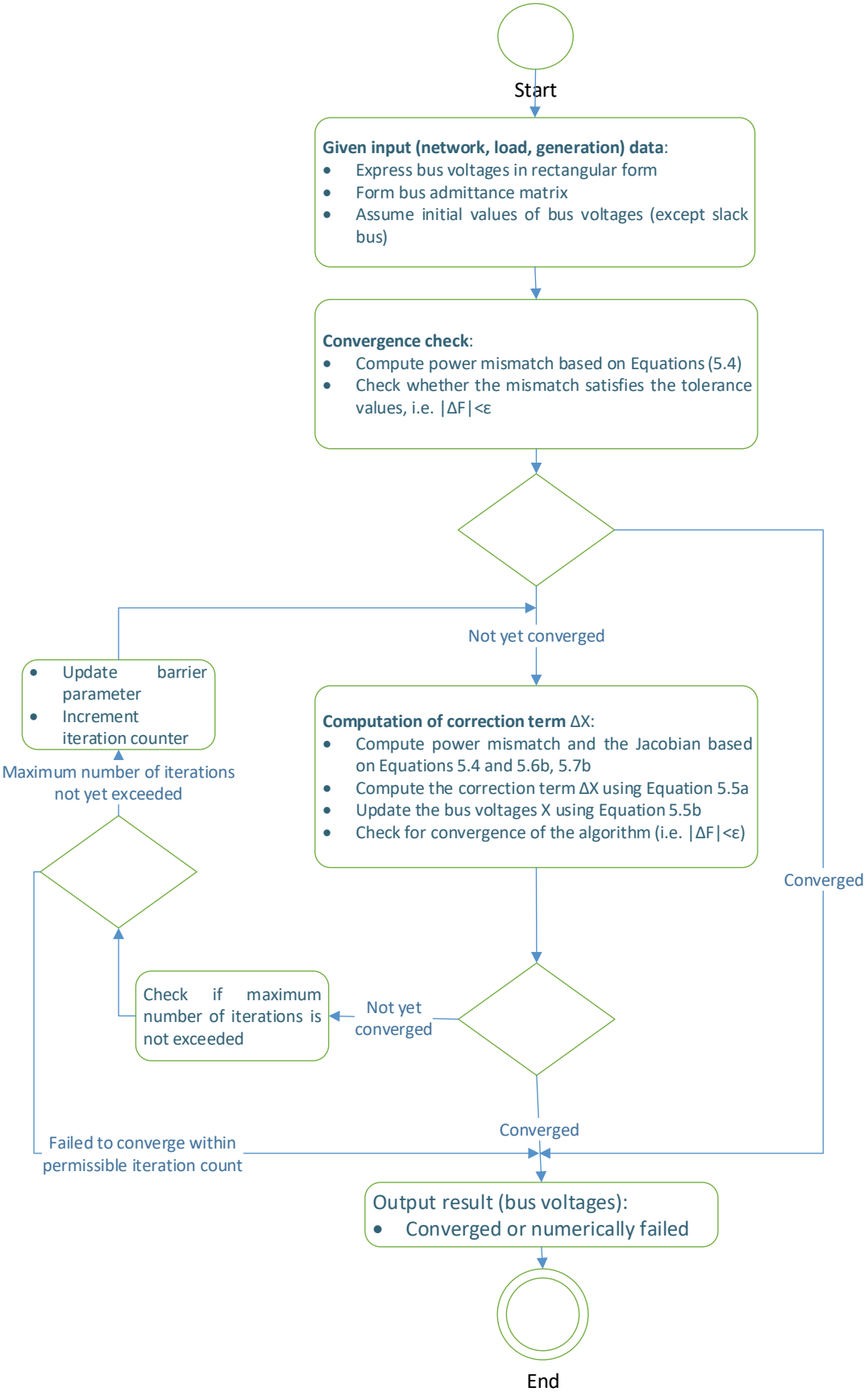

Fig. 5.2: Flowchart of the Newton-Raphson load flow algorithm

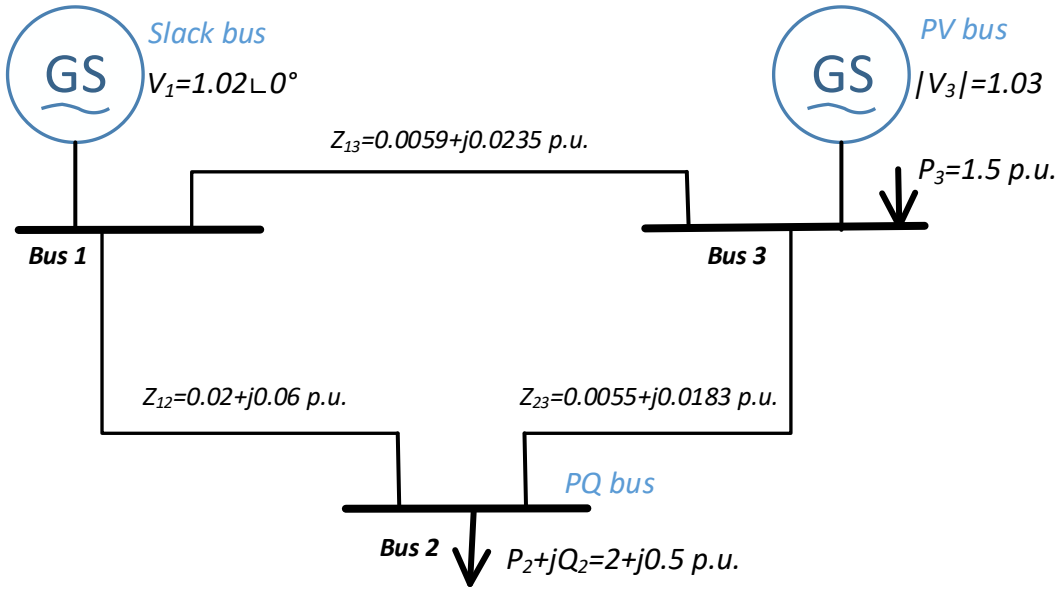

Fig. 5.3: Network diagram of the 3-bus system depicting the network data

The elements of equation (5.9b) for the system under consideration are defined as:

$$P_{2\Delta} = G_{22}\left(e_2^2 + f_2^2\right) + e_2\left(G_{21}e_1 + G_{23}e_3 - B_{21}f_1 - B_{23}f_3\right) + f_2\left(G_{21}f_1 + G_{23}f_3 + B_{21}e_1 + B_{23}e_3\right) + P_{d2}$$

$$Q_{2\Delta} = -B_{22}(e_2^2 + f_2^2) + f_2(G_{21}e_1 + G_{23}e_3 - B_{21}f_1 - B_{23}f_3) - e_2(G_{21}f_1 + G_{23}f_3 + B_{21}e_1 + B_{23}e_3) + Q_{d2}$$

$$P_{3\Delta} = G_{33}\left(e_3^2 + f_3^2\right) + e_3\left(G_{31}e_1 + G_{32}e_2 - B_{31}f_1 - B_{32}f_2\right) + f_3\left(G_{31}f_1 + G_{32}f_2 + B_{31}e_1 + B_{32}e_2\right) - P_{g3}$$

$$V_{3\Delta}^2 = V_3^2 - (e_3^2 + f_3^2)$$

The elements of equation (5.9c) are defined as:

$$\frac{\partial P_{2\Delta}}{\partial e_2} = 2G_{22}e_2 + G_{21}e_1 + G_{23}e_3 - B_{21}f_1 - B_{23}f_3$$

$$\frac{\partial P_{2\Delta}}{\partial f_2} = 2G_{22}e_2 + G_{21}f_1 + G_{23}f_3 + B_{21}e_1 + B_{23}e_3$$

$$\frac{\partial P_{2\Delta}}{\partial e_3} = G_{23}e_2 + B_{23}f_2$$

$$\frac{\partial P_{2\Delta}}{\partial f_3} = G_{23}f_2 - B_{23}e_2$$

$$\frac{\partial Q_{2\Delta}}{\partial e_2} = -2B_{22}e_2 - \left(G_{21}f_1 + G_{23}f_3 + B_{21}e_1 + B_{23}e_3\right)$$

$$\frac{\partial Q_{2\Delta}}{\partial f_2} = -2B_{22}f_2 + G_{21}e_1 + G_{23}e_3 - B_{21}f_1 - B_{23}f_3$$

$$\frac{\partial Q_{2\Delta}}{\partial e_3} = G_{23}f_2 - B_{23}e_2$$

$$\frac{\partial Q_{2\Delta}}{\partial f_3} = -G_{23}e_2 - B_{23}f_2$$

$$\frac{\partial P_{3\Delta}}{\partial e_2} = G_{32}e_3 + B_{32}f_3$$

$$\frac{\partial P_{3\Delta}}{\partial f_2} = G_{32}f_3 - B_{32}e_3$$

$$\frac{\partial P_{3\Delta}}{\partial e_3} = 2G_{33}e_3 + G_{31}e_1 + G_{32}e_2 - B_{31}f_1 - B_{32}f_2$$

$$\frac{\partial P_{3\Delta}}{\partial f_3} = 2G_{33}f_3 + G_{31}f_1 + G_{32}f_2 + B_{31}e_1 + B_{32}e_2$$

$$\frac{\partial V_{3\Delta}^2}{\partial e_2} = 0$$

$$\frac{\partial V_{3\Delta}^2}{\partial f_2} = 0$$

$$\frac{\partial V_{3\Delta}^2}{\partial e_3} = -2e_3$$

$$\frac{\partial V_{3\Delta}^2}{\partial f_3} = -2f_3$$

One of the key input data for the load flow computation is the bus admittance matrix, $Y_{ij} = G_{ij} + jB_{ij}$, which can be thought of as representing the network topology. For the system under consideration, it is determined to be (for details of its determination, reference can be made to any power systems textbook, such as Glover and Sarma, 2002):

$$Y = \begin{bmatrix} 15 - j55 & -5 + j15 & -10 + j40 \\ -5 + j15 & 20 - j65 & -15 + j50 \\ -10 + j40 & -15 + j50 & 25 - j90 \end{bmatrix} \qquad (5.10)$$

The only other component left to perform the load flow computation is to decide on an initial starting point for the (iterative) load flow algorithm. For this example, the following initial starting point is used:

$$X^0 = \begin{bmatrix} e_2^0 \\ f_2^0 \\ e_3^0 \\ f_3^0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1.03 \\ 0 \end{bmatrix} \qquad (5.11)$$

The slack-bus voltage is set to $V_1 = e_1 + jf_1 = 1.02 + j0$, and does not change throughout the load flow computation.

5.2.6 Results of the Newton-Raphson-based load flow computation

Based on the network data given in Figure (5.3) and the initial starting point as stated above, the initial mismatch vector and the Jacobian (Equations (5.9b) and (5.9c) respectively) are determined to be:

$$F(X^0) = \begin{bmatrix} P_{2\Delta}^0 \\ Q_{2\Delta}^0 \\ P_{3\Delta}^0 \\ (V_{3\Delta}^2)^0 \end{bmatrix} = \begin{bmatrix} 1.4481 \\ -1.3035 \\ -0.9310 \\ 0 \end{bmatrix}$$

$$J(X^0) = \begin{bmatrix} \frac{\partial P_{2\Delta}}{\partial e_2} & \frac{\partial P_{2\Delta}}{\partial f_2} & \frac{\partial P_{2\Delta}}{\partial e_3} & \frac{\partial P_{2\Delta}}{\partial f_3} \\ \frac{\partial Q_{2\Delta}}{\partial e_2} & \frac{\partial Q_{2\Delta}}{\partial f_2} & \frac{\partial Q_{2\Delta}}{\partial e_3} & \frac{\partial Q_{2\Delta}}{\partial f_3} \\ \frac{\partial P_{3\Delta}}{\partial e_2} & \frac{\partial P_{3\Delta}}{\partial f_2} & \frac{\partial P_{3\Delta}}{\partial e_3} & \frac{\partial P_{3\Delta}}{\partial f_3} \\ \frac{\partial V_{3\Delta}^2}{\partial e_2} & \frac{\partial V_{3\Delta}^2}{\partial f_2} & \frac{\partial V_{3\Delta}^2}{\partial e_3} & \frac{\partial V_{3\Delta}^2}{\partial f_3} \end{bmatrix}_{X=X^0} = \begin{bmatrix} 19.5108 & 66.9213 & -15.0627 & -50.1178 \\ 63.3142 & -20.6146 & -50.1178 & 15.0627 \\ -15.5146 & -51.6213 & 26.4186 & 90.9483 \\ 0 & 0 & -2.06 & 0 \end{bmatrix}$$

Equations (4.45a) and (4.45b) can then be used to update the vector X as follows:

$$\Delta X^0 = -J(X^0)^{-1}F(X^0) = -\begin{bmatrix} 19.5108 & 66.9213 & -15.0627 & -50.1178 \\ 63.3142 & -20.6146 & -50.1178 & 15.0627 \\ -15.5146 & -51.6213 & 26.4186 & 90.9483 \\ 0 & 0 & -2.06 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 1.4481 \\ -1.3035 \\ -0.9310 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0.01235 \\ -0.02782 \\ 0 \\ -0.00345 \end{bmatrix}$$

$$X^1 = X^0 + \Delta X^0 = \begin{bmatrix} 1 \\ 0 \\ 1.03 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.01235 \\ -0.02782 \\ 0 \\ -0.00345 \end{bmatrix} = \begin{bmatrix} 1.01235 \\ -0.02782 \\ 1.03 \\ -0.00345 \end{bmatrix}$$

Performing another two iterations leads to convergence of the algorithm. The results for the rest of the iterations are tabulated in Tables (5.2) and (5.3). The MATLAB program for this example is presented in Appendix B.

Table 5.2: 3-bus system change in bus power/voltage mismatch over iterations of the Newton-Raphson load flow computations

Iteration number	$P_{2\Delta}$	$Q_{2\Delta}$	$P_{3\Delta}$	$V_{3\Delta}^2$
0	1.4481	-1.3035	-0.9310	0
1	0.01928	0.0549	-0.0033	-1.12e-5
2	5.57e-7	5.2e-6	1e-5	-4.92e-8
3	1.28e-12	4.1e-11	1.16e-11	7.2e-14

Table 5.3: 3-bus system change in bus voltages over iterations of the Newton-Raphson load flow computations

	Bus 2	***Bus 3***
Iteration number	$e_2 + jf_2$	$e_3 + jf_3$
1	1.0123-j0.0278	1.03-j0.0034
2	1.0115-j0.028	1.03-j0.00367
3	1.0115-j0.028	1.03-j0.00367

As mentioned earlier, the developed Newton-Raphson load flow algorithm exhibits very fast convergence and high computational efficiency, which is very desirable for Volt/VAR optimization, since each iteration of the VVO algorithm requires a load flow computation as well. In the next sub-section, the implementation of the VVO algorithm incorporating the developed Newton-Raphson load flow algorithm is further discussed, particularly the formulation of the Lagrangian function for the problem, and subsequently derivation of the first-order optimality (KKT) conditions for the system. The PDIPM-VVO algorithm incorporating the Newton-Raphson load flow computation is depicted in the flowchart in Figure (5.4).

5.2.7 Lagrangian of PDIPM-VVO problem incorporating the Newton-Raphson load flow

With the (original) equality constraints (the active and reactive power balance equations, Equation 5.3b) incorporated into the Newton-Raphson load flow algorithm, the Lagrangian function of the VVO problem is formulated on the basis of only the logarithmic barrier function-augmented objective function (Equation 5.3a) and the

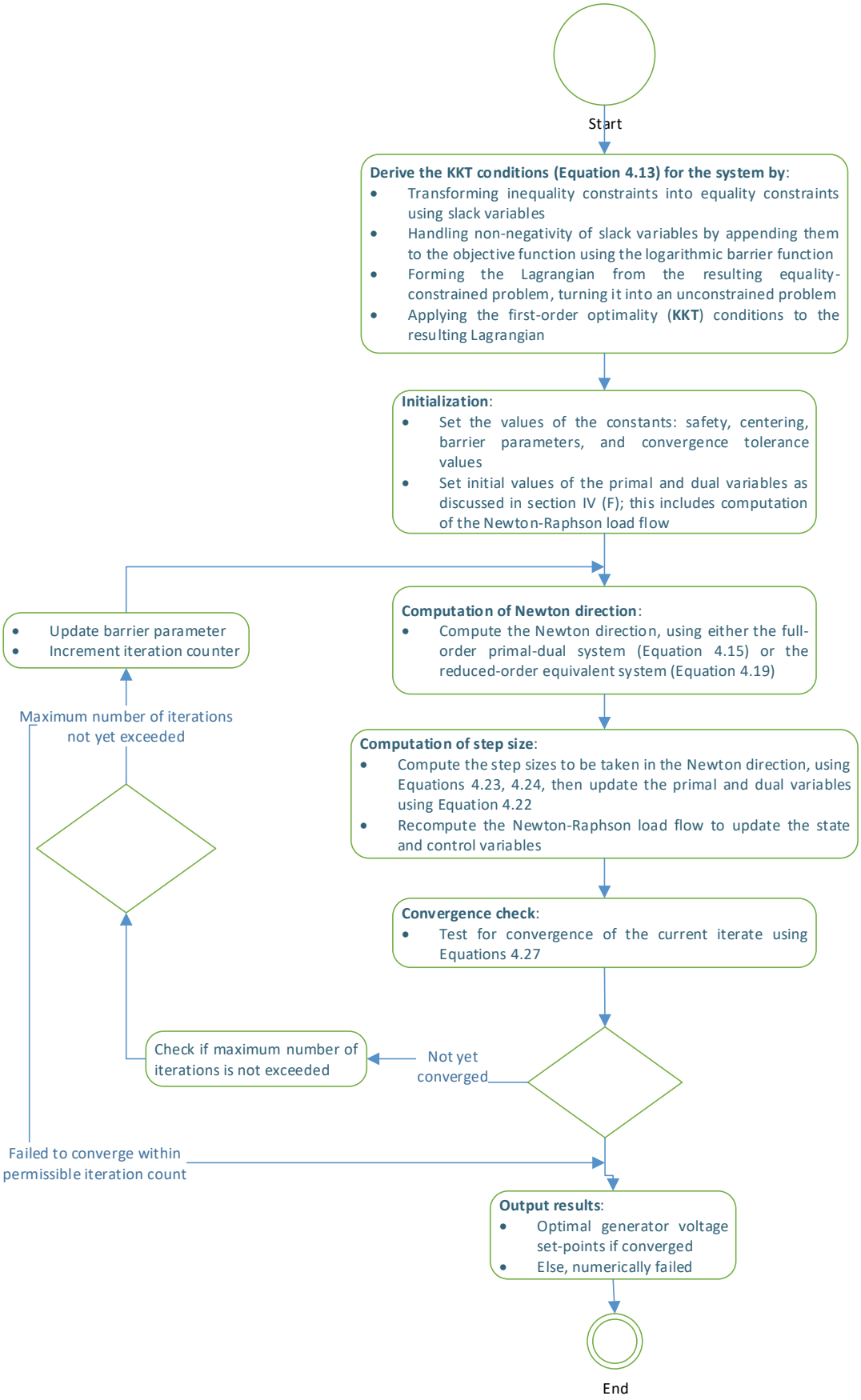

Fig. 5.4: Flowchart of the PDIPM-VVO algorithm incorporating the Newton-Raphson load flow computation

inequality constraints transformed into equality constraints by addition of slack variables (Equation 5.3c), resulting in:

$$\mathcal{L}_\mu = f(x) - \mu \sum_{i \in \mathcal{I}} \ln(s_i) + \lambda_I^T (h(x) + s) \tag{5.12}$$

The Lagrangian multiplier vector λ_I can be defined as:

$$\lambda_I = \left[\lambda_{IVim}, \lambda_{IViM}, \lambda_{IQgim}, \lambda_{IQgiM}, \lambda_{Iqcim}, \lambda_{IqciM}, \lambda_{Itim}, \lambda_{ItiM}\right]^T$$

The full expression of the Lagrangian function takes the form as given by equation (5.13).

$$\begin{aligned}\mathcal{L}_\mu = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} & G_{ij}\left[(e_i - e_j)^2 + (f_i - f_j)^2\right] \\ & - \mu\left[\ln s_{Vim} \right. \\ & \left. + \ln s_{ViM} + \ln s_{Qgim} + \ln s_{QgiM} + \ln s_{qcim} + \ln s_{qciM} + \ln s_{tim} + \ln s_{tiM}\right] \\ & + \lambda_{IVim}\left(-e_i^2 - f_i^2 + \left(V_i^{min}\right)^2 + s_{Vim}\right) + \lambda_{IViM}\left(e_i^2 + f_i^2 - (V_i^{max})^2 + s_{ViM}\right) \\ & + \lambda_{IQgim}\left(-Q_{gi} + Q_{gi}^{min} + s_{Qgim}\right) + \lambda_{IQgiM}\left(Q_{gi} - Q_{gi}^{max} + s_{QgiM}\right) \\ & + \lambda_{Iqcim}\left(-q_{ci} + q_{ci}^{min} + s_{qcim}\right) + \lambda_{IqciM}\left(q_{ci} - q_{ci}^{max} + s_{qciM}\right) \\ & + \lambda_{Itim}\left(-t_i + t_i^{min} + s_{tim}\right) \\ & + \lambda_{ItiM}\left(t_i - t_i^{max} + s_{tiM}\right)\end{aligned} \tag{5.13}$$

5.2.8 Derivation of the first-order optimality (**KKT**) conditions

The first-order optimality (**KKT**) conditions are given by equation (4.6) as derived in section (4.2.3), and can be stated for the **VVO** problem, based on the Lagrangian function for the problem (Equation 5.13), as:

$$\nabla_x \mathcal{L}_\mu = \begin{bmatrix} \nabla_e \mathcal{L}_\mu \\ \nabla_f \mathcal{L}_\mu \\ \nabla_t \mathcal{L}_\mu \\ \nabla_{qc} \mathcal{L}_\mu \end{bmatrix} = \begin{bmatrix} \nabla_e f(x) + \nabla_e h^T(x) \lambda_I \\ \nabla_f f(x) + \nabla_f h^T(x) \lambda_I \\ \nabla_t f(x) + \nabla_t h^T(x) \lambda_I \\ \nabla_{qc} f(x) + \nabla_{qc} h^T(x) \lambda_I \end{bmatrix} = 0 \tag{5.14a}$$

$$\nabla_s \mathcal{L}_\mu = \begin{bmatrix} \nabla_{sVim} \mathcal{L}_\mu \\ \nabla_{sViM} \mathcal{L}_\mu \\ \nabla_{sQgim} \mathcal{L}_\mu \\ \nabla_{sQgiM} \mathcal{L}_\mu \\ \nabla_{sqcim} \mathcal{L}_\mu \\ \nabla_{sqciM} \mathcal{L}_\mu \\ \nabla_{stim} \mathcal{L}_\mu \\ \nabla_{stiM} \mathcal{L}_\mu \end{bmatrix} = \begin{bmatrix} s_{Vim} \lambda_{IVim} - \mu \\ s_{ViM} \lambda_{IViM} - \mu \\ s_{Qgim} \lambda_{IQgim} - \mu \\ s_{QgiM} \lambda_{IQgiM} - \mu \\ s_{qcim} \lambda_{Iqcim} - \mu \\ s_{qciM} \lambda_{IqciM} - \mu \\ s_{tim} \lambda_{Itim} - \mu \\ s_{tiM} \lambda_{ItiM} - \mu \end{bmatrix} = 0 \tag{5.14b}$$

$$\nabla_{\lambda_I}\mathcal{L}_\mu = \begin{bmatrix} \nabla_{\lambda_{IVim}}\mathcal{L}_\mu \\ \nabla_{\lambda_{IViM}}\mathcal{L}_\mu \\ \nabla_{\lambda_{IQgim}}\mathcal{L}_\mu \\ \nabla_{\lambda_{IQgiM}}\mathcal{L}_\mu \\ \nabla_{\lambda_{Iqcim}}\mathcal{L}_\mu \\ \nabla_{\lambda_{IqciM}}\mathcal{L}_\mu \\ \nabla_{\lambda_{Itim}}\mathcal{L}_\mu \\ \nabla_{\lambda_{ItiM}}\mathcal{L}_\mu \end{bmatrix} = \begin{bmatrix} h_{Vim} \\ h_{ViM} \\ h_{Qgim} \\ h_{QgiM} \\ h_{qcim} \\ h_{qciM} \\ h_{tim} \\ h_{tiM} \end{bmatrix} = \begin{bmatrix} -e_i^2 - f_i^2 + \left(V_i^{min}\right)^2 + s_{Vim} \\ e_i^2 + f_i^2 + (V_i^{max})^2 + s_{ViM} \\ -Q_{gi} + Q_{gi}^{min} + s_{Qgim} \\ Q_{gi} - Q_{gi}^{max} + s_{QgiM} \\ -q_{ci} + q_{ci}^{min} + s_{qcim} \\ q_{ci} - q_{ci}^{max} + s_{qciM} \\ -t_i + t_i^{min} + s_{tim} \\ t_i - t_i^{max} + s_{tiM} \end{bmatrix} = 0 \qquad (5.14c)$$

On the basis of the **KKT** system (5.14), the full-order primal-dual system can be derived, as given by Equation (4.15). For reasons of computational efficiency (as discussed earlier), the reduced-order primal-dual system (according to Equations (4.16), (4.17), (4.19) – (4.21)) is used as the basis for the implementation of the PDIPM-based VVO solution. Moreover, since the equality constraints ($g(x)$) are no longer part of the VVO problem (as they are handled by the Newton-Raphson load flow algorithm), the algorithm is implemented in a similar manner to the example presented in section 4.3; that is, on the basis of the reduced-order system according to equations (4.36) – (4.38), rewritten here for ease of reference.

$$A = \nabla_{xx}^2\mathcal{L}_\mu + \nabla h^T(x)S^{-1}\Lambda_I\nabla h(x) \qquad (5.15a)$$

$$B = \nabla_x\mathcal{L}_\mu + \nabla h^T(x)S^{-1}\left(\mu\boldsymbol{e} + \Lambda_I h(x)\right) \qquad (5.15b)$$

$$\Delta x = -A^{-1}B \qquad (5.15c)$$

$$\Delta s = -h(x) - s - \nabla h(x)\Delta x \qquad (5.15d)$$

$$\Delta\lambda_I = S^{-1}(-S\lambda_I + \mu\boldsymbol{e} - \Lambda_I\Delta s) \qquad (5.15e)$$

$$\Delta\lambda_I = S^{-1}(-S\lambda_I + \mu\boldsymbol{e} - \Lambda_I\Delta s) \qquad (4.55e)$$

The derivation of the components in Equations (5.15) as are needed to solve the PDIPM-based VVO problem is demonstrated by means of the simple three-bus power system (also used in section 4.4.5) in the following sub-section.

5.2.9 Derivation of elements needed to implement the PDIPM-VVO algorithm: example for the three-bus system

The main elements needed to compute the PDIPM-based VVO solution according to Equations (5.15) are the gradient and Hessian of the Lagrangian function, the Jacobian and Hessian of the inequality constraints, the diagonal matrix of the slack variables, and the diagonal matrix of the equality-constraint Lagrangian multipliers, as can be deduced from Equation (5.15). Derivation of these components is illustrated in this section for the example system under consideration. The same can be done for systems of arbitrary size, although the process becomes fairly tedious for larger systems.

The objective function ($f(x) \in R^1$), its gradient ($\nabla_x f(x) \in R^k$) and its Hessian ($\nabla^2_{xx} f(x) \in R^{k \times k}$, where k is the dimension of x) are expressed in equations (5.16). Here the decision vector is taken to comprise of the generator terminal voltages (i.e. $x = [e_i,\ f_i]$ for every generator bus, where f_1 is fixed at zero, which corresponds to a slack-bus voltage phase angle reference of $0°$).

$$f(x) = f_{12} + f_{13} + f_{23}$$
$$= G_{12}[(e_1 - e_2)^2 + (f_1 - f_2)^2] + G_{13}[(e_1 - e_3)^2 + (f_1 - f_3)^2] + G_{23}[(e_2 - e_3)^2 + (f_2 - f_3)^2] \quad (5.16a)$$

$$\nabla_x f(x) = 2. \begin{bmatrix} G_{12}(e_1 - e_2) + G_{13}(e_1 - e_3) \\ -\left(G_{13}(e_1 - e_3) + G_{23}(e_2 - e_3)\right) \\ -\left(G_{13}(f_1 - f_3) + G_{23}(f_2 - f_3)\right) \end{bmatrix} \quad (5.16b)$$

$$\nabla^2_{xx} f(x) = 2. \begin{bmatrix} G_{12} + G_{13} & -G_{13} & 0 \\ -G_{13} & G_{13} + G_{23} & 0 \\ 0 & 0 & G_{13} + G_{23} \end{bmatrix} \quad (5.16c)$$

The (inequality) constraint function ($h(x)$), its Jacobian ($\nabla_x h(x)$) and its Hessian (actually, the Jacobian of $\nabla_x h(x)$ transposed and multiplied with the Lagrangian multiplier vector, that is, $\nabla^2_{xx} h^T(x)\lambda_I$) are expressed in equations (5.17). The constraints comprise the bus voltage magnitudes (as functional constraints) and the limits on generator reactive power outputs.

$$h(x) = \begin{bmatrix} h_{V1m} \\ h_{V1M} \\ h_{Qg1m} \\ h_{Qg1M} \\ h_{V2m} \\ h_{V2M} \\ h_{V3m} \\ h_{V3M} \\ h_{Qg3m} \\ h_{Qg3M} \end{bmatrix} = \begin{bmatrix} -e_1^2 - f_1^2 + (V_1^m)^2 \\ e_1^2 + f_1^2 - (V_1^M)^2 \\ -Q_{g1} + Q_{g1}^m \\ Q_{g1} - Q_{g1}^M \\ -e_2^2 - f_2^2 + (V_2^m)^2 \\ e_2^2 + f_2^2 - (V_2^M)^2 \\ -e_3^2 - f_3^2 + (V_3^m)^2 \\ e_3^2 + f_3^2 - (V_3^M)^2 \\ -Q_{g3} + Q_{g3}^m \\ Q_{g3} - Q_{g3}^M \end{bmatrix} \quad (5.17a)$$

$$\nabla_x h(x) = \begin{bmatrix} \frac{\partial h_{V1m}}{\partial e_1} & \frac{\partial h_{V1m}}{\partial e_3} & \frac{\partial h_{V1m}}{\partial f_3} \\ \frac{\partial h_{V1M}}{\partial e_1} & \frac{\partial h_{V1M}}{\partial e_3} & \frac{\partial h_{V1M}}{\partial f_3} \\ \frac{\partial h_{Qg1m}}{\partial e_1} & \frac{\partial h_{Qg1m}}{\partial e_3} & \frac{\partial h_{Qg1m}}{\partial f_3} \\ \frac{\partial h_{Qg1M}}{\partial e_1} & \frac{\partial h_{Qg1M}}{\partial e_3} & \frac{\partial h_{Qg1M}}{\partial f_3} \\ \frac{\partial h_{V2m}}{\partial e_1} & \frac{\partial h_{V2m}}{\partial e_3} & \frac{\partial h_{V2m}}{\partial f_3} \\ \frac{\partial h_{V2M}}{\partial e_1} & \frac{\partial h_{V2M}}{\partial e_3} & \frac{\partial h_{V2M}}{\partial f_3} \\ \frac{\partial h_{V3m}}{\partial e_1} & \frac{\partial h_{V3m}}{\partial e_3} & \frac{\partial h_{V3m}}{\partial f_3} \\ \frac{\partial h_{V3M}}{\partial e_1} & \frac{\partial h_{V3M}}{\partial e_3} & \frac{\partial h_{V3M}}{\partial f_3} \\ \frac{\partial h_{Qg3m}}{\partial e_1} & \frac{\partial h_{Qg3m}}{\partial e_3} & \frac{\partial h_{Qg3m}}{\partial f_3} \\ \frac{\partial h_{Qg3M}}{\partial e_1} & \frac{\partial h_{Qg3M}}{\partial e_3} & \frac{\partial h_{Qg3M}}{\partial f_3} \end{bmatrix} = \begin{bmatrix} -2e_1 & 0 & 0 \\ 2e_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -2e_3 & -2f_3 \\ 0 & 2e_3 & 2f_3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (5.17b)$$

$$\nabla_{xx}^2 h^T(x) = 2.\begin{bmatrix} \lambda_{I2} - \lambda_{I1} & 0 & 0 \\ 0 & \lambda_{I8} - \lambda_{I7} & 0 \\ 0 & 0 & \lambda_{I8} - \lambda_{I7} \end{bmatrix} \quad (5.17c)$$

The gradient and Hessian of the Lagrangian function are defined on the basis of Equations (5.16) and (5.17) as follows (modified forms of equations (4.5a) and (4.14) respectively):

$$\nabla_x \mathcal{L}_\mu = \nabla f(x) + \nabla h^T(x)\lambda_I \quad (5.18a)$$

$$\nabla_{xx}^2 \mathcal{L}_\mu = \nabla_{xx}^2 f(x) + \nabla_{xx}^2 h^T(x)\lambda_I \quad (5.18b)$$

Based on Equations (5.16) – (5.18), and by defining the diagonal matrices S and Lambda as done in section 4.3.8, the PDIPM-based VVO solution algorithm according equations (5.15) can be implemented. Other steps of the implementation (e.g. initialization, step size determination, checking for convergence, and updating of the barrier parameter) are as outlined in example presented in section 4.3.

5.3 Case studies

To evaluate the performance of the developed PDIPM-based VVO algorithm, a number of case studies have been performed. The case studies are based on the 3-bus system used in sections 5.2.5 and 5.2.9 to demonstrate the implementation of the Newton-Raphson and PDIPM-based VVO algorithms respectively, a 6-bus power system, and the IEEE 14-bus, IEEE 30-bus, and IEEE 118-bus power systems. The

choice of the case studies enables the demonstration of the scalability and efficiency of the algorithm independently of the system size (as it relates to the number of buses in the system). As part of the performance analysis of the algorithm, the following aspects are of particular interest:

- Magnitude of loss minimization
- Voltage profile improvement due to the Volt/VAR optimization
- Efficiency and speed of convergence of the algorithm, measured by the number of iterations taken for the algorithm to converge, and the elapsed time
- Impact of generator reactive power output variation on both the power loss minimization and the voltage profile improvement

The Matlab programs for the case studies presented in the following sub-sections are appended in Appendix B.

5.3.1 Case study 1: 3-bus power system

As already stated, the 3-bus power system has been presented in sections 5.2.5 and 5.2.9, in connection with the details of implementation of the Newton-Raphson and PDIPM-based VVO algorithms respectively. The corresponding network diagram is depicted in Figure (5.3). The data needed to perform the Volt/VAR optimization (i.e. network, load and generation data) appears in Figure (5.3), and is presented Appendix B as well. The analysis in this sub-section follows the points outlined at the end of the previous sub-section. That is, focus is on the improvement in the power system losses and the voltage profile of the power system, the efficiency of the algorithm in arriving at the solution, as well as a number of other qualitative aspects of the solution, such as the impact of the variation of generator reactive power output on both the real power loss minimization and voltage profile improvement.

Table (5.4) compares the initial generator bus voltage magnitudes with the corresponding optimal values following the execution of the PDIPM-based VVO algorithm. The same information is depicted in Figure (5.5) in the form of a bar chart, which shows that voltage magnitudes increase on all buses as a result of the Volt/VAR optimization.

Table 5.4: 3-bus system generator voltage magnitudes prior to and following VVO

Control variable	*Initial value*	*Optimal value*	*Lower limit*	*Upper limit*
V_{g1}	1.02	1.0281	0.95	1.1
V_{g3}	1.03	1.034	0.95	1.1

Table 5.5: 3-bus system loss reduction prior to and following VVO

	Initial	*Final*	*Percentage loss reduction*
Real power system losses (p.u.)	0.0195	0.0180	7.89%
Number of iterations	4		
Execution time (sec)	0.0843		

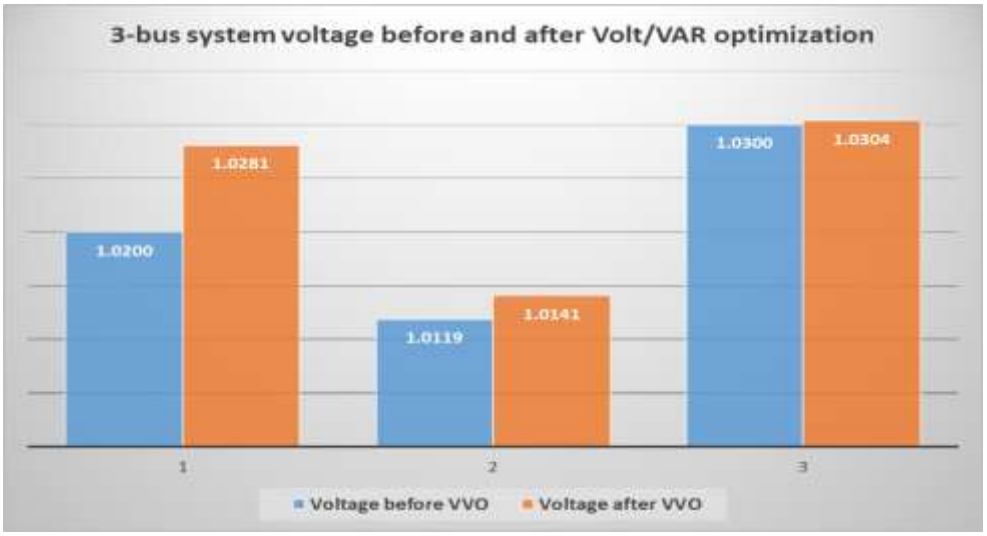

Fig. 5.5: 3-bus system generator voltage magnitudes before and after Volt/VAR optimization in bar chart form

As depicted in Table (5.5), executing the VVO algorithm results in a *7.89 percent* reduction in system real power losses. The algorithm converges within *four iterations*, and takes about *84.3 milliseconds* to execute (which includes running the Newton-Raphson load flow algorithm at each iteration of the VVO algorithm). Figure (5.6) depicts the real power loss change over the iterations of the VVO algorithm, and shows that by the third iteration, there is no appreciable change in the real power loss, thus demonstrating very fast convergence of the algorithm for this case study.

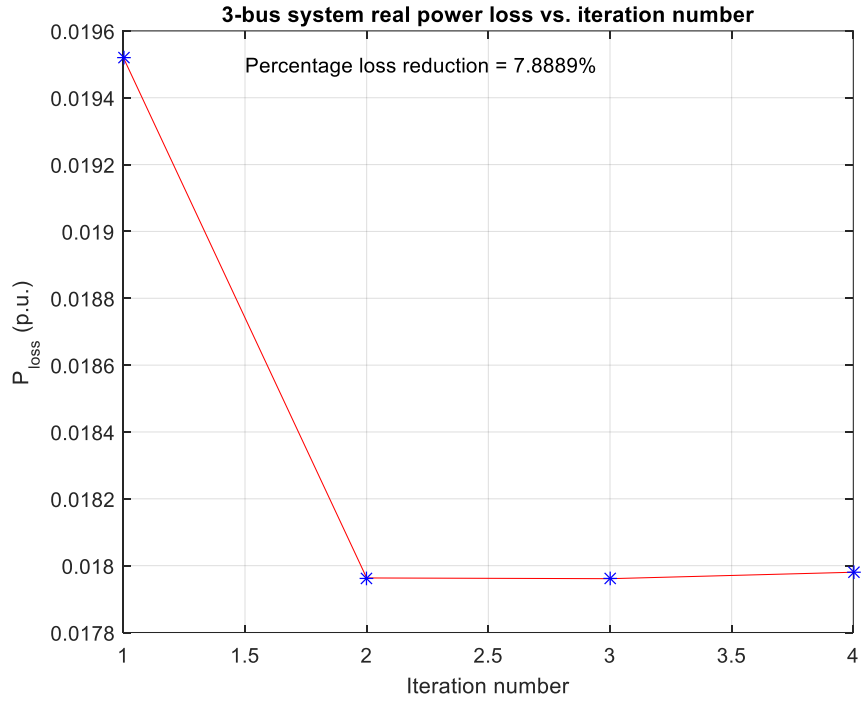

Fig. 5.6: 3-bus system real power losses plotted against the iteration number

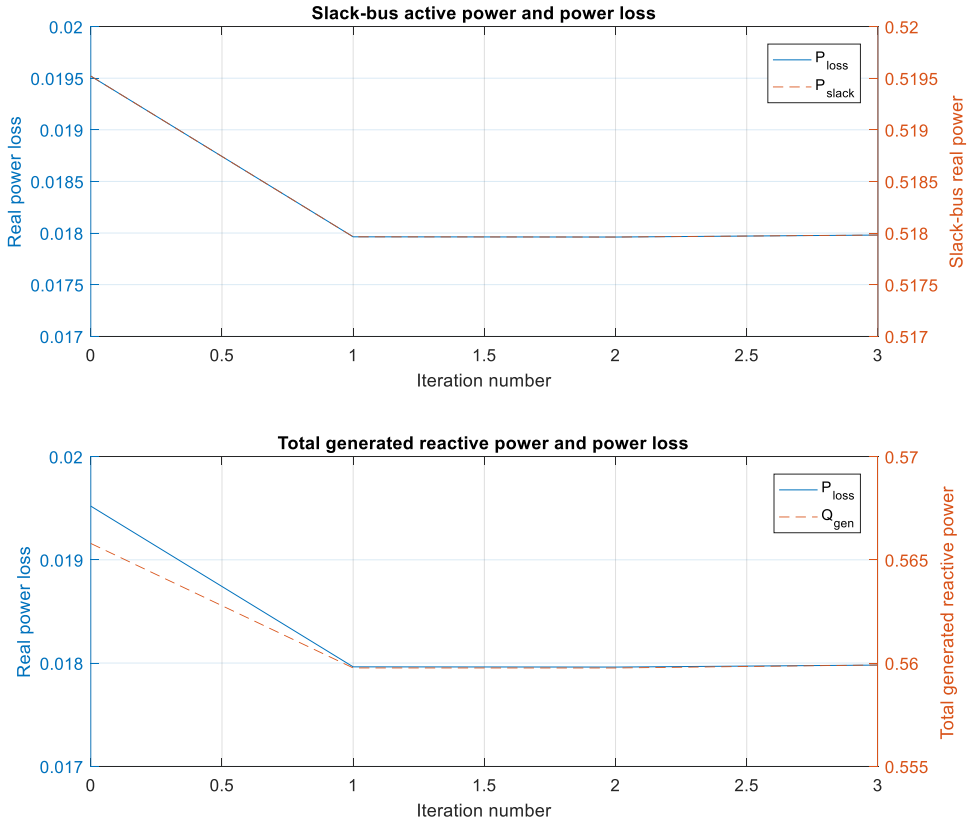

Fig. 5.7: 3-bus system comparison of real power loss with slack-bus active power (top plot) and with total generated reactive power (bottom plot)

An interesting comparison is depicted in Figure (5.7) between the real power loss trajectory over the iterations of the algorithm, and the slack-bus active power as well as

the total generated reactive power in the top and bottom plots respectively. The top plot shows that the real power loss curve coincides with that of the slack-bus active power, demonstrating that the real power loss reduction due to the Volt/VAR optimization corresponds to the reduction in the slack-bus active power output. The bottom plot also shows that the system reactive power generation tracks the real power loss reduction, implying that real power loss minimization simultaneously leads to decrease in reactive power generation as well.

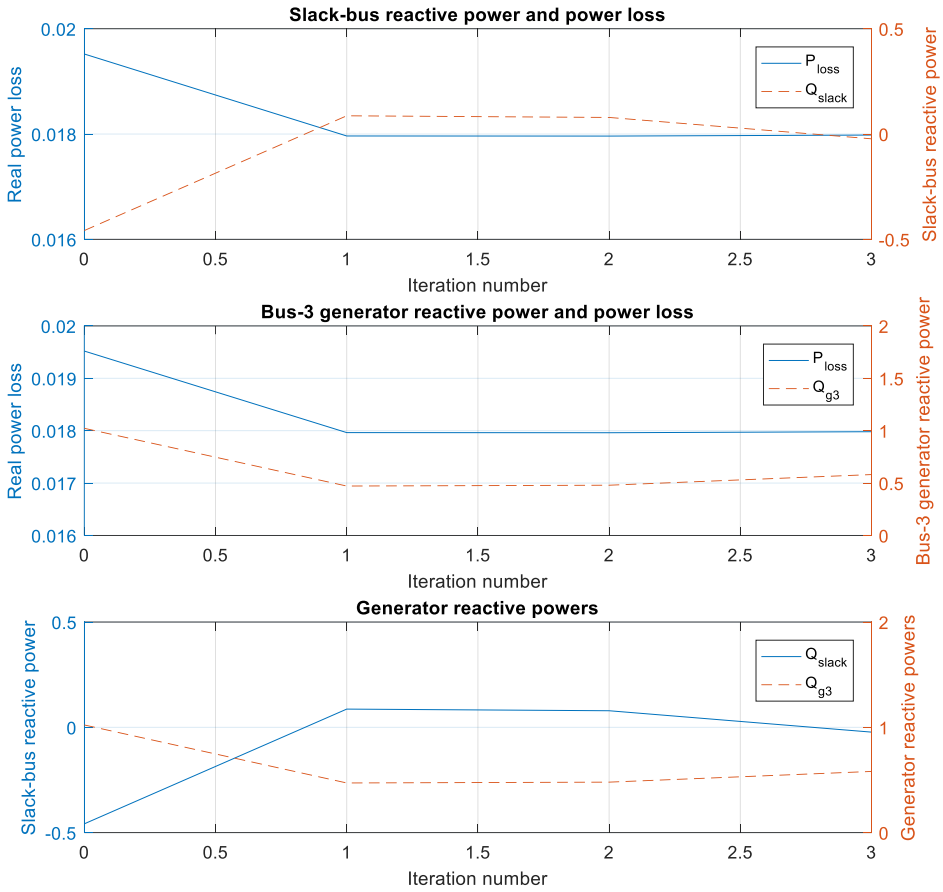

Fig. 5.8: 3-bus system comparison of change in each generator's reactive power output with change in real power loss

This is the total (or net) system reactive power generation though, as Figure (5.8) shows that slack-bus reactive power output increases while bus-3 generator reactive power output decreases over the iterations of the VVO algorithm. In Figure (5.9), the change in reactive power output of each generator is compared with the corresponding change in its terminal voltage, showing a proportional relationship between the two quantities (that is, increase in generator terminal voltage magnitude leads to corresponding increase in reactive power output), which confirms the well-established theory of power system operation (Glover & Sarma, 2002).

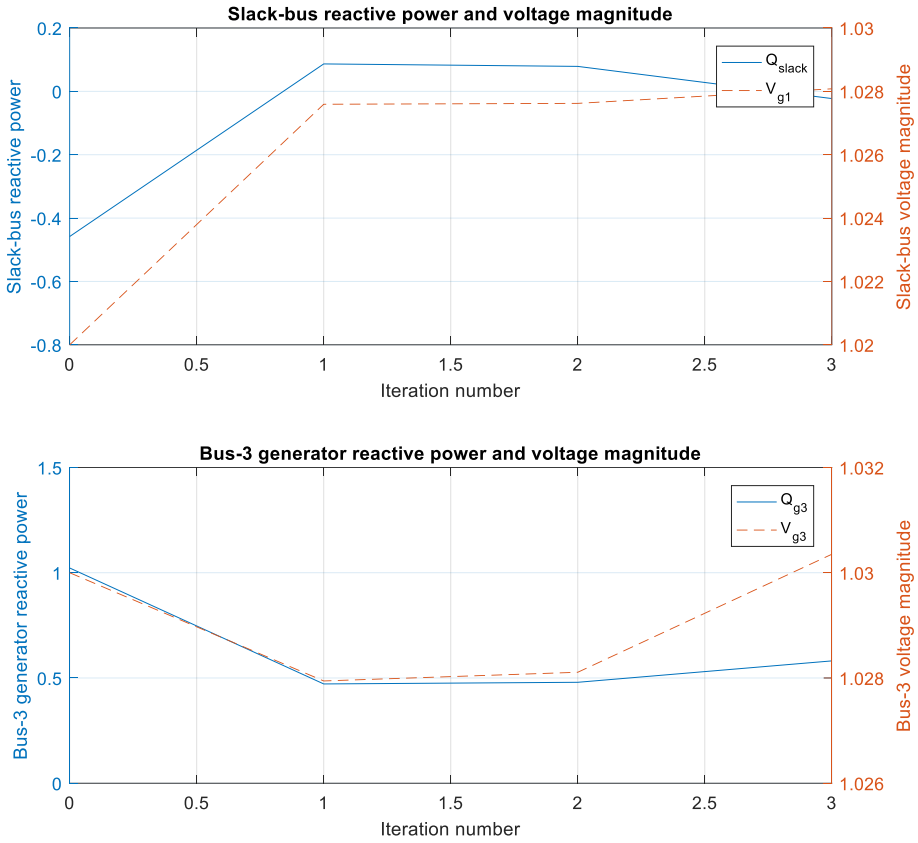

Fig. 5.9: 3-bus system comparison of change in each generator's reactive power output with change in its terminal voltage magnitude

5.3.2 Case study 2: 6-bus power system

Case study 2 is based on a 6-bus power system adapted from Wood *et al.* (2014), which has 3 generators (with the slack generator located at bus 1), 11 lines and 3 loads. The network, load and generation data for the system are presented in Appendix B.

The results of the Volt/VAR optimization for the 6-bus system are presented in Tables (5.6) and (5.7), as well as Figures (5.10) to (5.14). Table (5.6) displays the generator voltage magnitudes (as the control variables for the optimization) prior to and following the execution of the VVO algorithm, and shows an increase in the voltage magnitude in each case. The whole voltage profile of the system (including load-bus voltages) is depicted in Figure (5.10) in the form of a bar chart, which demonstrates that the Volt/VAR optimization leads to an increase in all system voltages, yet both lower and upper bounds on the voltage magnitudes are respected, as can be deduced from Table (5.6). Table (5.7) shows a power loss reduction of *3.372%*, achieved in about *13 iterations*, with an execution time of *178 milliseconds* (about twice the run-time of the algorithm for the 3-bus system case study).

Table 5.6: 6-bus system generator voltage magnitudes before and after Volt/VAR optimization

Control variable	*Initial value*	*Optimal value*	*Lower limit*	*Upper limit*
V_{g1}	1.07	1.099	0.95	1.1
V_{g2}	1.05	1.098	0.95	1.1
V_{g3}	1.05	1.10	0.95	1.1

Table 5.7: 6-bus system loss reduction before and after Volt/VAR optimization

	Initial	*Final*	*Percentage loss reduction*
Real power system losses (p.u.)	0.1335	0.1290	3.372%
Number of iterations	13		
Execution time (sec)	0.1775		

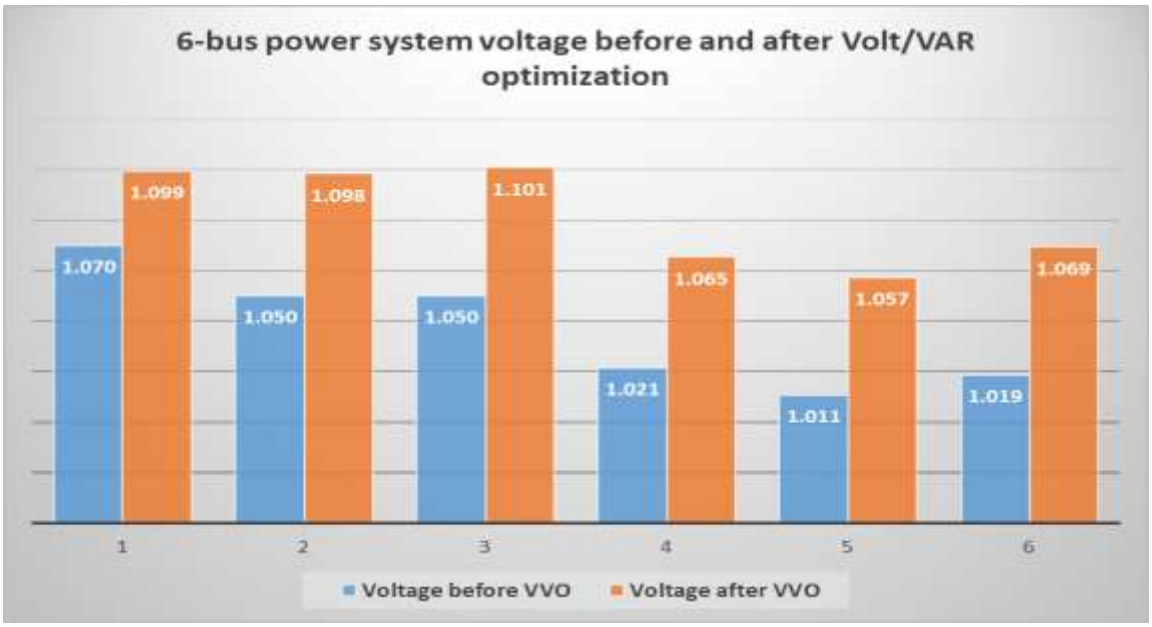

Fig. 5.10: bar chart of 6-bus system generator voltage magnitudes before and after Volt/VAR optimization

The power loss reduction of the 6-bus system over the iterations of the VVO algorithm is depicted pictorially in Figure (5.11), where it can be seen that the power loss initially increases, then decreases progressively as the algorithm continues to iterate. This behaviour of the algorithm can be explained by the fact that it is a local optimizer, and its convergence characteristics are impacted by the initial starting

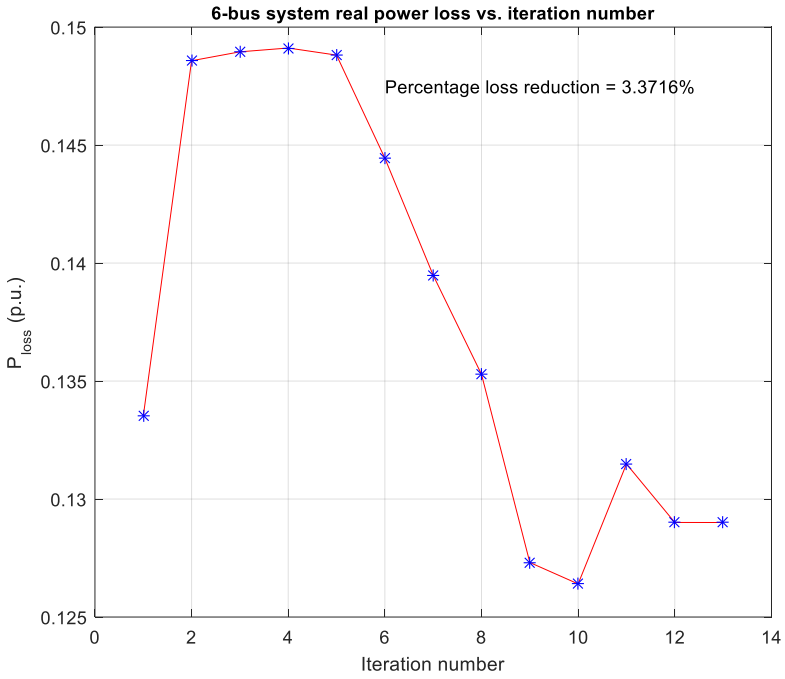

Fig. 5.11: 6-bus system real power losses plotted against the iteration number

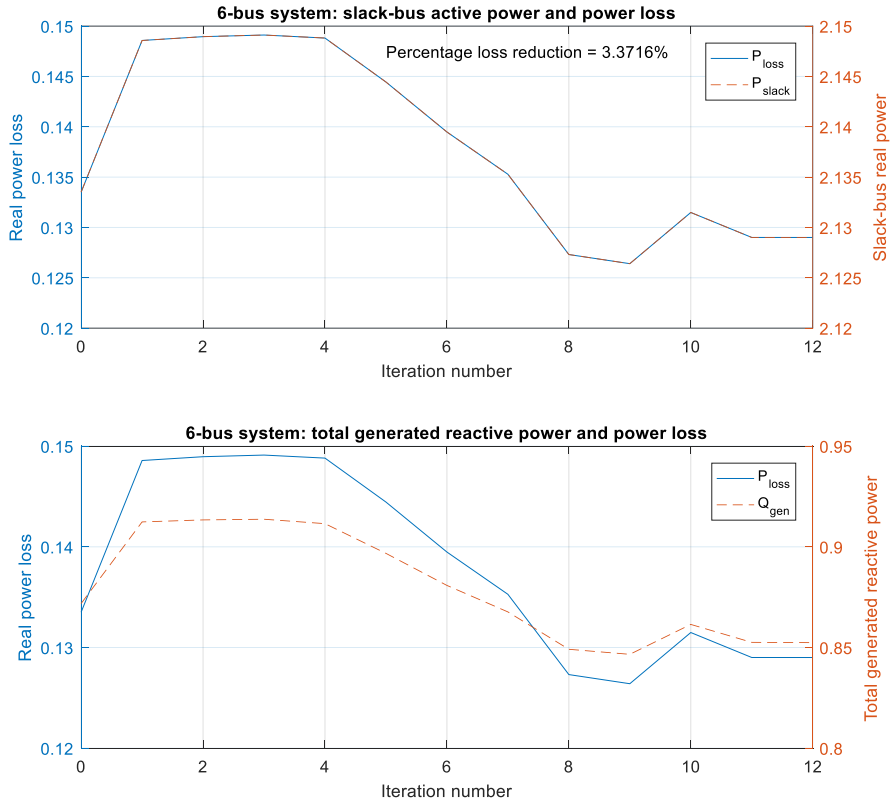

Fig. 5.12: 6-bus system comparison of real power loss with slack-bus active power (top plot) and with total generated reactive power (bottom plot)

point. Despite taking significantly more iterations to converge due to the initial increase in the objective function value, the overall execution time is nonetheless small, which demonstrates the efficiency of the algorithm.

Comparison of the real power loss trajectory with the slack-bus active power and the total generated reactive power in Figure (5.12) shows similar characteristics to those observed in Figure (5.7) for the 3-bus system case. That is, the real power loss curve coincides with that of the slack-bus active power, and the total system reactive power generation tracks the real power loss reduction, indicating simultaneous real and reactive power loss minimization.

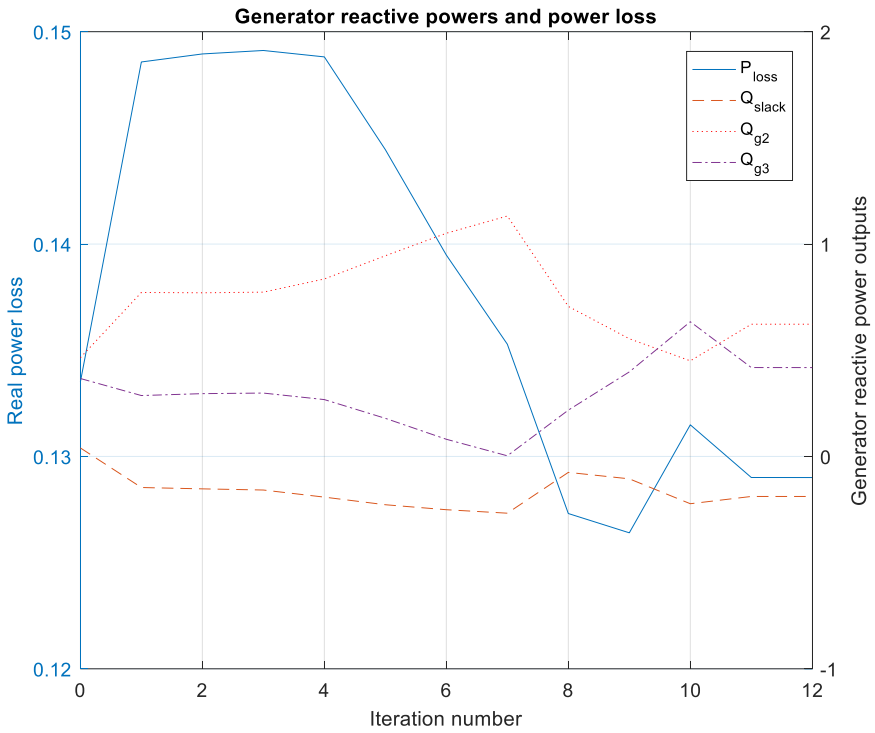

Fig. 5.13: 6-bus system comparison of change in each generator's reactive power output with change in real power loss

The individual generator reactive power outputs are compared with the real power loss curve in Figure (5.13). Although no clear relationship can be seen in this comparison, it is quite clear from the figure that running the Volt/VAR optimization algorithm leads to redistribution of reactive power generation in the system, which moreover results in overall reduction in both active and reactive power generation in the system, as can be deduced from Figure (5.12).

In Figure (5.14), the reactive power output variation of each generator is compared with the change in its terminal voltage magnitude, and a general proportional relationship between the two quantities can be observed, although with a varying correlation coefficient (i.e. relative proportional change) and responsiveness for the various generators.

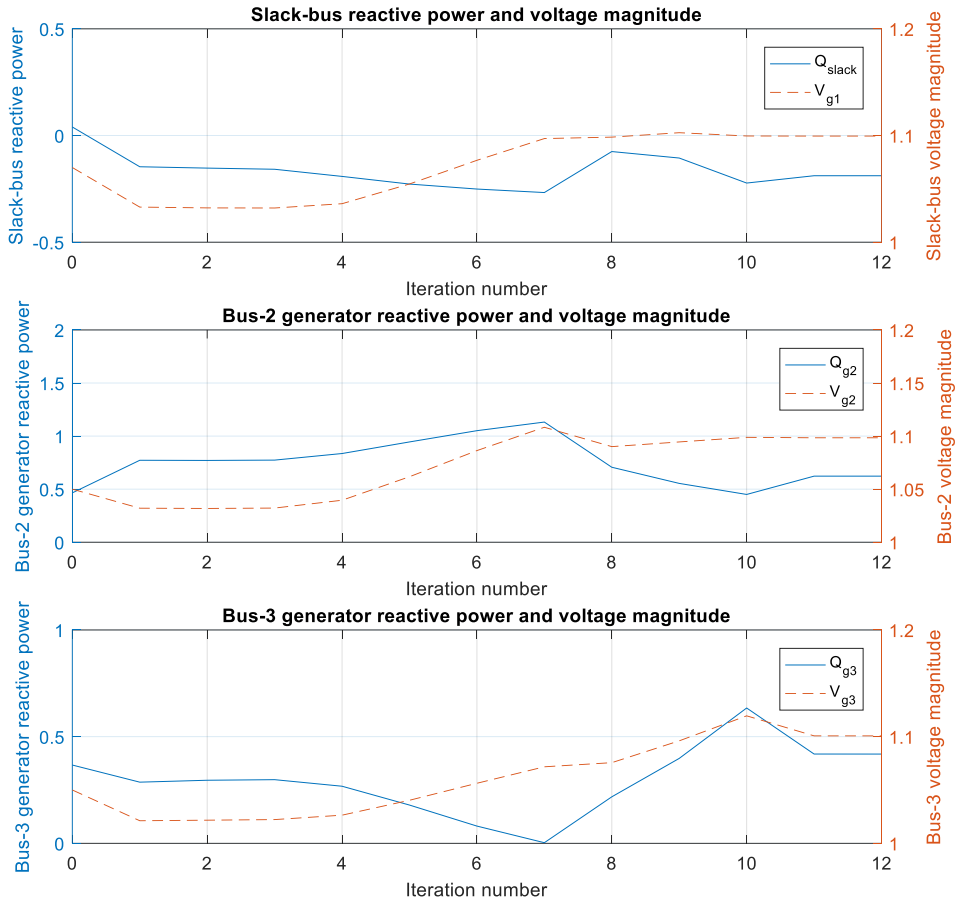

Fig. 5.14: 6-bus system comparison of change in each generator's reactive power output with change in its terminal voltage magnitude

5.3.3 Case study 3: IEEE 14-bus power system

The IEEE 14-bus system considered in this case study comprises 14 buses, 5 generators, 20 lines, and 11 loads. The network, load and generation data for the system have been taken from Zhu (2009), and are presented in Appendix B. The network represents a portion of the transmission system in the Midwest United States, as of 1962. It is characterized as having low base voltages and a lot of voltage control capability (Gonzalez-Longatt, 2015).

The results of the Volt/VAR optimization for this case study are presented in Tables (5.8) and (5.9), as well as Figures (5.15) to (5.19). The optimal set-points for the generator voltage magnitudes are listed in Table (5.8), which presents them along with the corresponding initial values. The final bus voltage magnitudes for the whole

Table 5.8: IEEE 14-bus system generator voltage magnitudes before and after Volt/VAR optimization

Control variable	*Initial value*	*Optimal value*	*Lower limit*	*Upper limit*
V_{g1}	1.06	1.1	0.95	1.1

V_{g2}	1.045	1.077	0.95	1.1
V_{g3}	1.01	1.061	0.95	1.1
V_{g6}	1.07	1.10	0.95	1.1
V_{g8}	1.09	1.036	0.95	1.1

Table 5.9: IEEE 14-bus system loss reduction before and after Volt/VAR optimization

	Initial	***Final***	***Percentage loss reduction***
Real power system losses (p.u.)	0.1353	0.1296	4.237%
Number of iterations	14		
Execution time (sec)	0.1477		

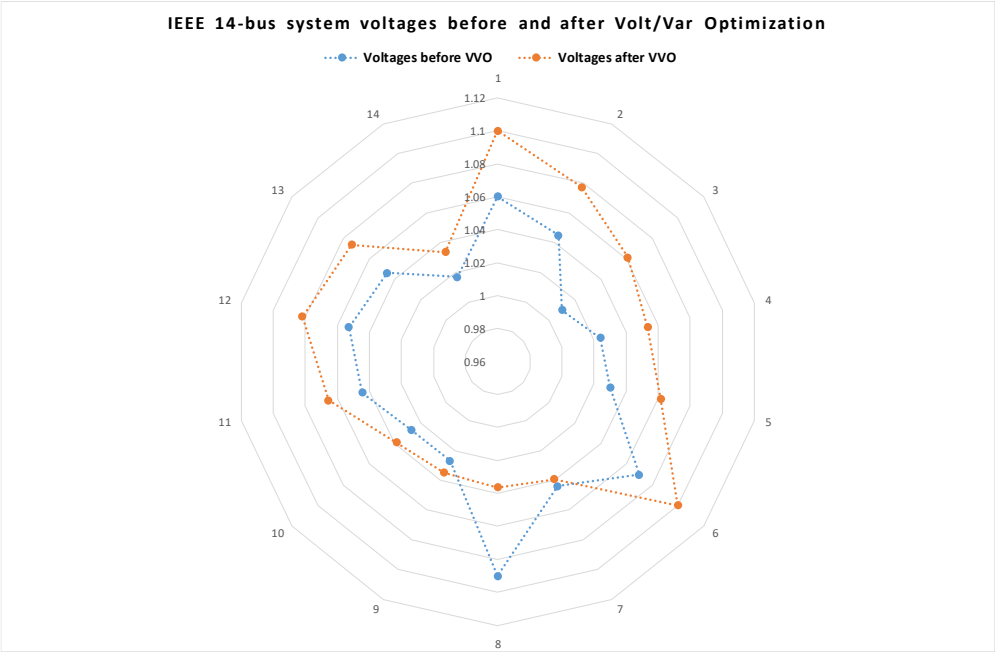

Fig. 5.15: IEEE 14-bus system generator voltage magnitudes prior to and following VVO in radar chart form

system are depicted in the radar chart in Figure (5.15). Table (5.8) reveals that nearly all optimal voltage set-points are greater than the corresponding initial values (except for bus 8, which is also the only final voltage lower than the initial value for the whole system, according to Figure (5.15)). The power loss reduction due to the optimization is *4.237%*, which is achieved within *14 iterations*, with an execution time of about *148 milliseconds*. This information is presented in Table (5.9). The number of iterations taken is only one more than that required for the 6-bus system case, and with a slightly lower runtime, which indicates that the execution time (to convergence) is not

quite proportional to the number of iterations. It may also be pointed out that the convergence rate of the algorithm will depend on other case-specific characteristics (such as the initial starting point).

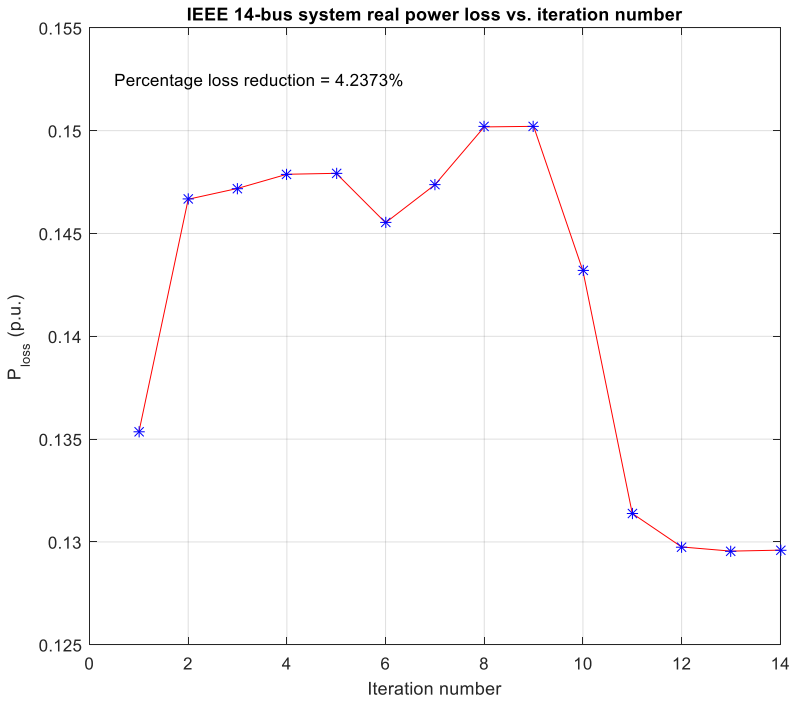

Fig. 5.16: IEEE 14-bus system real power losses plotted against the iteration number

Figure (5.16) depicts the power loss trajectory for the case under study, plotted against the iteration count. Similar to the 6-bus system case, the power loss initially increases, then decreases progressively in the subsequent iterations of the algorithm. Figure (5.17) shows characteristics similar those observed in the previous two cases, that is, the coincidence of the power loss and slack-bus active power trajectories, as well as the corresponding decrease in both the real power loss and the total system reactive power generation.

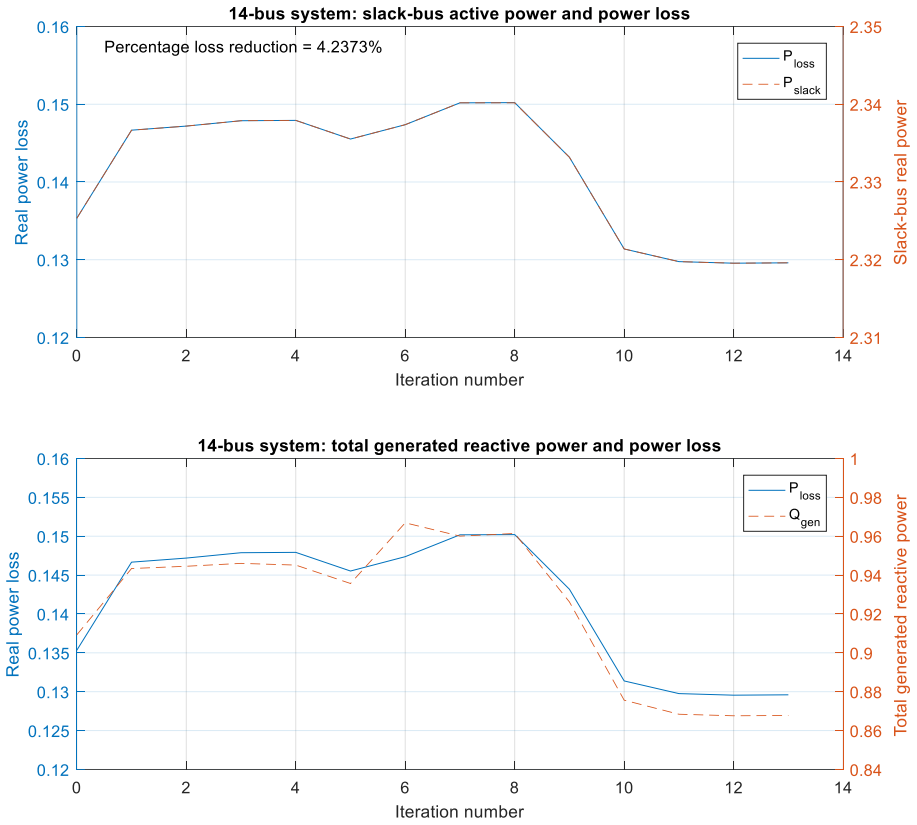

Fig. 5.17: IEEE 14-bus system comparison of real power loss with slack-bus active power (top plot) and with total generated reactive power (bottom plot)

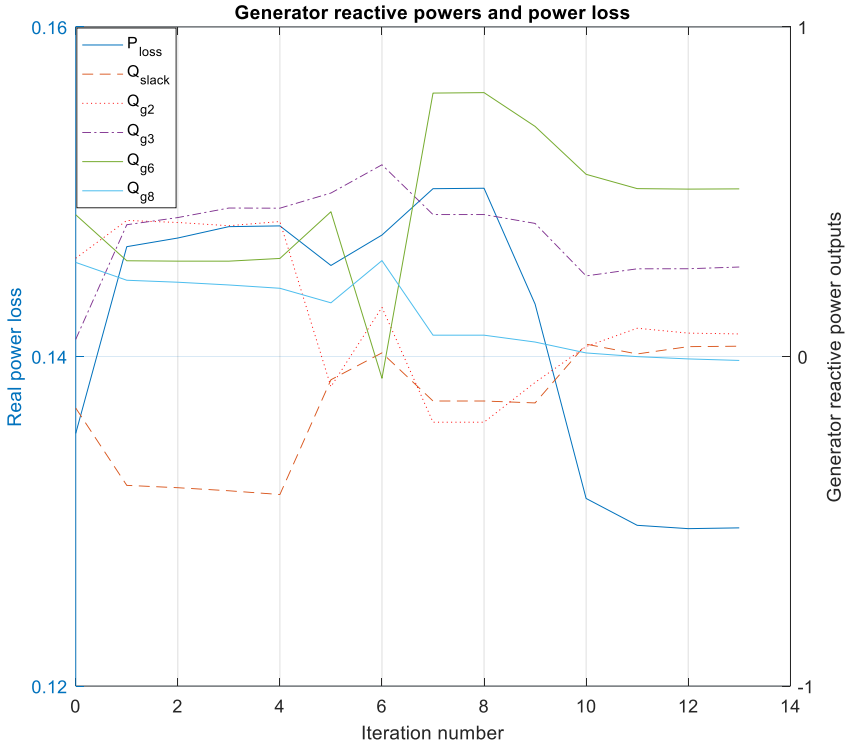

Fig. 5.18: IEEE 14-bus system comparison of change in each generator's reactive power output with change in real power loss

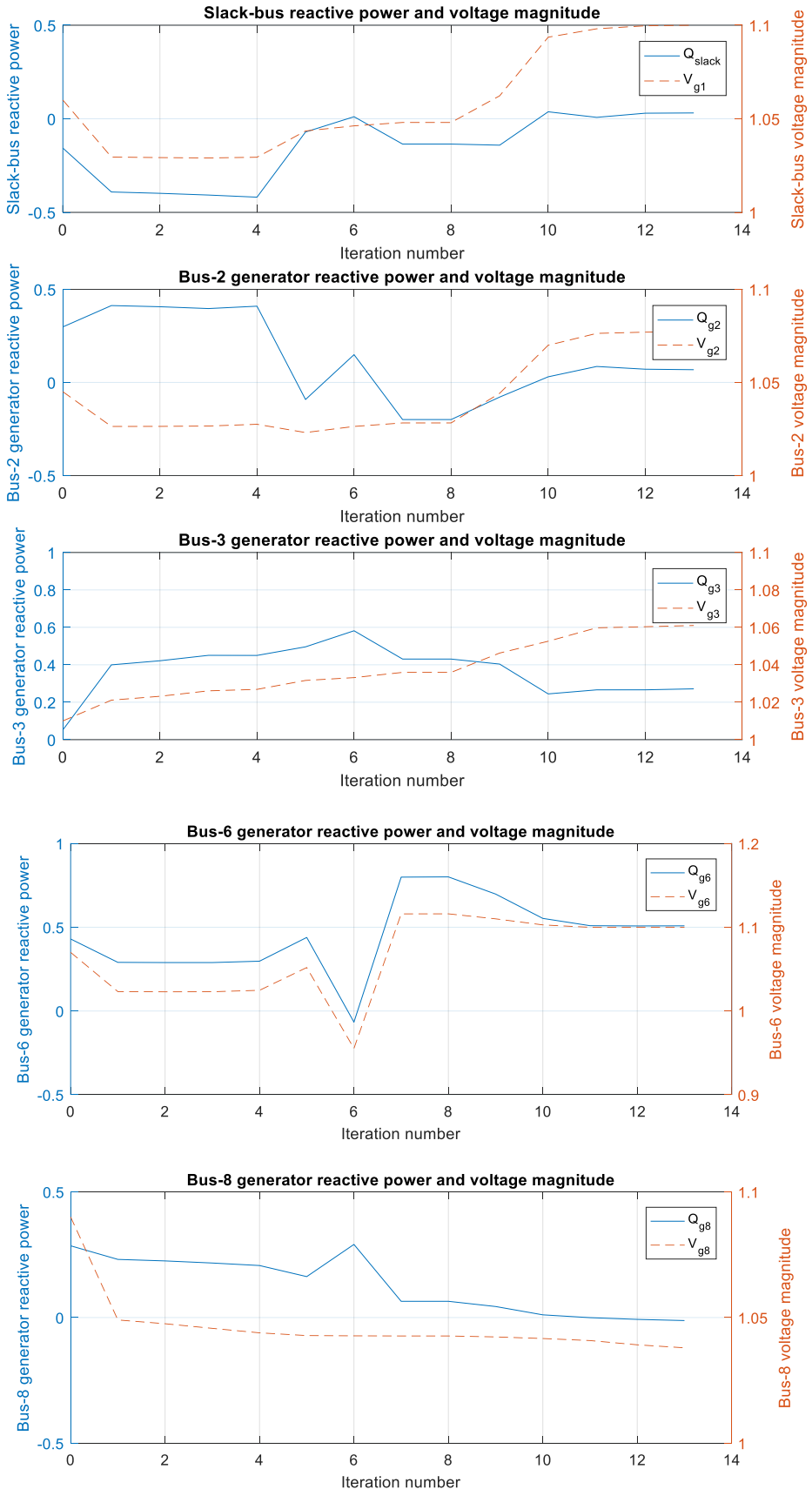

Fig. 5.19: IEEE 14-bus system comparison of change in each generator's reactive power output with change in its terminal voltage magnitude

Comparison of the individual generator reactive power outputs with the real power loss curve, as well as with the corresponding generator terminal voltage magnitudes,

is presented in Figures (5.18) and (5.19) respectively. Figure (5.19) shows that the majority of generator reactive power outputs change in tandem with the real power loss, the main difference being with the slack-bus reactive power output, whose change tends to be opposite to that of the aggregated (non-reference) generator outputs. The reactive power change mostly tracks the voltage set-point changes for all generators, as can be seen in Figure (5.19).

5.3.4 Case study 4: IEEE 30-bus power system

This case study is based on the IEEE 30-bus system, which comprises 30 buses, 6 generators, 41 lines, and 21 loads, based on the network, load and generation data taken from Zhu (2009), which are presented in Appendix B. Similar to the IEEE 14-bus system considered in the previous case study, the IEEE 30-bus system also represents a portion of the transmission system in the Midwest United States, as of December 1961 (Gonzalez-Longatt, 2014).

Table 5.10: IEEE 30-bus system generator voltage magnitudes before and after Volt/VAR optimization

Control variable	*Initial value*	*Optimal value*	*Lower limit*	*Upper limit*
V_{g1}	1.0	1.026	0.95	1.1
V_{g2}	1.0	1.027	0.95	1.1
V_{g5}	1.0	1.028	0.95	1.1
V_{g8}	1.0	1.029	0.95	1.1
V_{g11}	1.0	1.025	0.95	1.1
V_{g13}	1.0	1.025	0.95	1.1

Table 5.11: IEEE 30-bus system loss reduction before and after Volt/VAR optimization

	Initial	*Final*	*Percentage loss reduction*
Real power system losses (p.u.)	0.1141	0.1084	5.0298%
Number of iterations	14		
Execution time (sec)	0.3565		

The Volt/VAR optimization results for the IEEE 30-bus system are presented in Tables (5.10) and (5.11), as well as Figures (5.20) to (5.24). As detailed in table (5.10), all the generator initial voltage set-points are given a flat start (i.e. starting at

unity). Volt/VAR optimization leads to an average increase of about 2.5% for each generator terminal voltage magnitude.

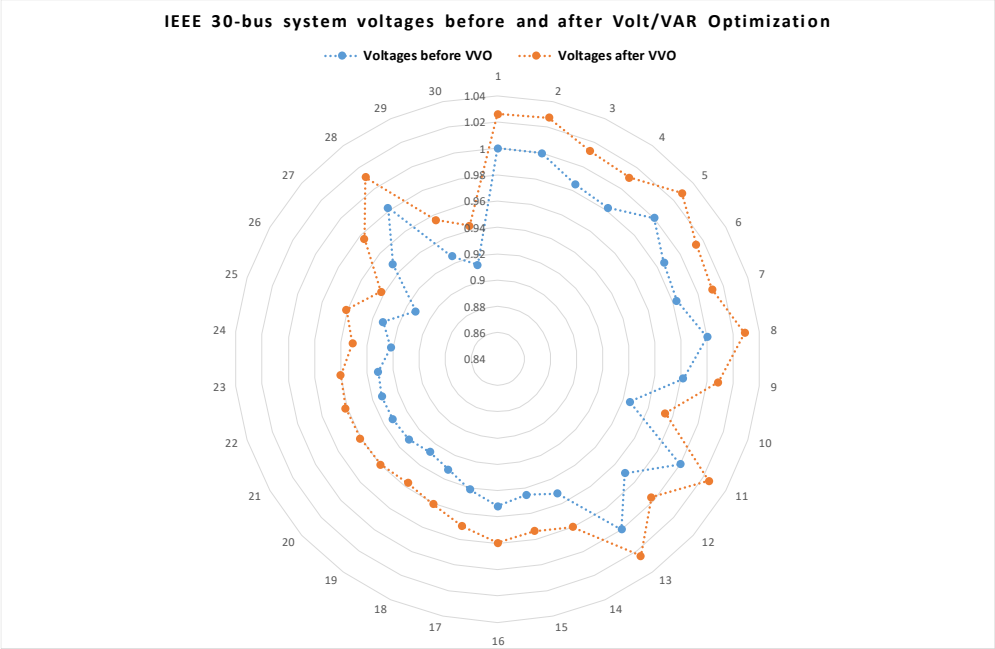

Fig. 5.20: IEEE 30-bus system generator voltage magnitudes before and after Volt/VAR optimization in radar char form

The voltage profile (before and after optimization) for the entire network is depicted in the radar chart in Figure (5.20), where an overall voltage profile improvement for the entire network can be deduced, particularly in terms of eliminating low-voltage violations for quite a few load buses.

The power loss reduction following the Volt/VAR optimization for this case is *5.03%*, which is achieved within *14 iterations*, with an execution time of about *357 milliseconds*, as can be read from Table (5.11). The number of iterations taken by the algorithm to converge for this case is the same as that required for the previous case study (i.e. the IEEE 14-bus system), but the execution time is longer (about 2.4 times longer), again showing that the relationship between the number of iterations and the execution time is not quite fixed, and is influenced by a number of case-specific characteristics, as mentioned earlier.

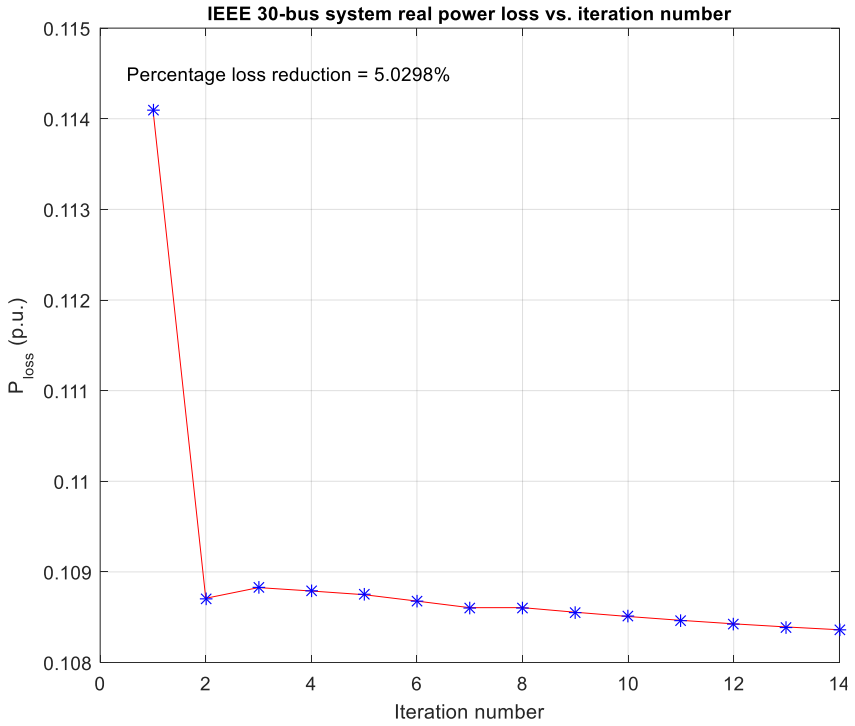

Fig. 5.21: IEEE 30-bus system real power losses plotted against the iteration number

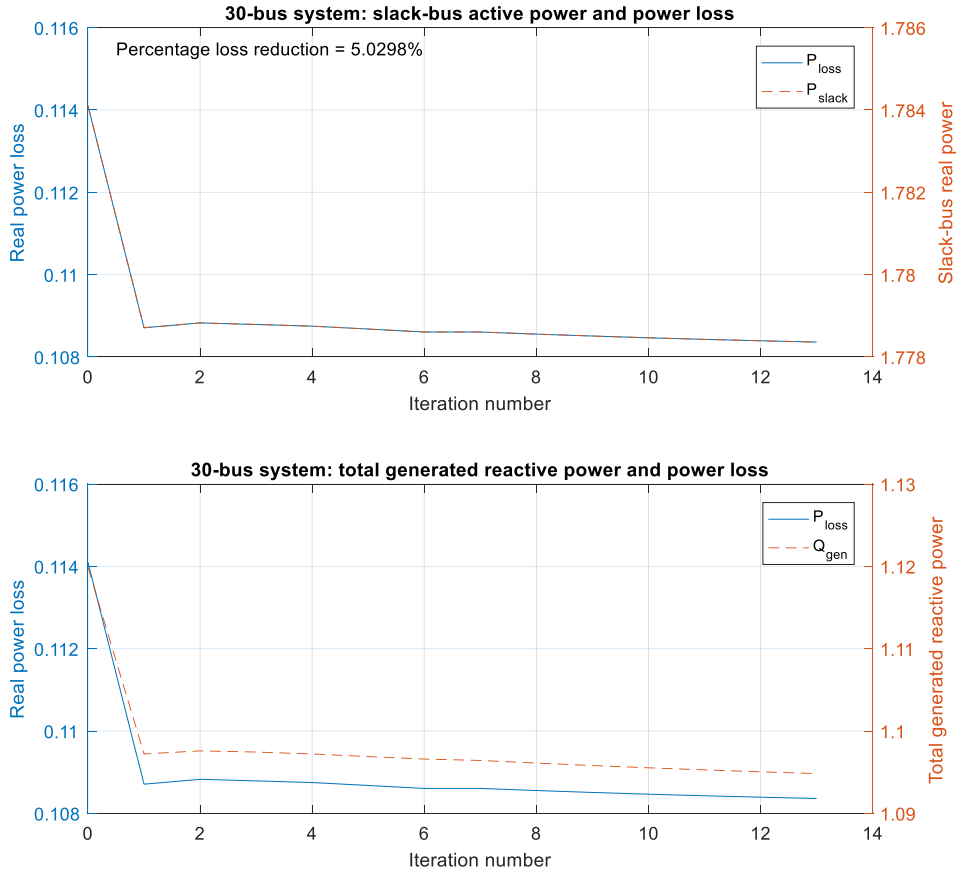

Fig. 5.22: IEEE 30-bus system comparison of real power loss with slack-bus active power (top plot) and with total generated reactive power (bottom plot)

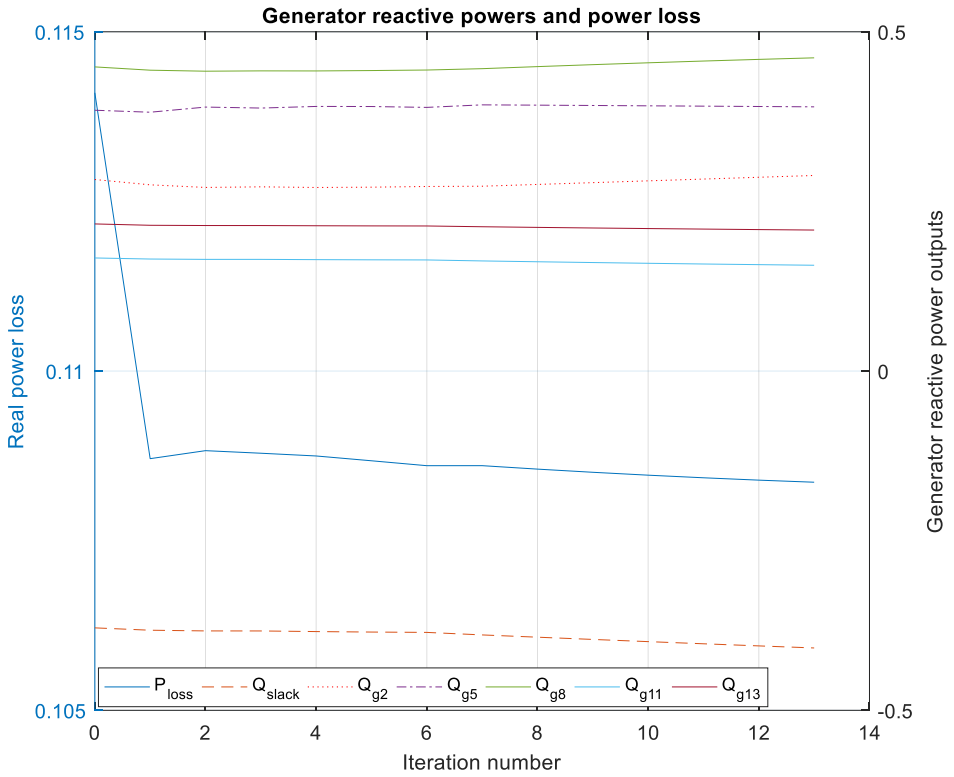

Fig. 5.23: IEEE 30-bus system comparison of change in each generator's reactive power output with change in real power loss

Plotting the real power loss trajectory together with the slack-bus active power (Figure 5.22, top plot) and the total system reactive power generation (Figure 5.22, bottom plot) reveals characteristics similar to those observed in the previous case studies. The individual generator reactive power outputs are together compared with the real power loss in Figure (5.23), and separately with the corresponding generator terminal voltage magnitudes in Figure (5.24). The comparison in Figure (5.23) does not seem to yield much information, but looking at the bottom plot in Figure (5.22) reveals a clear correlation between the real power loss and the aggregated reactive power generation trajectories. The trajectories of the individual generator reactive power outputs compared with the generator terminal voltage magnitudes in Figure (5.24) do not show a consistent relationship for the observed window. In some cases direct proportionality can be observed (albeit with a lagging effect on the response of the reactive power to the generator voltage set-point change), in other cases inverse proportionality seems to be exhibited. Overall, it can be seen that adjustment of the generator terminal voltage magnitude set-point always leads to a corresponding change in the generator reactive power output, and there is generally a positive correlation in the variation of the two quantities.

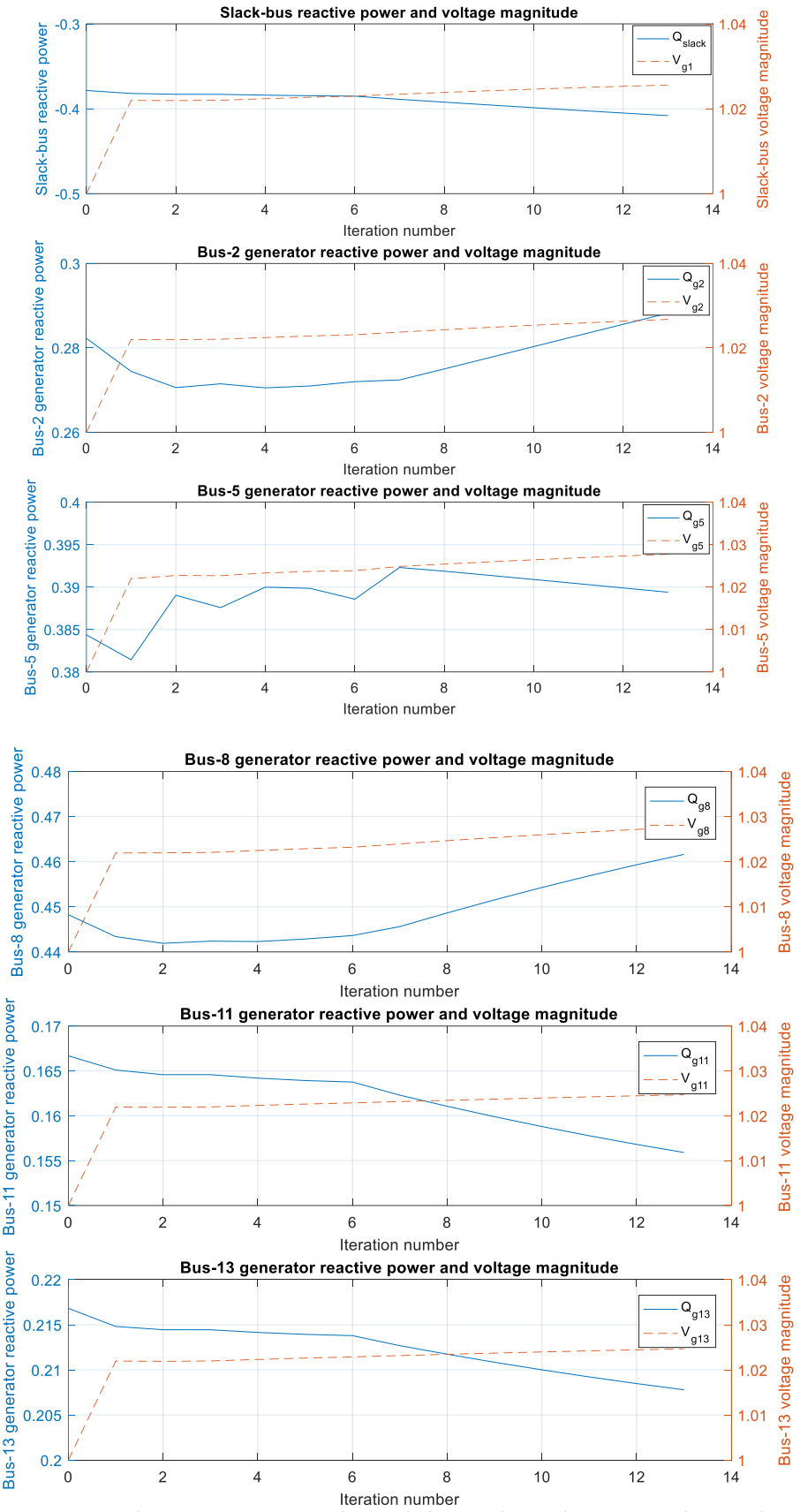

Fig. 5.24: IEEE 30-bus system comparison of change in each generator's reactive power output with change in its terminal voltage magnitude

5.3.5 Case study 5: IEEE 118-bus power system

The final case study considered for the analysis of the developed PDIPM-based VVO algorithm is the IEEE 118-bus system, which comprises 118 buses, 19 generators, 35

synchronous condensers (i.e. synchronous generators that are confined to generating reactive power only), 186 lines, and 99 loads. The network, load and generation data is taken from an appendix attributed to Springer Verlag (2012) and is listed in Appendix B.

The Volt/VAR optimization results for the IEEE 118-bus system are presented in Table (5.12) and Figures (5.25) to (5.29). Due to the very large number of generators (54 in total), the initial and optimal generator voltage set-points for this case study are presented in the form of a radar chart, depicted in Figure (5.25), unlike the previous cases in which this information was tabulated.

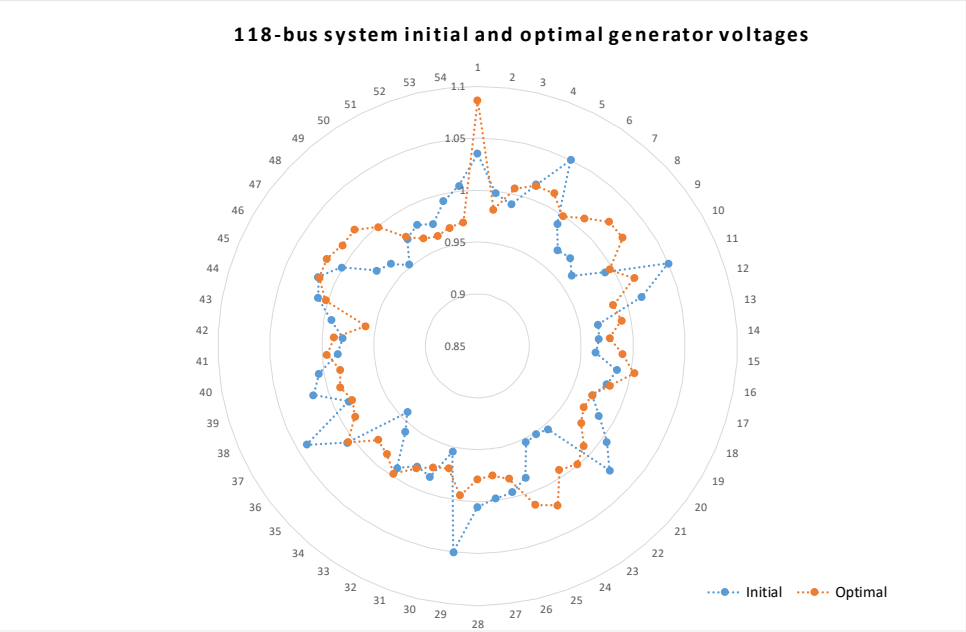

Fig. 5.25: IEEE 118-bus system generator voltage magnitudes before and after Volt/VAR optimization

Table 5.12: IEEE 118-bus system loss reduction before and after Volt/VAR optimization

	Initial	***Final***	***Percentage loss reduction***
Real power system losses (p.u.)	3.3939	3.2270	4.9167%
Number of iterations	8		
Execution time (sec)	2.012		

The main improvement that can be observed from Figure (5.25) is the relief in the low-voltage violations at a considerable number of buses. The voltage profiles before and after the optimization for the entire network are depicted in the radar chart in

Figure (5.26), which exhibits similar characteristics (of raising the voltage magnitudes from the lower limit) across the network buses.

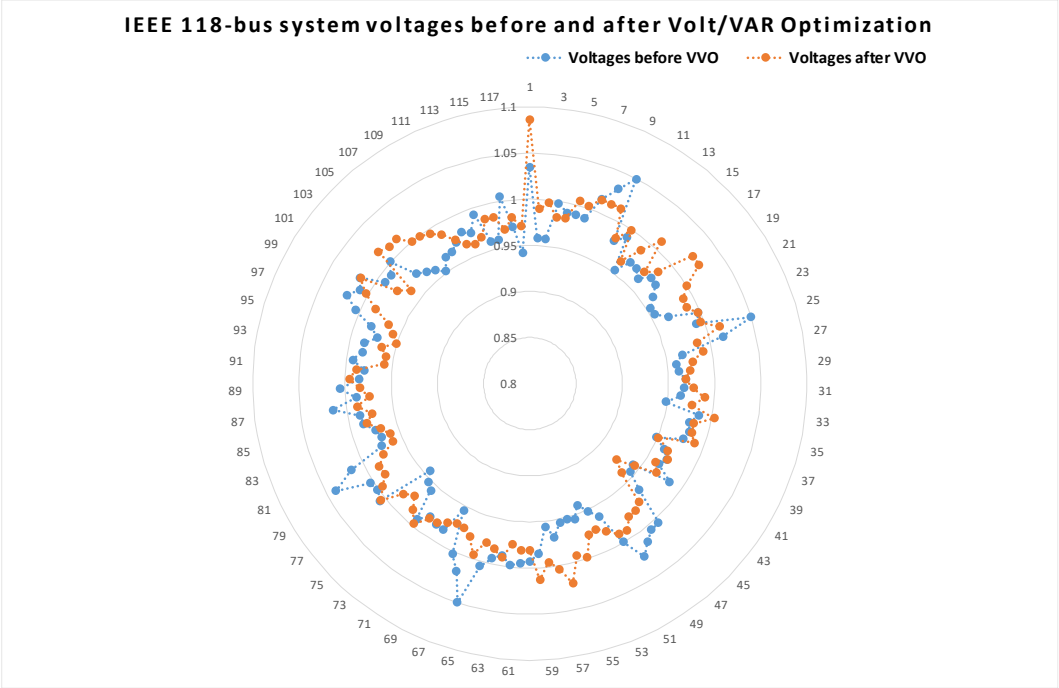

Fig. 5.26: IEEE 118-bus system generator voltage magnitudes before and after Volt/VAR optimization in radar char form

The power loss reduction following the Volt/VAR optimization for the IEEE 118-bus system is *4.917%*, which is achieved within *8 iterations*, with an execution time of about *2.012 seconds*, information which is tabulated in Table (5.12). For this case, the algorithm takes fewer iterations to converge compared with the IEEE 30-bus and the IEEE 14-bus systems (8 vs. 14), but the execution time is longer, implying that each iteration takes longer to execute, due to the much larger size of the system. Overall, the algorithm appears to scale very well with the system size. For example, comparing the IEEE 118-bus and IEEE 30-bus system cases, the former is about 4 times as large (based on the number of buses), and the execution time is only about 5 times longer, which is nearly a linear scaling of the execution time with the problem size. This applies particularly to the studied cases, and need not be taken as a guarantee that the algorithm will always exhibit these desirable performance characteristics.

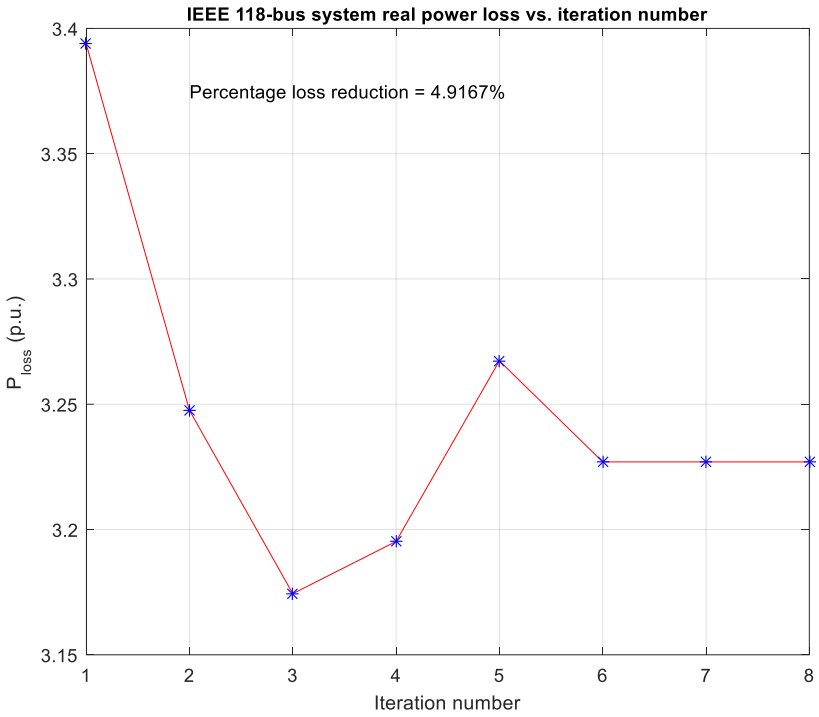

Fig. 5.27: IEEE 118-bus system real power losses plotted against the iteration number

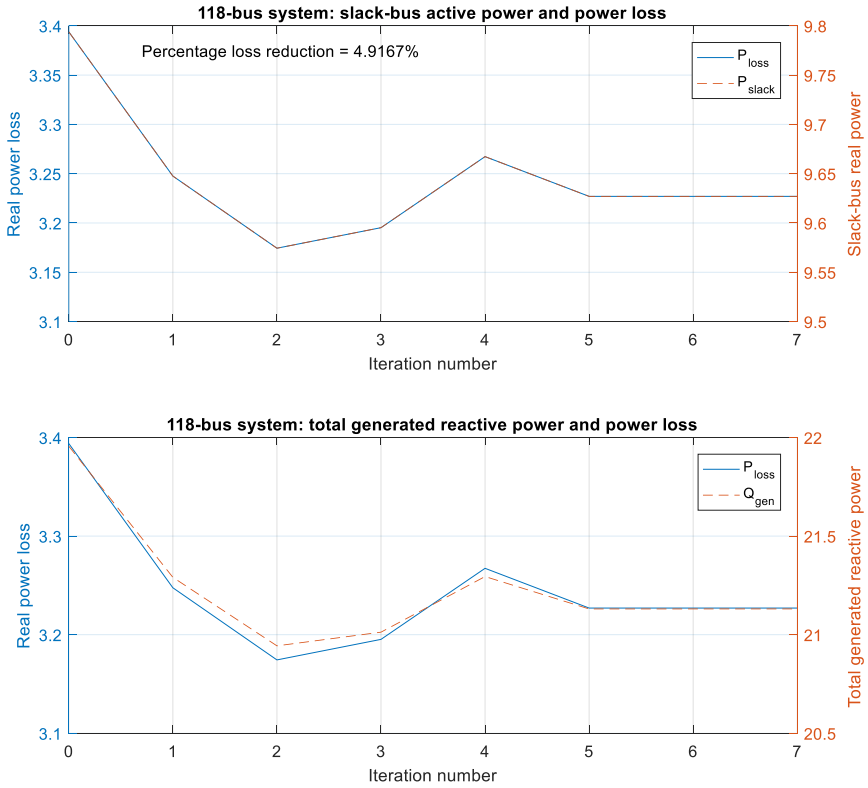

Fig. 5.28: IEEE 118-bus system comparison of real power loss with slack-bus active power (top plot) and with total generated reactive power (bottom plot)

The real power loss trajectory is plotted against the iteration count separately in Figure (5.27), and together with the slack-bus active power and the total system reactive power generation in the top and bottom plots of Figure (5.28) respectively. The top plot in Figure (5.29) compares the real power loss with the slack-bus reactive

power, whereas the bottom plot compares the slack-bus reactive power output with the slack-bus generator terminal voltage magnitude. The comparisons exhibit similar characteristics to those observed in the previous 4 case studies, thus demonstratting the consistency of the results produced by the algorithm when applied to different cases, irrespective of the system size.

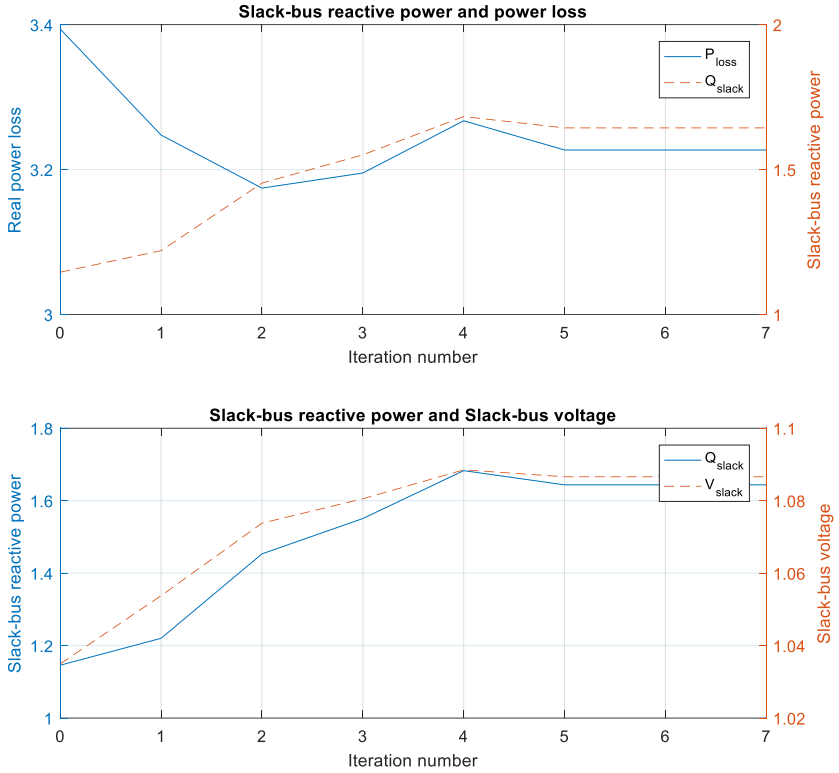

Fig. 5.29: IEEE 118-bus system comparison of slack-bus reactive power with real power loss (top plot) and slack-bus reactive power and voltage magnitude change (bottom plot)

Finally, the efficacy of the presented algorithm is demonstrated by comparing with results reported in the literature. The results are tabulated in Tables 5.13 and 5.14 for the IEEE 14-bus and IEEE-30-bus systems respectively. It can be deduced from the comparative analysis that the primal-dual interior-point algorithm presented in this chapter has superior performance, both in terms of solution quality (i.e. magnitude of real power loss minimization) and efficiency (i.e. required execution time).

Table 5.13: Performance comparison of the PDIPM-VVO algorithm presented in this chapter with other algorithms from the literature for the IEEE 14-bus system

	Interior-Point Method (Zhu, 2009)	**Linear Programming (Zhu, 2009)**	**PDIPM (presented in this chapter)**
Initial loss (p.u.)	0.11646	0.11646	0.1125
Final loss (p.u.)	0.11004	0.11108	0.1084
% Real power loss reduction	5.513	4.619	6.914
Number of iterations	-	-	16
Execution time (sec)	18.2	61.5	0.0578

Table 5.14: Performance comparison of the PDIPM-VVO algorithm presented in this chapter with other algorithms from the literature for the IEEE 30-bus system

	Modified Particle Swarm Optimization (Zhu, 2009)	Interior-Point Method (Zhu, 2009)	PDIPM (presented in this chapter)
Initial loss (p.u.)	-	-	0.0505
Final loss (p.u.)	0.050921	0.051109	0.0480
% Real power loss reduction	-	-	5.096
Number of iterations	-	-	20
Execution time (sec)	-	-	0.2064

5.4 Conclusion

The main result of this chapter is the design and implementation of a Volt/VAR optimization algorithm based on the Primal-Dual Interior-Point Method presented in chapter 4. All the pertinent aspects related to the implementation of the designed PDIPM-based Volt/VAR optimization (PDIPM-VVO) algorithm have been discussed in great detail. A distinctive feature of the PDIPM-VVO algorithm's implementation (particularly when compared with the more generic algorithm presented in chapter 4) is that it incorporates a load flow computation (by the Newton-Raphson method), which then requires the separate handling of the equality and inequality constraints of the VVO problem. After demonstrating the implementation of the PDIPM-VVO algorithm for a simple power system, 5 case studies are conducted, the aim being to analyse the performance of the developed algorithm in terms of the real power loss reduction and voltage profile improvement, as well as the efficiency and convergence characteristics of the algorithm, besides other qualitative aspects of the algorithm performance. The extensive analyses that have been conducted reveal the algorithm's effectiveness and efficiency, notably in being able to successfully solve the VVO problem for systems of widely varying size without much increase in computational cost or deterioration in the quality of the results. Based on the case studies conducted in this chapter, the developed PDIPM-VVO algorithm exhibits characteristics of fast convergence, high efficiency, and scalability to large-scale problems. The results obtained are very encouraging, and suggest carrying out more analyses with the goal of possibly further optimizing it so as to be able to effectively handle a wide variety of operational scenarios.

The following chapter will carry on with the developmental work, this time addressing the design and implementation of a Volt/VAR optimization algorithm based on the Particle Swarm Optimization, a prominent heuristic optimization method.

CHAPTER SIX

SOLUTION OF THE VOLT/VAR OPTIMIZATION PROBLEM BY THE PARTICLE SWARM OPTIMIZATION ALGORITHM

6.1 Introduction

Heuristic/intelligent search-based optimization techniques employ a variety of optimum-seeking strategies that are distinctly different from the approaches taken in conventional optimization algorithms, such as the primal-dual interior-point algorithm considered in the preceding chapter. The search strategies employed in these techniques are meant to overcome many of the deficiencies of the conventional optimization problems, such as the local (rather than global) nature of the search, the limited ability to handle combinatorial problems with discrete decision variables, and the requirement for the smoothness of the objective and constraint functions for gradient-based methods, among other factors (Frank et al., 2012b). Over the past few decades a wide variety of these heuristic optimum-seeking techniques have been developed, among the prominent ones being Genetic Algorithms (GA), Evolutionary Programming (EP), Particle Swarm Optimization (PSO), Fuzzy Set Theory, and Expert Systems (ES). In this chapter, the PSO algorithm is discussed, and its application to the solution of the Volt/VAR optimization algorithm is presented.

The outline of this chapter is as follows. After a brief discussion of the historical development of the PSO algorithm in section 6.2, the principle of operation and basic formulation of the algorithm is presented in section 6.3. This is followed by a detailed discussion of the key practical considerations and implementation aspects of the algorithm in section 6.4. The application of the PSO algorithm to the solution of the VVO problem is then presented in section 6.5. The section begins with a detailed outline of the implementation of the PSO-based VVO algorithm, after which five case studies and simulation results are presented and discussed, based on the 3-bus, 6-bus, 14-bus, 30-bus and 118-bus test systems. Section 6.6 details a comparative analysis of the PDIPM algorithm (developed and presented in the preceding chapter) and the PSO algorithm presented in this chapter, and interesting results of the comparative analysis are highlighted. Section 6.7 concludes the chapter with key results from the chapter. A pictorial representation of the content of this chapter is depicted in Figure (6.1).

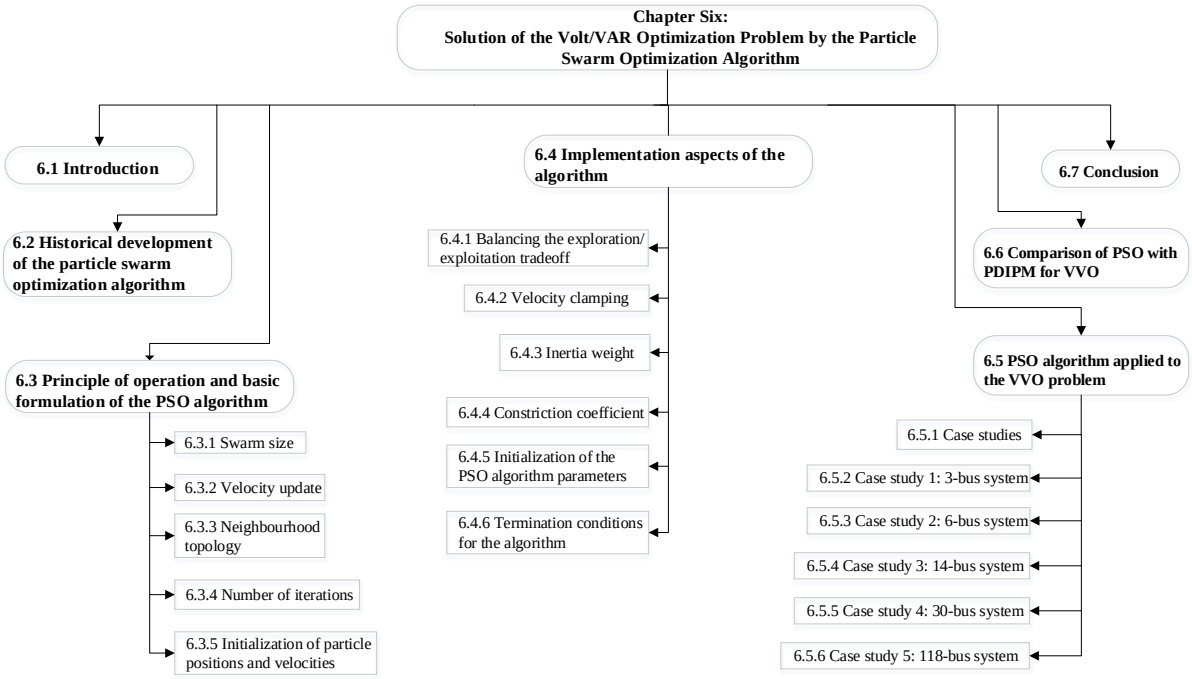

Fig. 6.1: Summary of the content covered in this chapter

The key contributions of this research as presented in this chapter are:

- Thorough presentation of both the historical development of the particle swarm optimization algorithm as well as the details of the most important implementation aspects that have a major influence on the algorithm's effectiveness and efficiency.
- Development and implementation of the PSO-based Volt/VAR optimization algorithm, formulated in rectangular coordinates, which incorporates the rectangular-coordinate Newton-Raphson load flow computation.
- Comprehensive performance analysis of the developed PSO-VVO algorithm, focusing on the quality of the solution (in terms of the magnitude of real power loss percentage reduction and the voltage profile improvement) and the computational efficiency of the algorithm (in terms of the required number of iterations and runtime).
- Study and analysis of the impact of the swarm size on the solution quality and the computational cost of the PSO algorithm.

6.2 Historical development of the particle swarm optimization algorithm

Particle swarm optimization (PSO) belongs to the class of heuristic optimization techniques collectively referred to as *Swarm Intelligence*, which constitutes a stream of Artificial Intelligence (AI) research that got established in the early 1990s, based on the study of the swarm behaviour of natural creatures, in terms of how the decision making of the individual is influenced by both the individual's own experience and the experiences of community members. PSO was conceptualized and developed by J.

Kennedy, a social psychologist, and R. Eberhart, an Electrical Engineer (1995). The main idea behind their conceptualization was to produce computational intelligence by exploiting simple analogues of social interaction among conspeciates, and was inspired by the works of Reynolds (1987), Heppner and Grenander (1990). Reynolds, as well as Heppner and Grenander, had studied the dynamics of bird social behaviour, out of which came the conjecture that the aesthetics and synchrony of flocking behaviour exhibited by birds was a function of the birds' efforts to maintain an optimal inter-individual distance among neighbouring members of the flock. With reference to fish schooling, E.O. Wilson, a socio-biologist, said (1975): "In theory at least, individual members of the school can profit from the discoveries of all other members of the school during the search for food. This advantage can become decisive, outweighing the disadvantage of competition for food items, whenever the resource is unpredictably distributed in patches." What could be inferred from this statement is that social sharing of information among community members offers an evolutionary advantage, and is the principal concept underlying the development of the PSO algorithm (Kennedy & Eberhart, 1995).

Work on the PSO algorithm started out as a simulation of a simplified social milieu, in which agents were conceptualized as portraying collision-proof birds, and were intended to simulate the graceful though unpredictable choreography of a bird flock. The resulting algorithm could be seen to have ties with artificial life (A-life) in general, and with bird flocking, fish schooling, and swarming theory in particular. Additional ties with evolutionary computation, such as genetic algorithms and evolutionary programming, are quite evident. Obvious relations with the evolutionary computation algorithms include the fact that it is population-based, it is highly dependent on stochastic processes for the evolution of the population, and it uses the concept of fitness to differentiate the quality of the candidate solutions (Kennedy & Eberhart, 1995).

A distinctive feature of PSO is the idea of flying candidate solutions through hyperspace in search of better solutions. The algorithm is characterized by simplicity and robustness. Its implementation requires only a few lines of code, making use of only primitive mathematical operators, with modest memory requirements, and only few parameters that need to be specified for any given problem. Out of this "natural simplicity" that is based on emulating nature emerges a powerful algorithm that has proved to be effective for a wide range of applications, notably the training of artificial neural network weights (Hu *et al.*, 2004).

Since its conceptualization in the early 1990s and eventual implementation in the subsequent years, the PSO has undergone a number of developments, among them being:

- The introduction of new parameters (e.g. inertia weight, constriction factor) to improve the algorithm's convergence characteristics.
- Modification of the basic algorithm to tailor it to different problem types (e.g. cooperative PSO).
- Hybridization with other heuristic optimization techniques, to enhance the effectiveness and efficiency of the algorithm.

Figure (6.2) provides a summary of these notable developments (Freitas *et al.*, 2020). Some of them are further discussed in subsequent sections in this chapter.

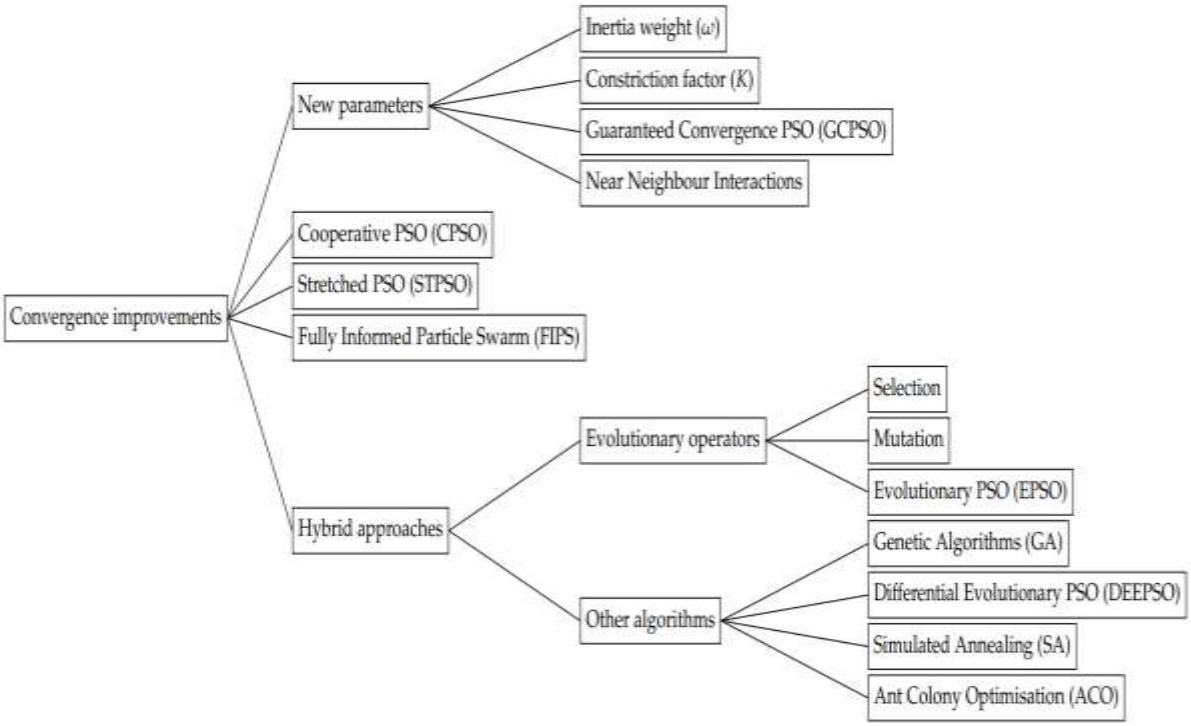

Fig. 6.2: Summary of the key developments of the particle swarm optimization algorithm over the years (adapted from Freitas *et al.*, 2020)

6.3 Principle of operation and basic formulation of the PSO algorithm

PSO can be characterized as a stochastic multi-agent parallel search algorithm in which each of a swarm of particles represents a candidate solution to an optimization problem. A particle can be thought of as an independent intelligent agent that "flies" through a multi-dimensional problem space in search of the optimal solution to the optimization problem, based on its own past flying experience, and that of the rest of the swarm (Hu, *et al.*, 2004). Each particle i in the swarm is comprised of three n-dimensional vectors (n being the dimensionality of the search space, R^n), which at

time ***k*** can be denoted as the current position, X_i^k, the previous best position, $p_{best,i}^k$, and the velocity, V_i^k (Poli *et al*., 2007).

From the perspective of the search space, the current position of each particle, X_i^k, constitutes a set of coordinates representing a point in space, and the movement of this point through the search space is what constitutes the search for the optimal solution to the optimization problem. From the perspective of the optimization problem, X_i^k constitutes the decision vector, which at each iteration is evaluated for "fitness" by means of the objective function of the optimization problem. The particle velocity, V_i^k, embodies the composite flying experience of the individual particle and of the rest of the swarm, and is used to update the individual particle position in an effort to advance it to a "better" position, as judged by its attaining an improved fitness evaluation. It can thus also be characterized as the step size of the algorithm. An iteration of the algorithm is complete once all particle positions have been updated, and their fitness values computed. Besides the current position and the velocity, each particle keeps track of the position corresponding to the best fitness value it has attained up to the latest iteration, denoted as $p_{best,i}^k$, which is then updated to the current position whenever the (updated) current position results in a better fitness value than the previous best value. As the iterations progress, the swarm as a whole, much like a flock of birds foraging for food, is likely to move towards the optimal point in the search space.

It is worth emphasizing, as mentioned earlier, that it is the social interaction and the sharing of information among the swarm's particles that underpins the effectiveness of the PSO algorithm. A particle by itself generally has no ability to solve an optimization problem. Problem solving is thus a collective task in the context of the particle swarm, and progress is possible only through particle interaction of some sort (Kennedy & Eberhart, 1995). This also makes the communication structure or topology, which describes the interconnectivity among the particles and can be thought of as a social network, a key characteristic of the particle swarm. A number of neighbourhood topologies exist, as further discussed later.

The core component of the particle swarm optimization algorithm is the iterative velocity update, which adjusts each particle's position so as to drive the entire swarm towards the optimal solution to the optimization problem. The original algorithm can roughly be outlined as follows (Poli *et al.*, 2007):

1. Initialize a population of particles with random positions and velocities, each of dimension ***n*** in the search space.

2. Iterate:
    2.1 For each particle, evaluate its fitness using the problem's objective function.
    2.2 Compare the particle's fitness value with the previous (personal) best value, and update the previous best position to the current position, if the current fitness value is better than the previous one.
    2.3 Determine the best fitness value among the personal best values of all the particles, and designate the corresponding position as the global best attained by any particle thus far.
    2.4 Compute the velocity update and adjust each particle's position according to the equations (Poli *et al.*, 2007):

$$\begin{cases} \vec{V}_i^{k+1} = \vec{V}_i^k + \vec{U}(0,\phi_1) \otimes \left(\vec{P}_{best,i}^k - \vec{X}_i^k\right) + \vec{U}(0,\phi_2) \otimes \left(\vec{G}_{best}^k - \vec{X}_i^k\right) \\ \vec{X}_i^{k+1} = \vec{X}_i^k + \vec{V}_i^{k+1} \end{cases} \quad (6.1)$$

    2.5 If stopping criteria are met (e.g. maximum number of iterations reached and/or sufficiently good fitness value attained), terminate the iterations.

In Equation (6.1), $\vec{U}(0,\phi_i)$ represents a vector of random numbers uniformly distributed in $[0,\phi_i]$, generated for each particle at each iteration. The symbol $\otimes$ denotes component-wise multiplication. The parameters ϕ_1 and ϕ_2 are commonly referred to as acceleration coefficients, their magnitudes determine the relative influence of the cognitive and social components on the flight of the particle, as further discussed later. A distinctive feature of the standard PSO algorithm (particularly when compared with other heuristic optimization techniques) is that it has relatively few parameters that need to be set for a given problem, and are briefly highlighted in the following sub-sections.

6.3.1 Swarm size

The *swarm size* (i.e. the number of particles, or the population size of the particle swarm) is a key parameter that must be decided upon for any given problem. A primary consideration in the setting of this parameter is the relation between the swarm size and the computational cost of executing the algorithm. While a large swarm size leads to a correspondingly large search space, which is especially desirable for large-scale, multi-dimensional problems, it tends to increase the computational cost of the algorithm. Empirical studies have shown that swarm sizes in the range [20 – 60] tend to cover the majority of problem requirements (Poli *et al*., 2007, Talukder, 2011). The higher end of the range is typically favoured for higher-

dimensional and more complex problems, which may benefit from a larger search space and potentially greater diversity of the population.

6.3.2 Velocity update

The *velocity update formula* comprises three components: (1) the *inertial component*, $\vec{V}_i^k$, which acts as a momentum component that induces the particle to maintain its current direction of motion, thus lessening the likelihood of drastic changes in the particle's direction; (2) the *cognitive component*, $\vec{U}(0,\phi_1)\otimes\left(\vec{P}_{best,i}^k - \vec{X}_i^k\right)$, which constitutes a memory of personal good past performance, and biases a particle towards search regions where it performed best in the past; and (3) the *social component*, $\vec{U}(0,\phi_2)\otimes\left(\vec{G}_{best}^k - \vec{X}_i^k\right)$, which represents the influence of the rest of the swarm on a particle's flight trajectory, and serves to bias the particle's movement towards the global best position of the entire swarm. The relative influence of these three components on the overall velocity update is very impactful on the performance of the algorithm, and a number of cases that may hypothetically arise can be identified. In the absence of the inertial component, for instance, past velocity has no influence on future velocity, and particles may be susceptible to abrupt changes in the direction of motion, with potentially adverse impact on the algorithm's performance. When the cognitive component is much less than the social component (i.e. $\phi_1 << \phi_2$) or even non-existent altogether, the particle is almost wholly drawn towards the global best position, with little to no influence of its past good performance. In the opposite case, when the social component is much less than the cognitive component (i.e. $\phi_1 >> \phi_2$) or even non-existent altogether, then the personal best performance is the dominant factor in guiding the particle's future trajectory in the search space. When neither cognitive nor social component contributes to the velocity update (i.e. $\phi_1 = \phi_2 = 0$), the particle's future movement is determined only by the inertial velocity component. When the cognitive and social components are of about the same size and non-zero (i.e. $\phi_1 = \phi_2 > 0$), their overall influence on the particle's search trajectory is essentially an arithmetic mean of the two components. The latter case (i.e. $\phi_1 = \phi_2 > 0$) is found to work well for most applications, with a typical setting being $\phi_1 = \phi_2 = 2$ (Talukder, 2011).

6.3.3 Neighbourhood topology

The interconnectivity among the particles, also referred to as the *neighbourhood topology*, determines the extent of social interaction among the particles, and has a considerable impact on the performance of the PSO algorithm. The PSO model given

in Equation (6.1) is referred to as the *Gbest*-model. In this model, each particle has the rest of the swarm as its neighbour, and is thus a fully connected topology. Another commonly considered neighbourhood topology is the *Lbest* (local-best) model, in which each particle is connected only to the particles adjacent to it (for example, in an array with *i* representing the array index, particle *i* would have as its neighbours particles *i-1* and *i+1*). An important consideration regarding neighbourhood topologies is the impact on the convergence characteristics of the algorithm. A fully connected topology (such as *Gbest*) typically has a high number of interactions, and is thus likely to have fast convergence, although fast convergence may also increase the likelihood of premature convergence (to a local minimum). A small-neighbourhood topology (such as *Lbest*) may exhibit slow convergence, but may also be less susceptible to premature convergence to a local optimum. The problem type (e.g. whether unimodal or multimodal) may also influence the choice of the neighbourhood topology.

6.3.4 Number of iterations

The *number of iterations* needed to obtain a good result is another key parameter that needs proper consideration. It is desirable to allow for a sufficient number of iterations to give the algorithm enough room to find a good solution. The larger the problem size, the higher the number of iterations that are likely to be needed. On the other hand, an excessive number of iterations may lead to excessively high execution time of the algorithm. The setting of the maximum number of iterations is thus likely to be problem-dependent, and obviously involves a compromise between two somewhat mutually exclusive objectives (i.e. the objective of finding a good solution, which may entail a large number of iterations, and the objective of limiting the execution time of the algorithm, which may benefit from a small number of iterations).

6.3.5 Initialization of particle positions and velocities

Initialization of the particle positions and velocities is another important consideration that generally has a significant influence on the performance of the PSO algorithm. As mentioned earlier, PSO is a population-based stochastic algorithm, factors which play a key role in the initialization of the algorithm. The issue of initialization of the PSO algorithm parameters is further discussed in section 6.4.5.

The next section focuses on some practical implementation aspects of the PSO algorithm, which, when combined with the basic algorithm presented in this section, leads to a more practical algorithm for implementation purposes.

6.4 Implementation aspects of the PSO algorithm

The algorithm presented in the previous section requires a few enhancements and additional considerations in order to improve its efficiency and effectiveness for practical implementation purposes. Particularly, the following implementation aspects are discussed in this section (Freitas *et al.*, 2020; Talukder, 2011; Poli *et al.*, 2007):

- Balancing the exploration/exploitation trade-off.
- Controlling the velocity to improve convergence characteristics by means of:
    - Velocity clamping.
    - Inertia weight.
    - Constriction coefficient.
- Initialization of algorithm parameters.
- Termination conditions for the algorithm.

6.4.1 Balancing the exploration/exploitation trade-off

Two important characteristics of any population-based optimization technique are exploration and exploitation. Exploration refers to the algorithm's ability to cover as wide an area of the search space as possible, in order to enhance the algorithm's likelihood of finding the globally optimal solution. Exploitation refers to the algorithm's ability to concentrate the search on an area of the search space that seems to be promising. A good balance between the two characteristics holds the key to the effectiveness and efficiency of the algorithm. Generally, the early phase of the algorithm's execution can largely be classified as the exploratory phase, when the algorithm is "encouraged" to widen its search as far as possible. As the iterations progress, the algorithm should gradually transition into the exploitative phase, when the search can possibly be narrowed down to promising areas of the search space, which implies a relatively more fine-tuned search. Control of the velocity plays an important role in achieving this exploration/exploitation balance. While large velocity adjustments may be considered to be desirable during the exploratory phase, they may need to be significantly dampened in the exploitative phase.

Velocity control is actually a critical component that has a large bearing on the efficiency of the PSO algorithm, a fact that was recognized quite early in the developmental stages of the algorithm, as discussed in the following sub-sections (Poli *et al*., 2007).

6.4.2 Velocity clamping

Effective velocity control is important not only for preventing possible divergence of the algorithm (which would occur if the velocity were to be allowed to build up uncontrollably), but it also impacts the speed of convergence of the algorithm, the

exploration/exploitation balance, and the ability to find a quality solution within a reasonable amount of time. In fact, in an effort to prevent potential velocity explosion in certain contexts, the use of *velocity clamping*, was suggested, which essentially places bounds on the value of the velocity update, V_i^k in Equation (6.1), so that it is kept within the range $[-V_{\max}, +V_{\max}]$. While preventing potential velocity explosion and helping to keep the particles within the boundaries of the search space, this approach has been found to have a number of significant drawbacks. One obvious issue is the choice of the optimal value of the $V_{\max}$, which tends to be problem-dependent, and may require extensive empirical studies to establish a suitable value to use for a given application. Additionally, clamping the velocities may actually negatively impact the algorithm's ability to converge, perhaps causing the algorithm to be trapped in local minima, particularly when $V_{\max}$ is set improperly for a given problem. In some studies, $V_{\max}$ was observed to simply chop off the particle's oscillations, rather than reduce the granularity of the search as typically would be desirable in an exploitative phase of the algorithm, which results in poor convergence characteristics (Poli *et al*., 2007). On account of these and other performance issues, the use of an "inertia weight" as a velocity control mechanism was suggested, as discussed in the following sub-section.

6.4.3 Inertia weight

The limitations of velocity clamping as discussed in the preceding sub-section encouraged the search for alternative ways of regulating the velocity update to achieve exploration/exploitation balance, prevent velocity explosion, and enhance the convergence characteristics of the algorithm. Efforts in this direction led Shi and Eberhart (1998) to suggest a modification to the velocity update formula (given in Equation 6.1) that applies a scaling to the inertial component of the velocity update formula, as given by Equation (6.2).

$$\vec{V}_i^{k+1} = \omega \vec{V}_i^k + \vec{U}(0,\phi_1) \otimes \left(\vec{P}_{best,i}^k - \vec{X}_i^k\right) + \vec{U}(0,\phi_2) \otimes \left(\vec{G}_{best}^k - \vec{X}_i^k\right) \qquad (6.2)$$

with the term ω referred to as the "inertia weight." One interpretation of the inertia weight that has been suggested is that it resembles a "fluidity" of the medium in which the particles move, such that it seems appropriate to set it to a relatively high value in the early stages of the algorithm execution, which would correspond to a low-viscosity medium, so as to promote exploration, then gradually decrease it as the iterations progress, thereby transitioning the algorithm into an exploitative phase. With this scheme, a linearly decreasing inertia weight has been found to provide good results for many applications, which is adjusted according to Equation (6.3).

$$\omega^{k+1} = \omega_{\max} - \left(\frac{\omega_{\max} - \omega_{\min}}{\max\_iter} \right) k, \qquad \omega_{\max} > \omega_{\min} \tag{6.3}$$

where $\omega_{\max}$, $\omega_{\min}$ are the initial and final values of the inertia weight, respectively, k, $\max\_iter$, are the current iteration number and maximum number of iterations respectively. Initial and final values of the inertia weight that have been found to work well are $\omega_{\max} = 0.9$, $\omega_{\min} = 0.4$.

The linear monotonic decrement in the inertia weight as expressed by Equation (6.3) has the drawback of being quite rigid, in the sense that adjustment is only in one direction, and so may adversely impact the diversity of search (i.e. particularly in the exploratory phase). Naturally, other dynamic inertia weight adjustment schemes have been experimented with. Eberhart and Shi (2000) for example, applied fuzzy logic to the adaptation of the inertia weight, where the inputs to the fuzzy system were taken to be the fitness value of the global best position and the current value of the inertia weight, and the output represented the suggested adjustment to the value of the inertia weight, based on the membership function classification of the input variables (the possible fuzzy classifications being low, medium or high). The authors reported significantly improved PSO performance using this scheme. It is worth noting that when $\omega > 1$, the velocity tends to increase uncontrollably, and when $\omega << 1$, the influence of the previous velocity on the current position adjustment becomes nearly insignificant, which may make the particle susceptible to abrupt changes in the direction of motion, similarly to what was discussed in section 6.3. The inertia weight approach to velocity regulation has proven to be so effective as to render the velocity clamping (discussed in section 6.4.2) largely unnecessary, which then also eliminates the many issues it introduces into the PSO algorithm (Poli *et al.*, 2007).

6.4.4 Constriction coefficient

The need to dampen the particle dynamics by somehow restraining velocity build-up has long been recognized as a key factor to ensuring the convergence of the PSO algorithm. Otherwise (without some form of velocity restraint), the velocity rapidly increases to unacceptable levels within just a few iterations of the algorithm. The search for alternative mechanisms to achieve this (besides the velocity clamping and the inertia weight discussed in sections 6.4.2 and 6.4.3 respectively) led Clerc and Kennedy (2002) to suggest the use of a constriction coefficient, χ, which would be applied to the velocity update formula, as given in Equations (6.4) and (6.5).

$$\begin{cases} \vec{V}_i^{\,k+1} = \chi\left(\vec{V}_i^{\,k} + \vec{U}(0,\phi_1) \otimes \left(\vec{P}_{best,i}^{\,k} - \vec{X}_i^{\,k}\right) + \vec{U}(0,\phi_2) \otimes \left(\vec{G}_{best}^{\,k} - \vec{X}_i^{\,k}\right)\right) \\ \vec{X}_i^{\,k+1} = \vec{X}_i^{\,k} + \vec{V}_i^{\,k+1} \end{cases} \tag{6.4}$$

$$\chi = \frac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}}, \quad where \quad \phi = \phi_1 + \phi_2 > 4 \tag{6.5}$$

It is worth noting that unlike the inertia weight (which scales only the inertial component of the velocity update, $\vec{V}_i^k$), the constriction coefficient is applied to all the three terms of the velocity update formula. Parameters that have been found to work well for the constriction coefficient are $\phi_1 = \phi_2 = 2.05$, such that $\chi \cong 0.7298$. With these parameter settings, the previous velocity is scaled by about 0.7298, and the cognitive and social components of the velocity update formula are each scaled by a random number with an upper limit of $0.7298 \times 2.05 \cong 1.49618$. This form of velocity constriction enables convergence of the algorithm without the need for velocity clamping. Interestingly, the inertia-weight and constriction-coefficient approaches to velocity regulation can be seen to be algebraically equivalent when the variable transformations $\omega \leftrightarrow \chi$ and $\phi_i \leftrightarrow \chi\phi_i$ are considered. In this case, the corresponding parameter settings for the inertia weight approach (considering the suggested values $\phi_1 = \phi_2 = 2.05$) would be a fixed inertia weight $\omega = 0.7298$ and the upper bounds for the acceleration coefficients would be $\phi_1 = \phi_2 = 1.49618$.

6.4.5 Initialization of the PSO algorithm parameters

PSO is a population-based algorithm and makes use of stochastic processes to perform the search for the solution to an optimization problem. Proper initialization of the PSO parameters is important for the effectiveness and efficiency of the algorithm. Some of the parameters that need to be initialized or otherwise have their values determined have been discussed in the preceding sections, particularly the *acceleration coefficients*, the *inertia weight* and the *constriction coefficient*. Deciding upon the *swarm size* has also been discussed (in section 6.3). Once the swarm size has been determined, the particle positions and velocities have to be initialized. A key consideration here is the diversity of the initial population, that is, how much of the search space the initial population covers, and how well-distributed the particles are throughout the search space. The particle positions are usually initialized by means of a uniform random distribution over the search space, while the particle velocities can be set to zero, since the randomization of the particle positions ensures the randomization of the initial directions of motion as well. It may be possible to initialize the velocities by means of a uniform random distribution as well, in which case care must be taken that the magnitudes of the initial velocities are not too large (Talukder, 2011). Otherwise this may adversely impact the convergence characteristics of the

algorithm. A formula for initializing the position of each particle X_i that can be appropriate for most applications is:

$$X_i^0 = X_{\min,i} + r_i\left(X_{\max,i} - X_{\min,i}\right), \quad r_i \in U(0,1) \tag{6.6}$$

where $X_{\min,i}$ and $X_{\max,i}$ denote the lower and upper bounds of the magnitude of the particle position X_i respectively, and $r_i \in U(0,1)$ is a uniformly distributed random number bounded between 0 and 1.

6.4.6 Termination conditions for the algorithm

As PSO is an iterative search algorithm, a mechanism is needed to establish when the search for the solution may be terminated, whether successfully (i.e. optimal solution found) or not (i.e. the algorithm failed to converge). A number of termination criteria may be considered, for example, once:

- There is no appreciable (improving) change in the fitness value of the global best position over a number of iterations.
- The change in the global best position becomes insignificant over a number of iterations.
- The predetermined maximum number of iterations is reached.

These criteria can be used in any combination (either one or several of them). With regards to the maximum number of iterations, setting the value too small may lead to premature termination of the search, and setting it too large may make the computational cost of running the algorithm prohibitive, as discussed earlier.

6.5 PSO algorithm applied to the VVO problem

The Volt/VAR optimization (VVO) problem formulation has been presented in chapter 3, and the solution based on the primal-dual interior-point method (PDIPM) is detailed in chapter 5. In this section, the PSO algorithm discussed in the preceding sections is adapted for application to the VVO problem. The rectangular formulation of the VVO problem presented in section 3.2.3 is used. In applying the PSO algorithm to any optimization problem, the mechanics of the algorithm have to be mapped to the structure of the optimization problem. Particularly, the mapping needs to be made between the particle positions and velocities, and the decision vector of the optimization problem, along with the adjustment process of the decision vector in the search for the optimal solution to the problem. For the VVO problem, the mapping can be stated as follows:

- The decision vector comprises the generator voltages, expressed in rectangular coordinates; thus, each particle is constructed by combining the real and imaginary components of all the generator voltages in the system.
- For the slack bus, the phase angle is required to be maintained at a predetermined constant value, and so the imaginary component of the slack-bus voltage does not form part of the decision vector.
- The length (i.e. number of elements) of each particle is thus $2n_g - 1$, where n_g represents the number of generators in the system, including the slack-bus generator.
- The velocities of the particles represent the step size adjustments to the decision-vector components (i.e. particle positions), and their computation is one of the main tasks performed in each iteration of the algorithm.

The steps of the PSO algorithm applied to the VVO problem can be outlined as follows:

Step 1:

Load the system parameters: this includes the (1) bus voltages in rectangular coordinates, the (2) generator scheduled active generation outputs (reactive generation outputs are set to zero, since they are unscheduled), (3) the load active and reactive power demands, and the (3) line impedance (i.e. resistance and reactance) data. The impedance and bus connectivity matrices are computed on the basis of the input line data.

Step 2:

Initialize the PSO algorithm parameters: this includes the acceleration coefficients (ϕ_1, ϕ_2), the swarm size, the problem dimension (i.e. number of elements comprising each particle), and the maximum number of iterations.

Step 3:

Compute the initial particle positions and velocities: the particle positions are initialized according to Equation (6.6), which is reformulated below as it applies to the VVO problem:

$$V_{gen\_i}^{0} = V_{gen\_i}^{\min} + r_i\left(V_{gen\_i}^{\max} - V_{gen\_i}^{\min}\right), \quad r_i \in U(0,1) \tag{6.7}$$

where $V_{gen\_i}^{0}$ is the i^{th} generator's initial voltage magnitude, and $V_{gen\_i}^{\min}$, $V_{gen\_i}^{\max}$ are the minimum and maximum generator voltage magnitudes respectively. For the VVO problem, the generator-bus voltage magnitudes have the bounds:

$$0.95 \le V_{gen} \le 1.1 \tag{6.8}$$

So based on Equation (6.8), the initial generator voltage magnitude (which corresponds to particle position X_i in Equation 6.6) can be set according to Equation (6.7) as:

$$V_{gen\_i}^{0} = 0.95 + 0.15 \cdot r_i \tag{6.9}$$

In effect, for the rectangular-coordinate representation of the generator voltages, Equation (6.9) is actually used to compute the real component of the voltage (e_i), after which the imaginary component (f_i) is computed by means of Equation (6.10), as also discussed in section 3.2.1.

$$V_i^2 = e_i^2 + f_i^2 \tag{6.10}$$

Initialization of the particle positions by means of Equations (6.9) and (6.10) ensures that they are all feasible with respect to the bound constraints according to Equation (6.8).

The initial velocities are set to zero for this study. As discussed in section 6.4.5, they could be set to uniformly distributed random values as well; the rationale for setting them to zero in this study is that the randomized initial positions provide sufficient diversity, in terms of both magnitude and direction, to enable a well-diversified search of the search space for the PSO algorithm.

Step 4:

Compute each particle's fitness value based on initial positions: the objective function for the VVO problem in rectangular coordinates is given by Equation (3.8), which is the transmission real power loss function, restated below for ease of reference:

$$f(X) = P_L = \sum_{i \in N} \sum_{\mathrm{j} \in \mathrm{N}} G_{ij} \left[\left(e_i - e_j \right)^2 + \left(f_i - f_j \right)^2 \right] \tag{6.11}$$

The constraints considered in this study are all bound constraints (Equations 3.10e, 3.10f in section 3.2.3). Each bound constraint is handled such that when it violates its bound constraint, its value is set to the violated bound. The generator voltage magnitude, for example, has its value set according to:

$$V_{gen\_i} = \begin{cases} V_{gen\_i}^{\min} & if \quad V_{gen\_i} < V_{gen\_i}^{\min} \\ V_{gen\_i} & if \quad V_{gen\_i}^{\min} \le V_{gen\_i} \le V_{gen\_i}^{\max} \\ V_{gen\_i}^{\max} & if \quad V_{gen\_i} > V_{gen\_i}^{\max} \end{cases} \tag{6.12}$$

Before computing the fitness values using Equation (6.11), limit violations are checked and corrected according to Equation (6.12).

After this initial computation of fitness values, the (initial) personal best position and corresponding fitness value of each particle is set to the current (i.e. initial) position and its corresponding fitness value. That is, for each particle, X_i:

$$p_{best,i}^0 = X_i^0 \tag{6.13}$$

The initial global best position is determined as the value of $p_{best,i}^0$ giving the best (i.e. minimum) fitness value, determined as:

$$g_{best}^0 = \arg\min\left(f\left(p_{best,i}^0\right)\right),\ i = \overline{1,...,p} \tag{6.14}$$

where p is the number of particles in the swarm, and the operator $\arg\min()$ returns the argument $p_{best,i}^0$ that yields the minimum value of the fitness function $f(p_{best,i}^0)$.

Step 5:

Compute the Newton-Raphson load flow: the effect of the PSO algorithm is to adjust the generator voltage magnitude set-points at each iteration of the algorithm. Similar to the approach used in the PDIPM-VVO algorithm (please refer to section 5.2.4), a load flow computation is run at each iteration of the PSO algorithm in order to determine the load-bus voltages, subject to the active and reactive power balance equations (Equations 3.10b and 3.10c, section 3.2.3). The Newton-Raphson algorithm presented in section 5.2.4 is used to compute the load flow at each iteration of the PSO algorithm. For the load flow computation, the global best position (g_{best}) is used, since this is assumed to be the best available solution.

Step 6:

Recompute the objective function value: a converged load flow computation implies that the solution obtained in step 4 is feasible with respect to both the equality and inequality constraints of the VVO problem. Recomputing the objective function value following the load flow computation is meant to track the objective value of the current best feasible solution of the optimization problem.

Step 7:

Compute the velocity update and adjust the particle positions: this is essentially the beginning of the iterative loop of the PSO algorithm, where the particle velocity is iteratively computed and then used to adjust the particle position. In this study, the velocity and position update are computed on the basis of Equations (6.4) and (6.5), restated below for ease of reference

$$\begin{cases} \vec{V}_i^{k+1} = \chi\left(\vec{V}_i^{k} + \vec{U}(0,\phi_1)\otimes\left(\vec{P}_{best,i}^{k} - \vec{X}_i^{k}\right) + \vec{U}(0,\phi_2)\otimes\left(\vec{G}_{best}^{k} - \vec{X}_i^{k}\right)\right) \\ \vec{X}_i^{k+1} = \vec{X}_i^{k} + \vec{V}_i^{k+1} \end{cases} \tag{6.4}$$

$$\chi = \frac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}}, \quad where \quad \phi = \phi_1 + \phi_2 > 4 \tag{6.5}$$

That is, the constriction coefficient is used as the velocity regulation mechanism, as discussed in section 6.4.4.

Step 8:

Compute the fitness value of each particle and update the personal and global best positions: after adjusting the particle positions, limit violations are checked and corrected for using Equation (6.12), after which the fitness value of each particle is computed using Equation (6.11). Once the fitness value of each particle has been computed, the personal best $p_{best,i}^k$ of each particle i is updated as follows:

$$p_{best,i}^k = \begin{cases} X_i^k \ \ if \ \ f(X_i^k) < f(X_i^{k-1}) \\ p_{best,i}^{k-1} \ \ if \ \ f(X_i^k) \geq f(X_i^{k-1}) \end{cases} \tag{6.15}$$

Then the global best position is updated according to Equation (6.14), restated here for ease of reference:

$$g_{best}^k = \arg\min\left(f\left(p_{best,i}^k\right)\right), \ i = \overline{1,...,p} \tag{6.16}$$

Step 9:

Recompute the Newton-Raphson load flow: similar to step 5, perform a load flow computation to determine the new load-bus voltages, with the global best position (computed in step 8) acting as the new generator voltage set-points.

Step 10:

Recompute the objective function value: similar to step 6, the objective function value is recomputed to account for the change in the load-bus voltages due to the load flow computation. The recomputed objective function value constitutes the optimal value of the current best feasible solution. The algorithm is assumed to have advanced in the desired direction if the recomputed objective function value is better (i.e. less) than the one computed in the previous iteration.

Step 11:

Check for convergence of the algorithm: the PSO algorithm is considered to have converged successfully to an optimal solution when there isn't an appreciable change in the objective function value over a number of successive iterations, and the current objective function value is better than the initial value. Otherwise it is terminated with a result of "failure" when it fails to achieve an objective function value minimization within the predetermined number of iterations. In summary, the PSO algorithm will be terminated if any one of the following two conditions is satisfied:

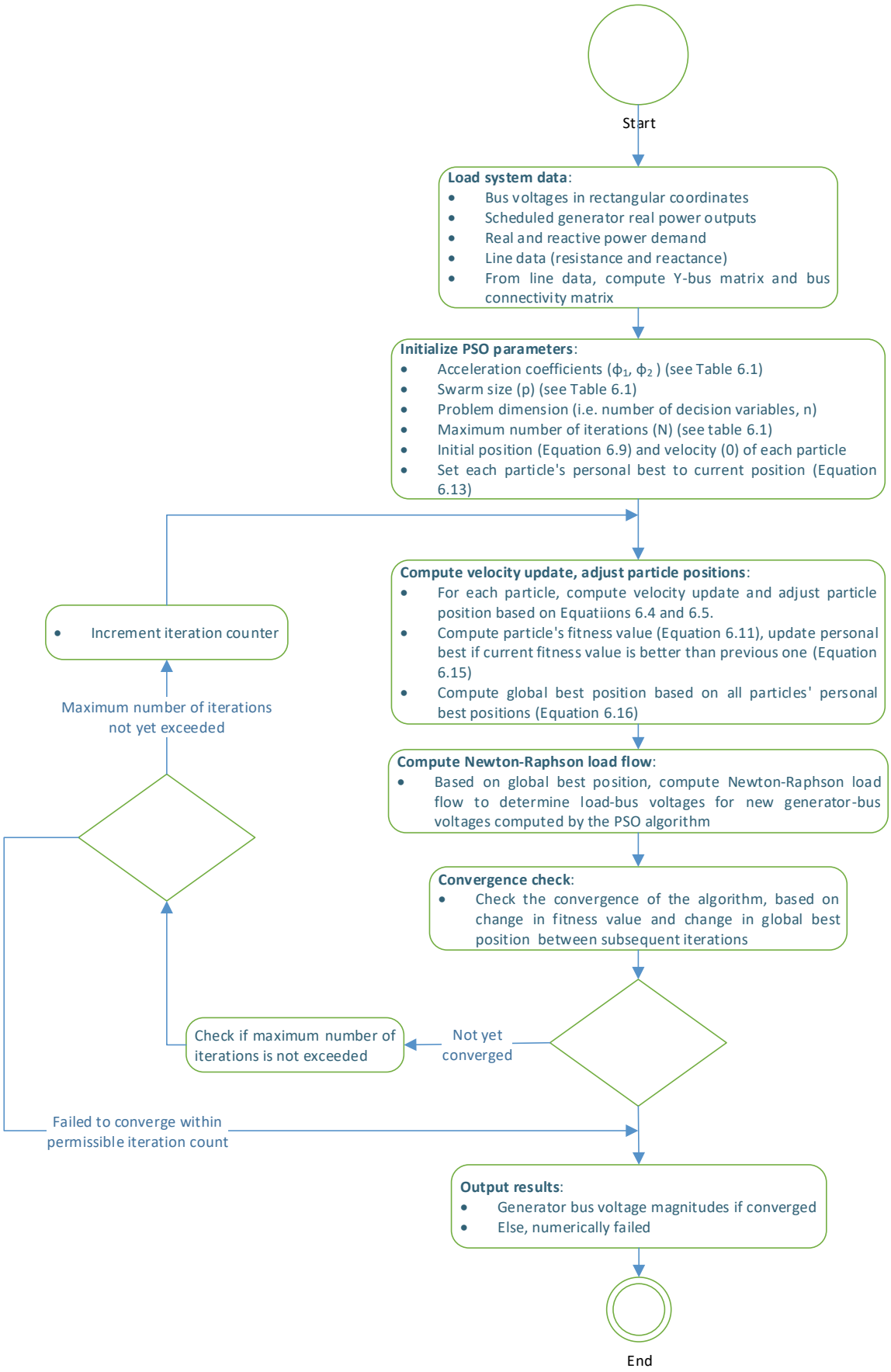

Fig. 6.3: Flowchart of the particle swarm optimization algorithm

1. There is no appreciable improvement in the objective function value over a number of successive iterations, and the current objective function value is better than the initial one
2. The maximum number of iterations has been reached

If neither of the two conditions is satisfied, the iteration counter (k) is incremented and the algorithm loops back to step 7, repeating steps 7 to 11 until termination conditions are satisfied.

The flowchart in Figure (6.3) summarises the steps of the PSO algorithm outlined above as adapted for application to the VVO problem.

6.5.1 Case studies

There are quite a number of parallels in the implementation of the PSO–based VVO algorithm presented in this chapter and the PDIPM-VVO algorithm presented in the previous chapter, for example:

- Use of the rectangular formulation of the VVO problem presented in chapter 3.
- Incorporation of the rectangular–form Newton–Raphson load flow computation in each iteration of the VVO algorithm, as indicated in the flowchart in Figure (6.3).
- Performance analysis of the algorithm by means of the 3–bus, 6–bus, IEEE 14–bus, IEEE 30–bus and IEEE 118–bus test systems.

Results of applying the PSO algorithm to each of the case studies are presented in the following sub-sections. Performance analysis of the PSO algorithm focuses on the following performance metrics:

- Magnitude of loss minimization.
- Voltage profile improvement due to the Volt/VAR optimization.
- Impact of particle swarm size on the quality of the solution and on the number of iterations needed for the algorithm to converge, and the resulting execution time.

The PSO parameters used in the case studies are given in Table (6.1) (values of acceleration coefficients are according to Poli *et al.*, 2007). It can be noted that the swarm size has been specified as a range of values. As stated in the last bullet point above, one key metric analysed in this study is the impact that the particle swarm size has on the quality of the solution, and on the computational cost of the algorithm, as measured by the number of iterations and the execution time of the algorithm. For each case study, the case is run for values of the swarm size ranging from 10 to 50, in increments of 10. Since PSO is a stochastic algorithm, the approach taken is to

make several runs of the algorithm for each value of the swarm size. The statistical variance can then be assessed by looking at the minimum, maximum and average values of the key results. Such statistical analysis is likely to yield insightful information. The MATLAB programs for the case studies presented in the following sub-sections are presented in Appendix C.

Table 6.1: PSO algorithm parameters used in the VVO case studies

Parameter	***Setting***
Cognitive acceleration coefficient, ϕ_1	2.05
Social acceleration coefficient, ϕ_2	2.05
Swarm size, p	10 – 50
Maximum number of iterations, $\max\_iter$	200

6.5.2 Case study 1: 3-bus power system

This case study parallels the one presented in section 5.3.1 for the PDIPM algorithm. The case data is the same as that used in chapter 5. The analysis in this sub-section follows the points outlined at the end of the previous sub-section. The PSO algorithm is used to solve the VVO problem for the 3-bus power system, similarly to what was done using the PDIPM algorithm. The key performance metrics, as outlined above, are the magnitude of loss minimization, the voltage profile improvement, and the computational efficiency of the algorithm. Several runs of the algorithm have been executed for different values of the swarm size, averaging about five runs for each value of the swarm size. The results are then averaged and tabulated, as presented in Table (6.2).

Table 6.2: 3-bus system summary of PSO-VVO algorithm simulation results

Swarm size	**Initial loss (p.u.)**			**Final loss (p.u.)**			**Number of iterations**			**Run time (sec)**			**Average % loss reduction**
	Min	**Max**	**Average**	**Min**	**Max**	**Average**	**Min**	**Max**	**Average**	**Min**	**Max**	**Average**	
10	0.0255	0.1490	0.0735	0.0156	0.0180	0.0160	12	14	14	0.0159	0.0221	0.0195	78.78
20	0.0185	0.1843	0.0716	0.0156	0.0170	0.0158	3	19	13	0.0057	0.0294	0.0167	78.22
30	0.1157	0.2625	0.16088	0.0156	0.0165	0.0160	8	17	13	0.0138	0.0312	0.0231	90.30
40	0.0159	0.1482	0.07484	0.0156	0.0185	0.0162	8	32	17	0.0153	0.0576	0.0327	79.16
50	0.0180	0.1189	0.0693	0.0156	0.0171	0.0159	4	17	11	0.0103	0.0416	0.0247	77.48

What can be deduced from Table (6.3) is that the PSO algorithm consistently yields a substantial real power loss reduction, in the range of about 77% to about 90%. Although the initial power loss for the various runs is quite varied (from as low as

0.018 to as high as 0.184) the final power loss is consistently at around 0.016, which appears to be the globally optimal solution for this cases study. The number of iterations averages at about 14, and appears to be independent of the swarm size, although the average run time appears to increase with the swarm size. Overall for this case study, increasing the swarm size (from 10 to 50) does not seem to lead to an appreciable improvement in the solution quality, and moreover the computational cost does not increase excessively either.

The convergence behaviour of the PSO algorithm for the 3-bus system is depicted in Figure (6.4), which shows traces of the change in the global best position (top trace of Figure 6.4), the change in the fitness value of the global best position (middle trace), and the fitness value of the global best position (bottom trace). It can be seen that this case converges in about six iterations, when the change in the fitness value of the global best position becomes effectively zero.

The voltage profile of the 3-bus system before and after the Volt/VAR optimization is depicted in Figure (6.5) in the form of a bar chart. The voltage profile shows that the post-optimization voltage magnitude is greater than the pre-optimization value for each bus. For buses 1 and 3 which are the generator buses, the voltage magnitudes actually hit their upper limits. This is an expected result, since higher system voltages generally tend to lead to reduced real power loss.

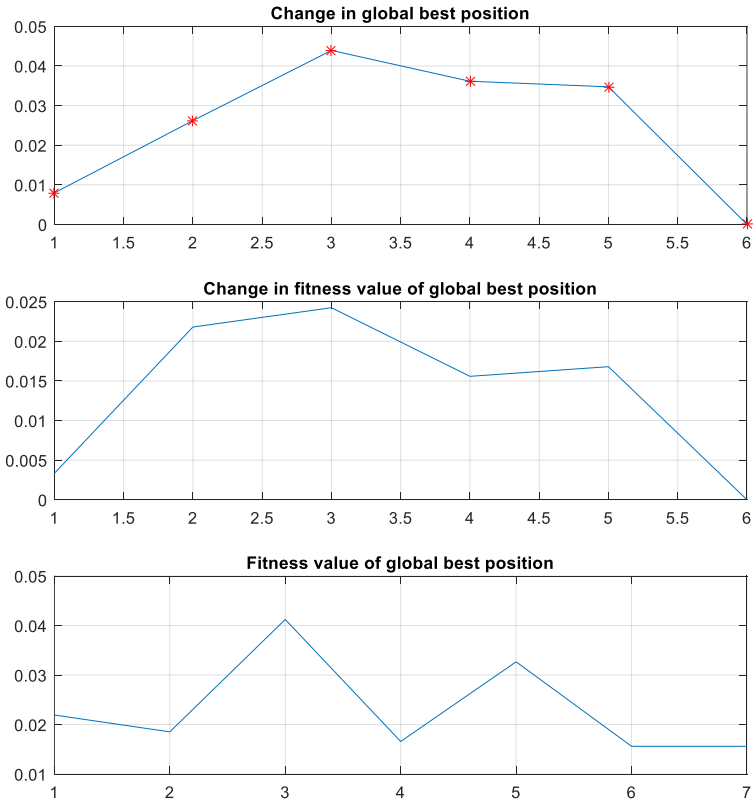

Fig. 6.4: 3-bus system convergence behaviour of the PSO algorithm

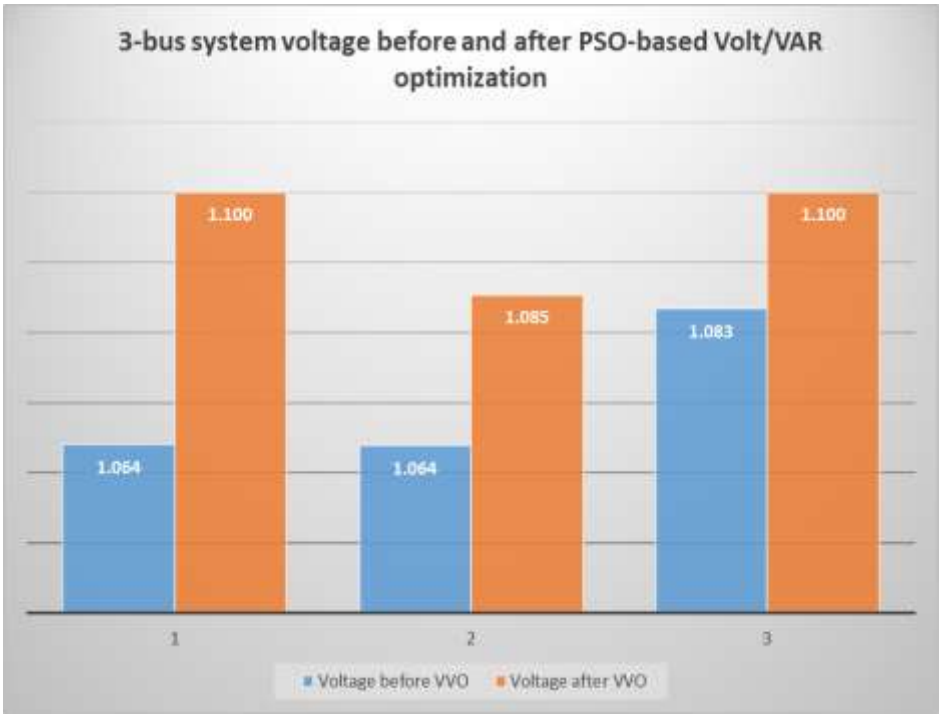

Fig. 6.5: 3-bus system voltage magnitudes before and after PSO-based VVO

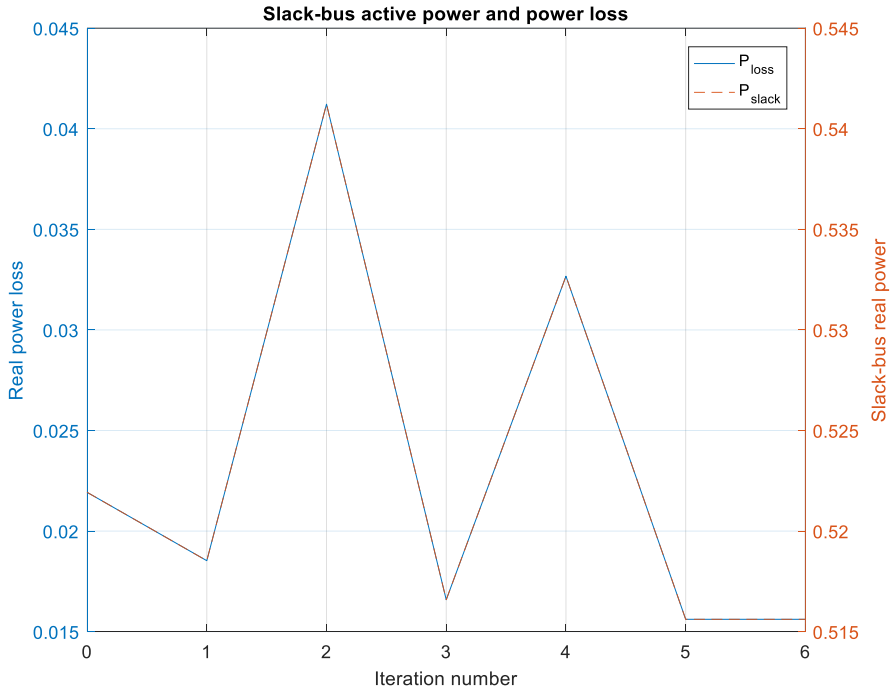

Fig. 6.6: 3-bus system real power loss and slack-bus active power plotted against number of iterations

Figure (6.6) depicts the real power loss trajectory plotted along with the slack-bus active power output against the number of iterations. Similar to the case study in section 5.3.1, the two trajectories coincide, showing that the change in the system real power loss corresponds to an equal change in the slack-bus active power output.

Figure (6.7) compares the real power loss trajectory with that of the total system generated reactive power. It can be seen that the two variable track each other, implying that real power loss minimization simultaneously achieves a reduction in total system reactive power generation as well.

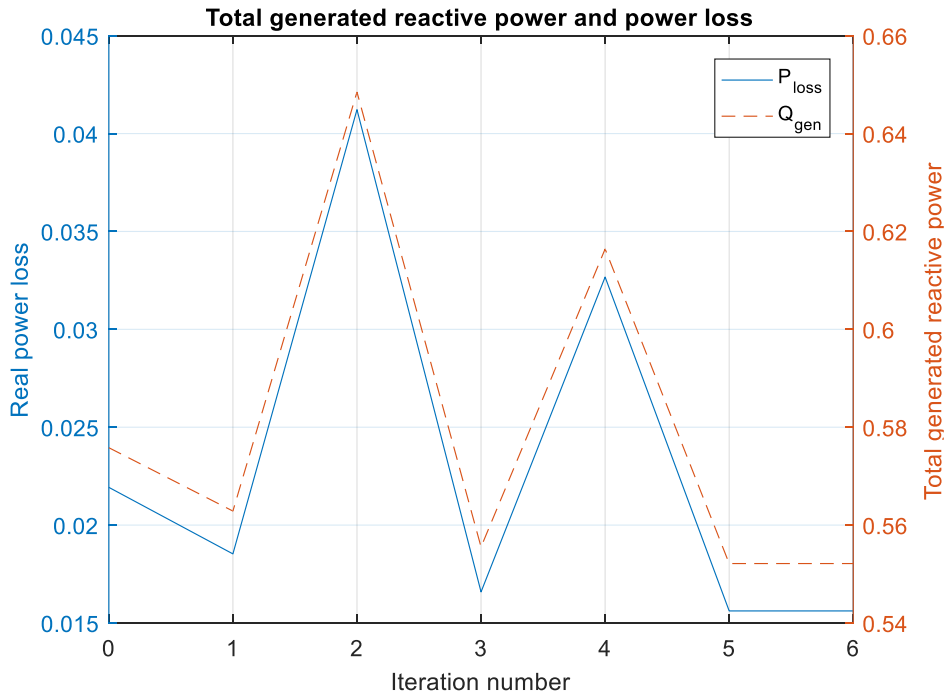

Fig. 6.7: 3-bus system real power loss and total generated reactive power plotted against number of iterations

6.5.3 Case study 2: 6-bus power system

Case study 2 parallels the one presented in section 5.3.2, which is based on a 6-bus power system adapted from Wood *et al.* (2014), having 3 generators, 11 lines and 3 loads. The case data is the same as that used in chapter 5. Results of applying the PSO algorithm to the 6-bus system for solving the VVO problem are presented in Table (6.3) and Figures (6.8) to (6.10).

Table 6.3: 6-bus system summary of PSO-VVO algorithm simulation results

Swarm size	Initial loss (p.u.)			Final loss (p.u.)			Number of iterations			Run time (sec)			Average % loss reduction
	Min	Max	Average	Min	Max	Average	Min	Max	Average	Min	Max	Average	
10	0.1276	0.2441	0.1647	0.1259	0.1339	0.1302	28	71	40	0.0378	0.3638	0.1385	20.95
20	0.1378	0.2885	0.2215	0.1265	0.1318	0.1293	13	102	54	0.0280	0.2056	0.1060	41.62
30	0.1273	0.3275	0.1811	0.1262	0.1336	0.1284	16	56	31	0.0381	0.1363	0.0713	29.10
40	0.1275	0.1670	0.1613	0.1255	0.1302	0.1284	8	32	19	0.0249	0.1531	0.0615	20.40
50	0.1303	0.2212	0.1569	0.1263	0.1332	0.1294	10	133	49	0.0275	0.3185	0.1216	17.53

The results in Table (6.3) reveal a consistent and substantial real power loss reduction by the PSO algorithm, with the lowest (average) percentage loss reduction being 17.93%, and the highest being 41.62%. The average number of iterations lies in the range between 19 and 54 for all cases. There is quite a large dispersion between the average minimum (8) and maximum (133) number of iterations. The execution time shows a similar dispersion (minimum average value of 0.0713 sec and maximum value of 0.1385). As in the 3-bus system case, increasing the swarm size does not seem to significantly influence the solution quality. The impact on the computational cost of the algorithm is also not very noticeable.

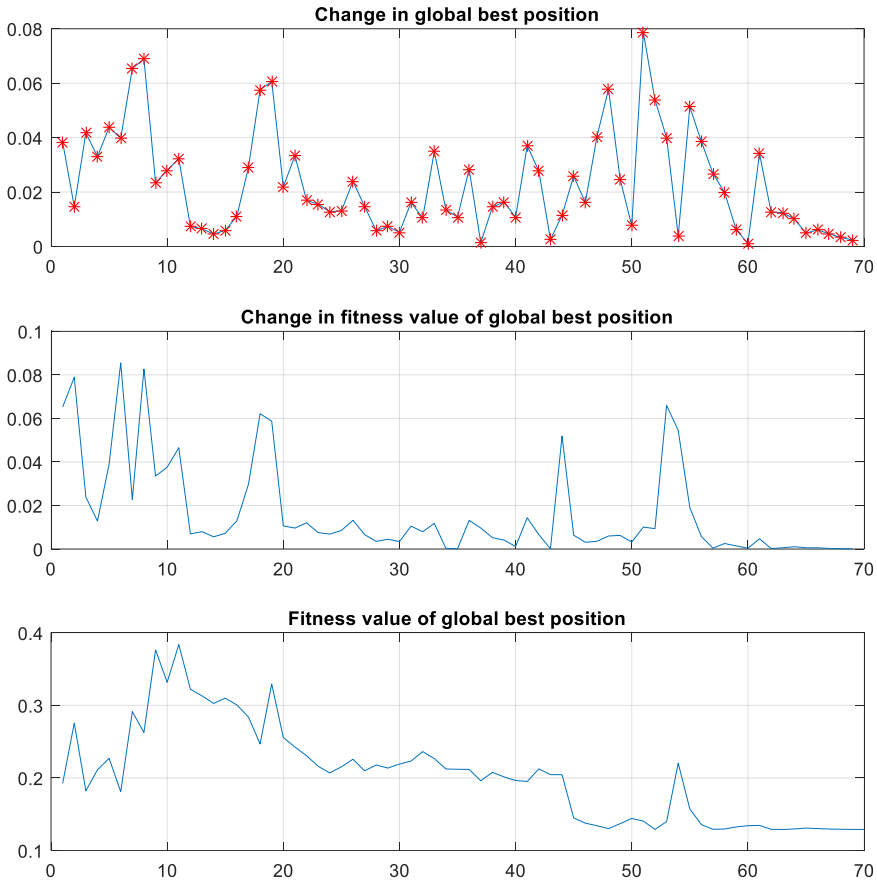

Fig. 6.8: 6-bus system convergence behaviour of the PSO algorithm

The convergence behaviour of the PSO algorithm for the 6-bus system is depicted in Figure (6.8), which plots the change in the global best position, the change in the fitness value of the global best position, and the fitness value of the global best position in the top, middle and bottom traces respectively. This case takes relatively long to converge, about 70 iterations, requiring about 0.13 seconds. This result can be considered to lie on the higher end of the range of the average results presented in Table (6.3).

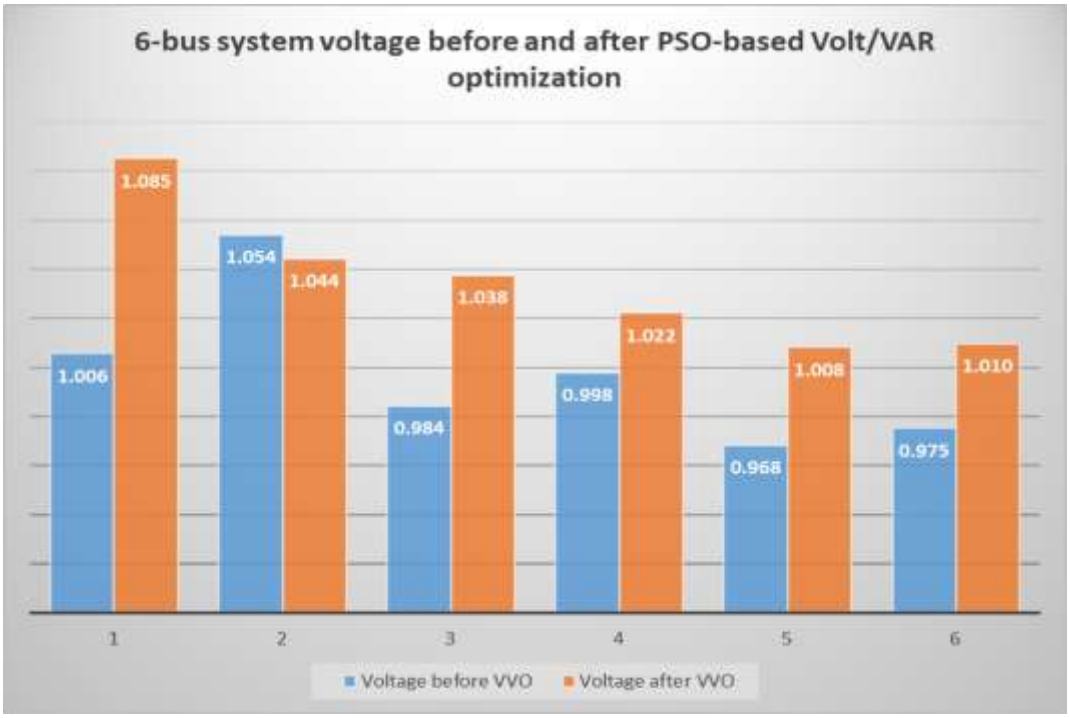

Fig. 6.9: 6-bus system voltage magnitudes before and after PSO-based VVO

The voltage profile of the 6-bus system before and after the Volt/VAR optimization is depicted in Figure (6.9) in the form of a bar chart. The post-optimization voltage magnitudes are greater than the pre-optimization values for all buses, except bus 2, where the pre-optimization voltage is slightly higher than the post-optimization voltage. In all cases, both the pre- and post-optimization voltages are within the range of nominal values (i.e. 0.95 – 1.1).

Figure (6.10) depicts the real power loss trajectory plotted together with the slack-bus active power output against the number of iterations. Unlike the 3-bus system case where these two graphs coincided, this case just shows them to be changing in tandem, again showing the close relationship between the change in the two quantities. The same relationship can be seen in Figure (6.11), which plots the real power loss reduction together with the total system reactive power generation, also showing that real power loss reduction is accompanied by system reactive power generation reduction as well.

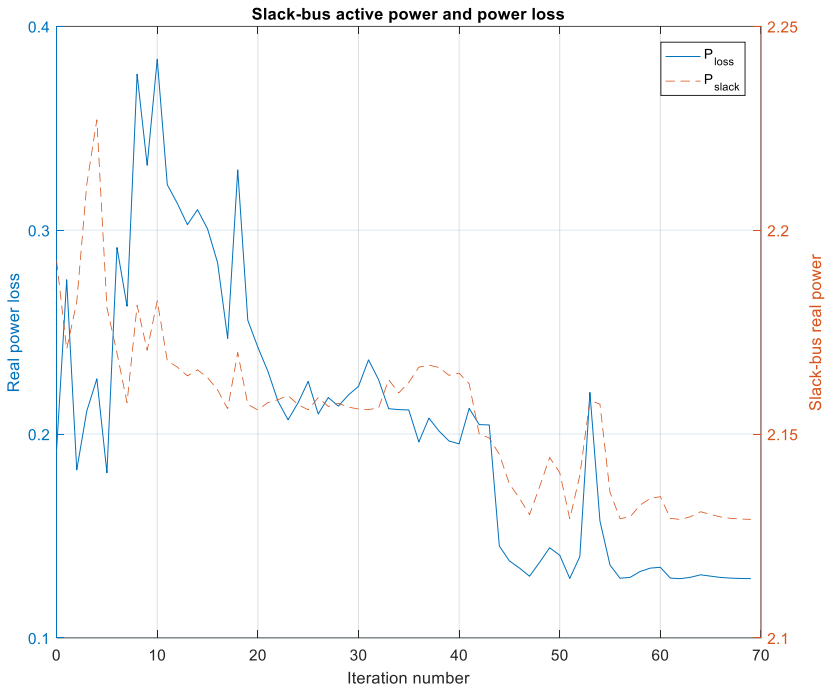

Fig. 6.10: 6-bus system real power loss and slack-bus active power plotted against number of iterations

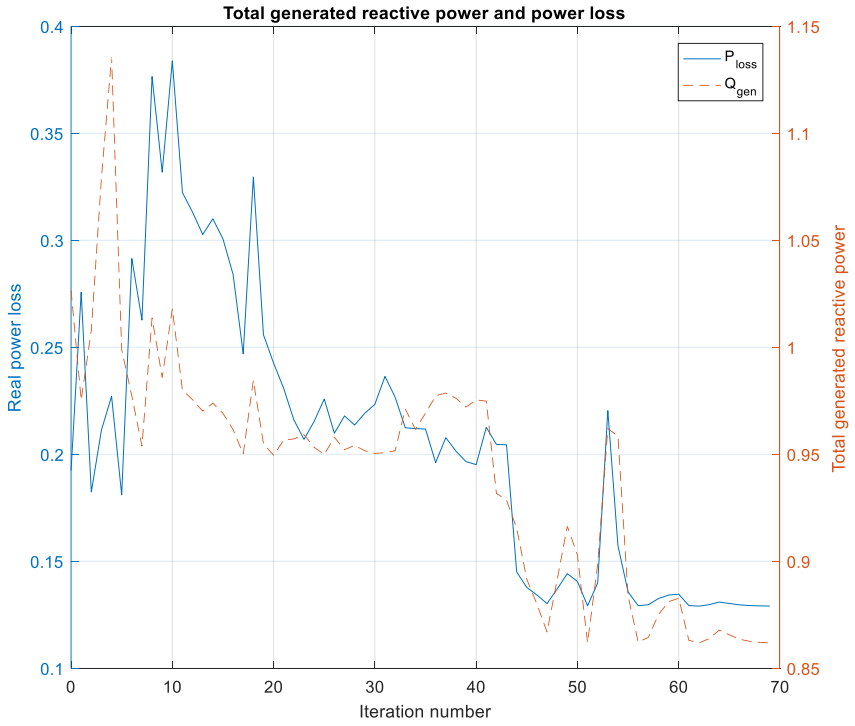

Fig. 6.11: 6-bus system real power loss and total generated reactive power plotted against number of iterations

6.5.4 Case study 3: 14-bus power system

Case study 3 is based on the IEEE 14-bus test system, the same as that considered in section 5.3.3. The network, load and generation data is taken from Zhu (2009), and is the same as that used in chapter 5. Results of applying the PSO algorithm to the 14-bus system for solving the VVO problem are presented in Table (6.4).

Table 6.4: 14-bus system summary of PSO-VVO algorithm simulation results

Swarm size	Initial loss (p.u.)			Final loss (p.u.)			Number of iterations			Run time (sec)			Average % loss reduction
	Min	Max	Average	Min	Max	Average	Min	Max	Average	Min	Max	Average	
10	0.1347	0.2011	0.1613	0.1235	0.1291	0.1279	6	40	21	0.0203	0.2432	0.1196	20.71
20	0.1353	0.1652	0.1439	0.1268	0.1290	0.1282	4	24	11	0.0214	0.1375	0.0662	10.91
30	0.1381	0.2408	0.1873	0.1290	0.1301	0.1293	5	71	41	0.0307	0.5822	0.2867	30.97
40	0.1416	0.1921	0.1613	0.1273	0.1290	0.1287	4	109	29	0.0237	0.6544	0.1791	20.22
50	0.1335	0.2385	0.1595	0.1290	0.1308	0.1294	5	51	21	0.0488	0.2931	0.1394	18.85

Based on the results presented in Table (6.4), the average real power loss reduction ranges from 10.91% in the case of the simulation with 20 particle swarms, to 30.97% in the case of the simulation with 30 particle swarms. The average number of iterations lies in the range between 11 and 49 for all cases. The minimum number of iterations is 4, recorded when the swarm size is 20 and 40, and the maximum number of iterations is 109, recorded in the case of a swarm size of 40. It can be noticed from the results presented in Table (6.4) that while increasing the swarm size does not necessarily lead to increased execution time, there is a direct relationship between the number of iterations and the execution time. Thus, the maximum execution time (0.6544 seconds) is recorded in connection with the swarm size of 40, which also happens to coincide with the maximum number of iterations recorded for all the runs. The average execution time ranges from 0.0662 seconds (for swarm size of 20) to 0.2867 seconds (for swarm size of 30). The lowest absolute real power loss achieved is 0.1235 per-unit, with a swarm size of 10. It can thus be seen here that the global minimum (in the context of the presented results) is attained with a swarm size of 10, implying that increasing the swarm size for this case does not necessarily lead to an improvement in the quality of the solution. It is worth noting that the (average) minimum number of iterations (11) is attained in the case of a swarm size of 20, which also has the minimum average execution time (0.0662 seconds).

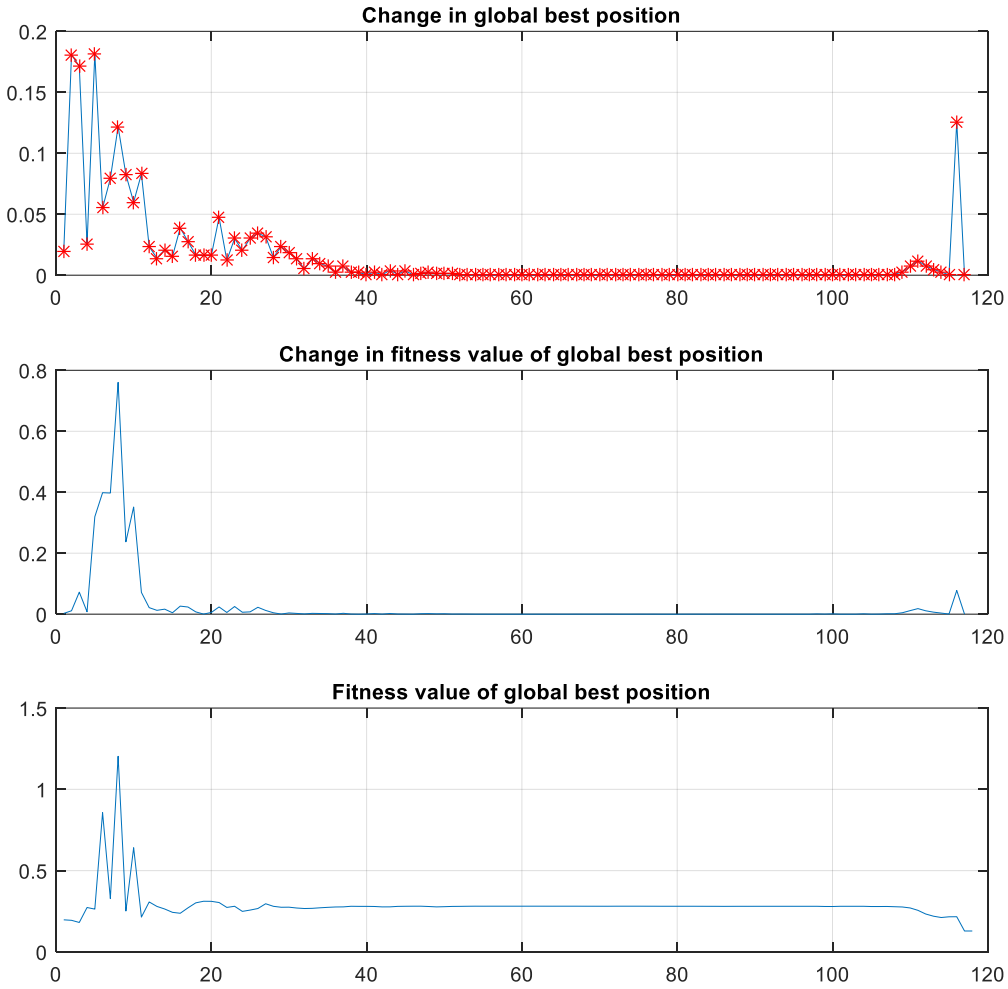

Fig. 6.12: 14-bus system convergence behaviour of the PSO algorithm

The convergence behaviour of the PSO algorithm for the 14-bus system is depicted in Figure (6.12) which, similar to Figure (6.8) for the 6-bus system, plots the changes in the global best position and in the fitness value of the global best position, as well as the fitness value of the global best position. This case takes relatively long to converge, about 120 iterations, requiring about 0.5 seconds. It can be noticed, however, that the change in both the global best position and in its corresponding fitness value is not significant beyond the 40^{th} iteration, although the termination conditions for the algorithm are satisfied only much later, leading to the high number of iterations.

The voltage profile of the 14-bus system before and after the Volt/VAR optimization is depicted in Figure (6.13) in the form of a radar chart. The post-optimization voltage magnitudes are greater than the pre-optimization values for all buses. It can be seen also for this case study that both the pre- and post-optimization voltages are within the range of nominal values (i.e. 0.95 – 1.1) for all the buses.

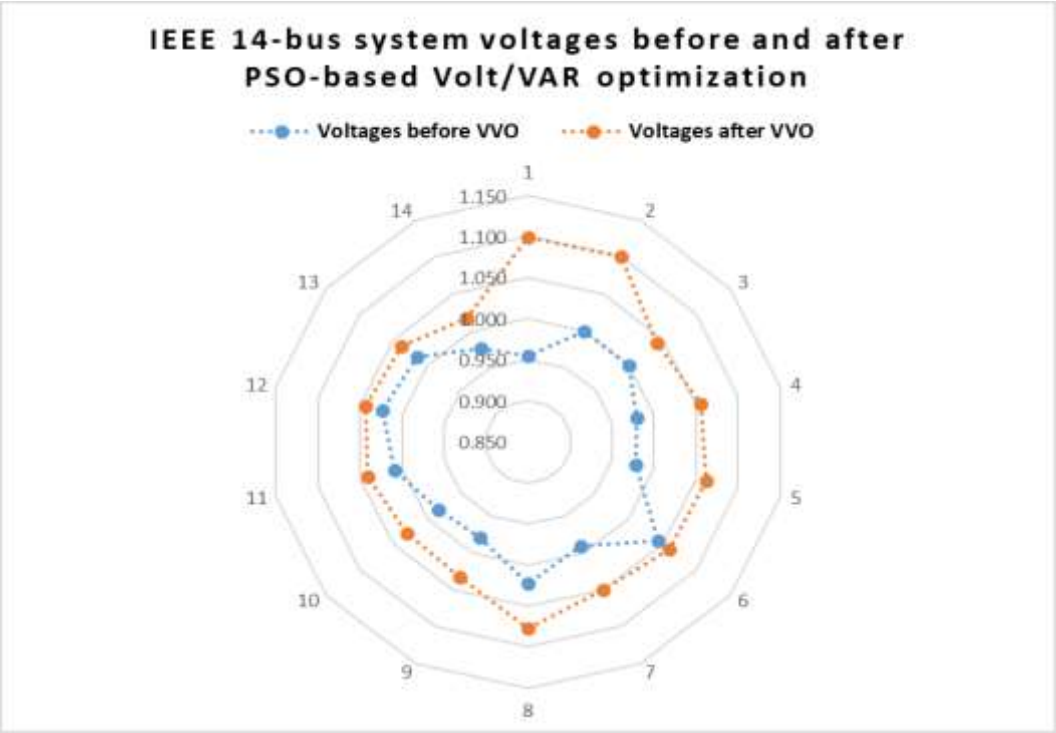

Fig. 6.13: Radar chart of 14-bus system voltage profiles before and after PSO-based VVO

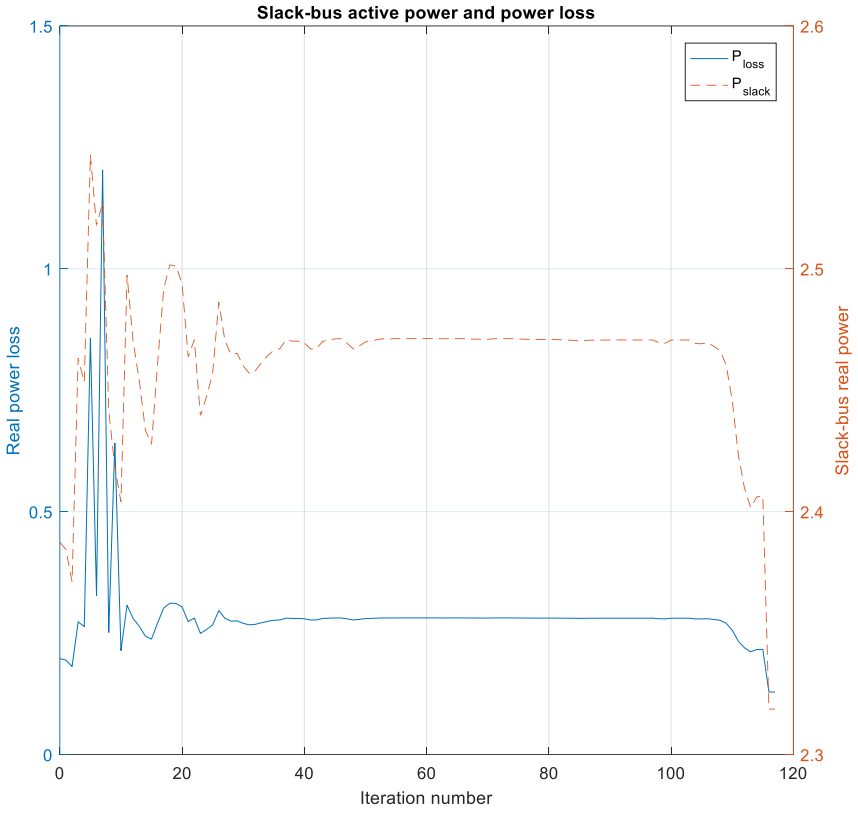

Fig. 6.14: 14-bus system real power loss and slack-bus active power plotted against number of iterations

Figure (6.14) depicts the real power loss trajectory plotted together with the slack-bus active power output against the number of iterations. Similar to the two previous case studies, the close relationship between the change in the two quantities is clearly noticeable, and the displacement between them can be attributed to the difference in the scale of the two quantities. Figure (6.15) compares the trajectories of the real power loss and the total generated system reactive power, which also depicts a relationship between the two quantities similar to that observed in the case of the real power loss and the slack-bus active power.

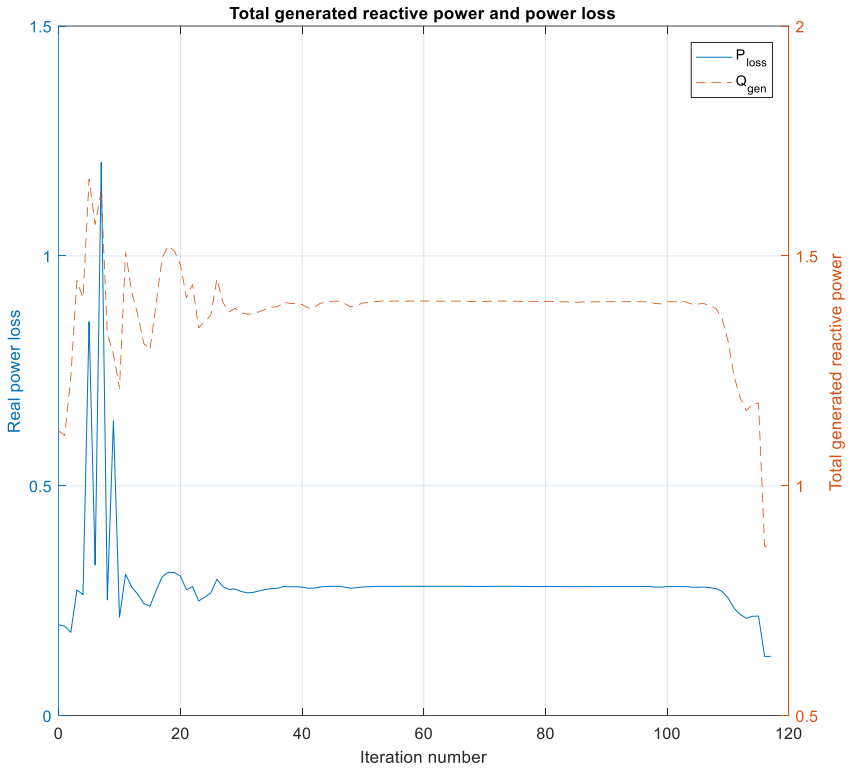

Fig. 6.15: 14-bus system real power loss and total generated reactive power plotted against number of iterations

6.5.5 Case study 4: 30-bus power system

Case study 4 is based on the IEEE 30-bus test system, the same as the one considered in section 5.3.4. It comprises 30 buses, 6 generators, 41 lines, and 21 loads. The network, load and generation data is also taken from Zhu (2009), and is the same as that used in chapter 5. Results of applying the PSO algorithm to the 30-bus system for solving the VVO problem are presented in Table (6.5).

Table 6.5: 30-bus system summary of PSO-VVO algorithm simulation results

Swarm size	Initial loss (p.u.)			Final loss (p.u.)			Number of iterations			Run time (sec)			Average % loss reduction
	Min	Max	Average	Min	Max	Average	Min	Max	Average	Min	Max	Average	
10	0.1353	2.0046	0.5253	0.0925	0.1067	0.0979	23	108	58	0.4386	2.1174	1.3169	81.29
20	0.1180	1.1763	0.4763	0.0925	0.0993	0.0947	32	179	80	0.7819	4.7353	2.2789	80.12
30	0.1203	0.8751	0.3089	0.0925	0.1054	0.0953	40	164	84	0.7581	3.7345	2.0272	69.15
40	0.1053	0.5211	0.2628	0.0925	0.0980	0.0937	20	170	96	0.4842	3.2412	1.6484	64.34
50	0.1008	1.1291	0.3987	0.0925	0.1031	0.0953	3	200	67	0.0448	4.8044	1.5187	76.10

The results presented in Table (6.5) show that there is substantial real power loss reduction in all the cases, ranging from 64.34% (attained with a swarm size of 20) to 81.29% (attained with a swarm size of 10). The average number of iterations is quite high for all the cases, lying in the range between 58 and 96. The dispersion between the minimum and maximum number of iterations is also quite high, the minimum and maximum values being 3 and 200 respectively, both obtained with a swarm size of 50. The average execution time ranges from a low of 1.3169 to a high of 2.2789, obtained with swarm sizes of 10 and 20 respectively. Compared with the previous case studies, the 30-bus system requires significantly more iterations to converge, and the execution is correspondingly longer. As in the previous cases, increasing the swarm size does not seem to have a large impact on either the quality of the solution or the computational cost of the algorithm.

The convergence behaviour of the PSO algorithm for the 30-bus system is depicted in Figure (6.16), and shows the changes in the global best position and in the fitness value of the global best position, as well as the fitness value of the global best position. The case depicted in the figure takes converges relatively quickly, requiring about 12 iterations and an execution time of 0.19 seconds.

The voltage profile of the 30-bus system before and after the Volt/VAR optimization is depicted in Figure (6.17) in the form of a radar chart. The post-optimization voltage magnitudes are greater than the pre-optimization values for the majority of buses. Some voltages hit their lower or upper limits, but there is no voltage violation for any of the buses.

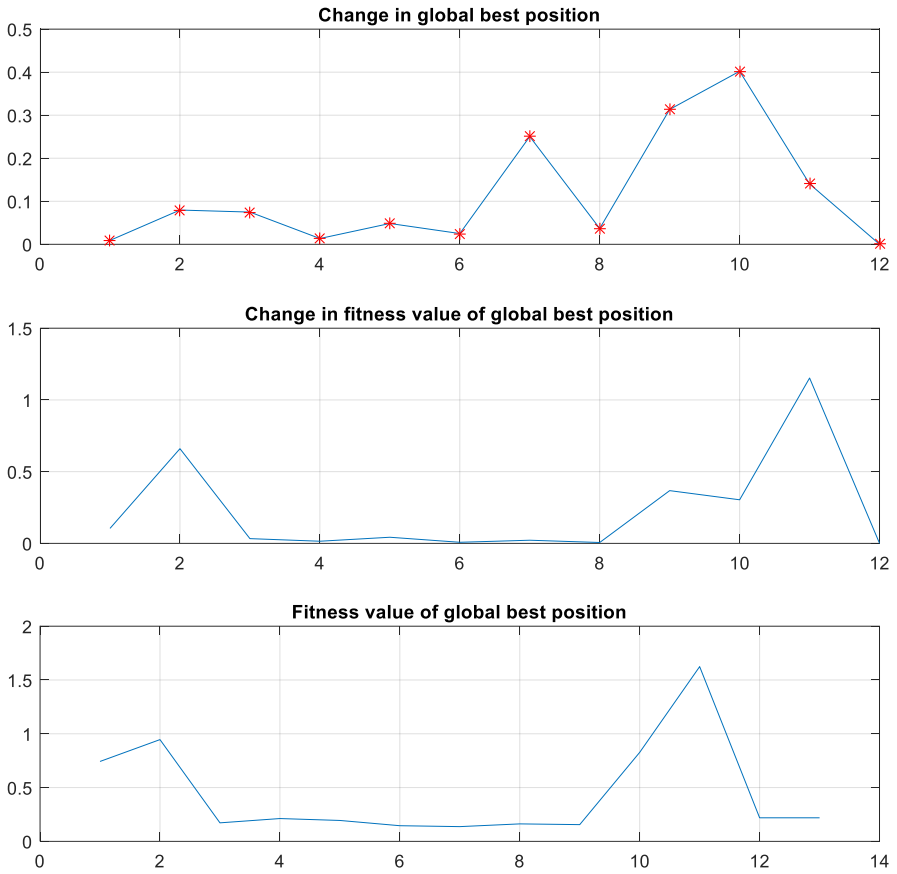

Fig. 6.16: 30-bus system convergence behaviour of the PSO algorithm

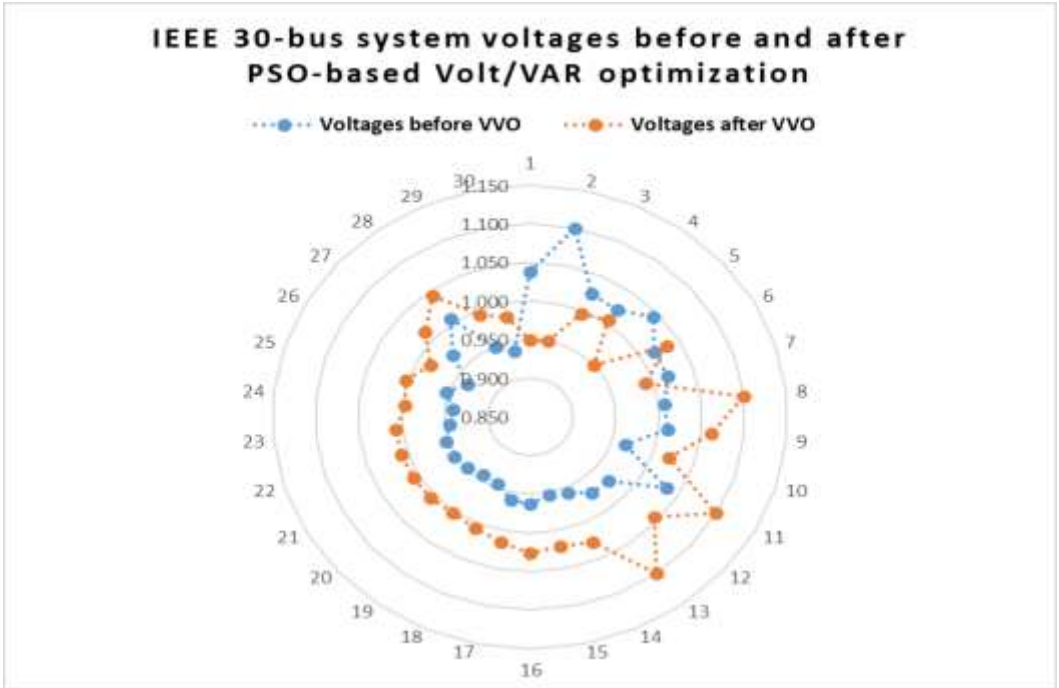

Fig. 6.17: Radar chart of 30-bus system voltage profiles before and after PSO-based VVO

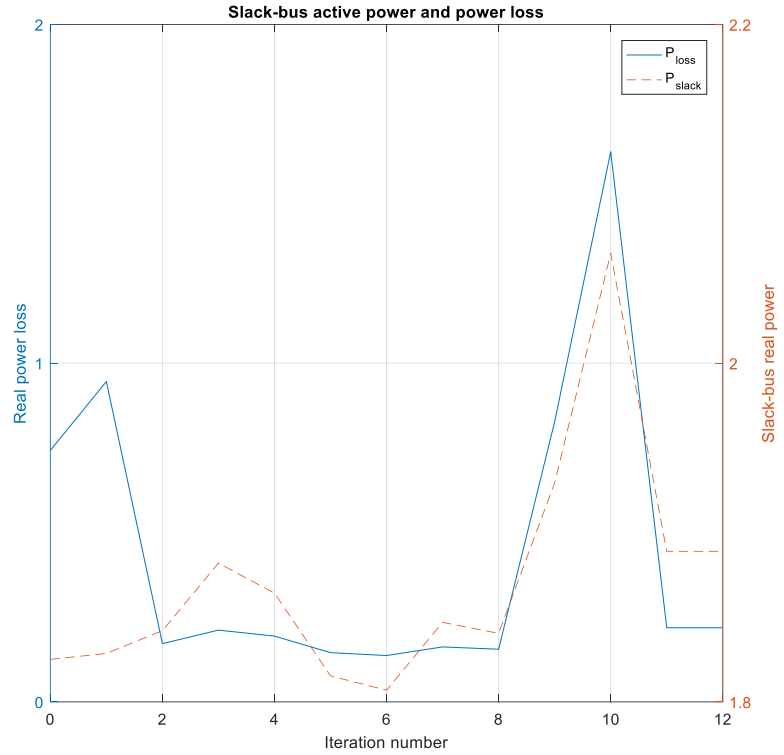

Fig. 6.18: 30-bus system real power loss and slack-bus active power plotted against number of iterations

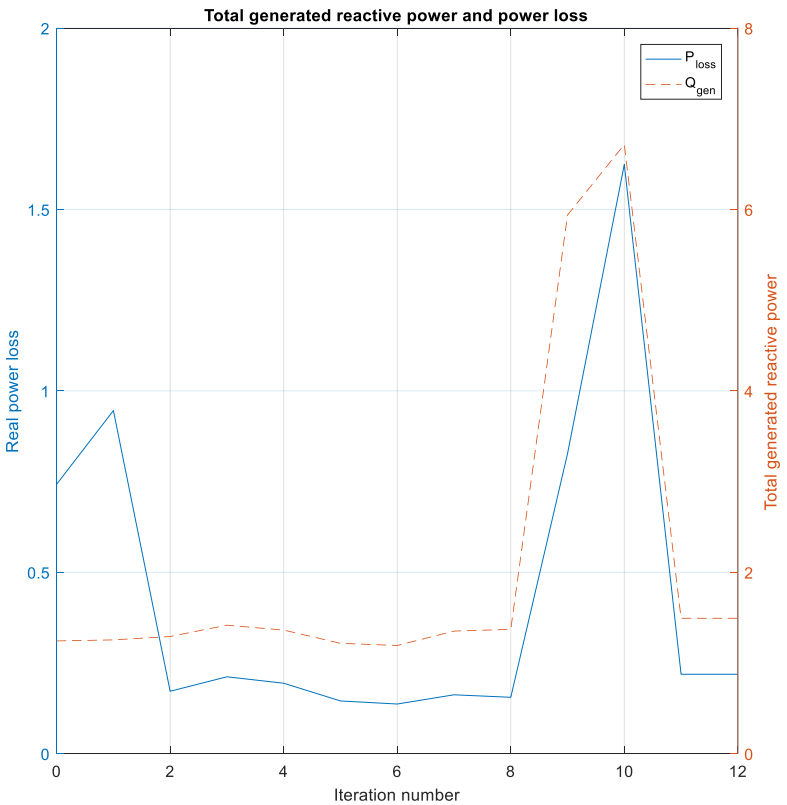

Fig. 6.19: 30-bus system real power loss and total generated reactive power plotted against number of iterations

Figure (6.18) plots the real power loss trajectory together with the slack-bus active power output against the number of iterations, and Figure (6.19) dose the same for

the real power loss and total generated system reactive power trajectories. Similar characteristics can be observed as those observed in the preceding case studies.

6.5.6 Case study 5: 118-bus power system

The final case study considered in this section is that of the IEEE 118-bus test system, the same as that considered in section 5.2.5. The system comprises 118 buses, 54 generators (35 of which are synchronous condensers), 186 lines, and 99 loads. The network, load and generation data is adapted from an appendix attributed to Springer Verlag (2012), and is the same as that used in chapter 5. Results of applying the PSO algorithm to the 118-bus system for solving the VVO problem are presented in Table (6.6), as well as in Figures (6.20) and (6.21)

Table 6.6: 118-bus system summary of PSO-VVO algorithm simulation results

Swarm size	Initial loss (p.u.)			Final loss (p.u.)			Number of iterations			Run time (sec)			Average % loss reduction
	Min	Max	Average	Min	Max	Average	Min	Max	Average	Min	Max	Average	
10	4.4286	6.7848	5.3158	2.3799	2.8516	2.6342	200	200	200	81.8725	82.7431	82.3317	50.45
20	5.0498	11.8646	7.9246	2.3778	3.0462	2.4514	111	200	183	45.7052	93.5918	77.8222	67.44
30	4.2654	11.0633	6.2787	2.3935	3.0599	2.5487	200	200	200	83.9581	88.0829	85.9590	59.41
40	4.4071	9.8267	5.9971	2.3822	2.5720	2.4510	200	200	200	82.2720	85.6926	84.2950	59.13
50	4.6573	11.1649	8.6517	2.3743	2.5027	2.4172	200	200	200	85.4243	86.8651	85.7809	72.06

The results for the 118-bus system presented in Table (6.6) show the average real power loss reduction among all the simulated cases to range from 50.45% (attained with a swarm size of 10) to 72.06% (attained with a swarm size of 50). The average number of iterations is quite high for all the cases. In fact, all cases except the case with a swarm size of 20 reach the pre-set maximum number of iterations. The average execution time is also quite high, especially when compared with the preceding case studies, and ranges from 77.8 seconds to 85.9 seconds. The minimum real power loss achieved is 2.37 per-unit. The results are quite consistent over the range of the swarm size, implying that the algorithm's performance is not significantly influenced by changing the swarm size, something that has been observed across all the preceding case studies as well.

The convergence behaviour of the PSO algorithm for the 118-bus system is displayed in Figure (6.20), which depicts the changes in the global best position in the top trace and the fitness value of the global best position in the bottom trace of the figure. It can be seen from the figure that the algorithm exhibits oscillatory behaviour beyond the 35^{th} iteration. This also seems to explain the results tabulated in Table (6.6), the oscillatory behaviour being the reason behind the algorithm exhausting the pre-set

maximum number of iterations. By adjusting higher the tolerance value of the termination condition (i.e. the change in the fitness value over successive iterations), the algorithm successfully terminates in much fewer iterations, as depicted in Figure (6.21).

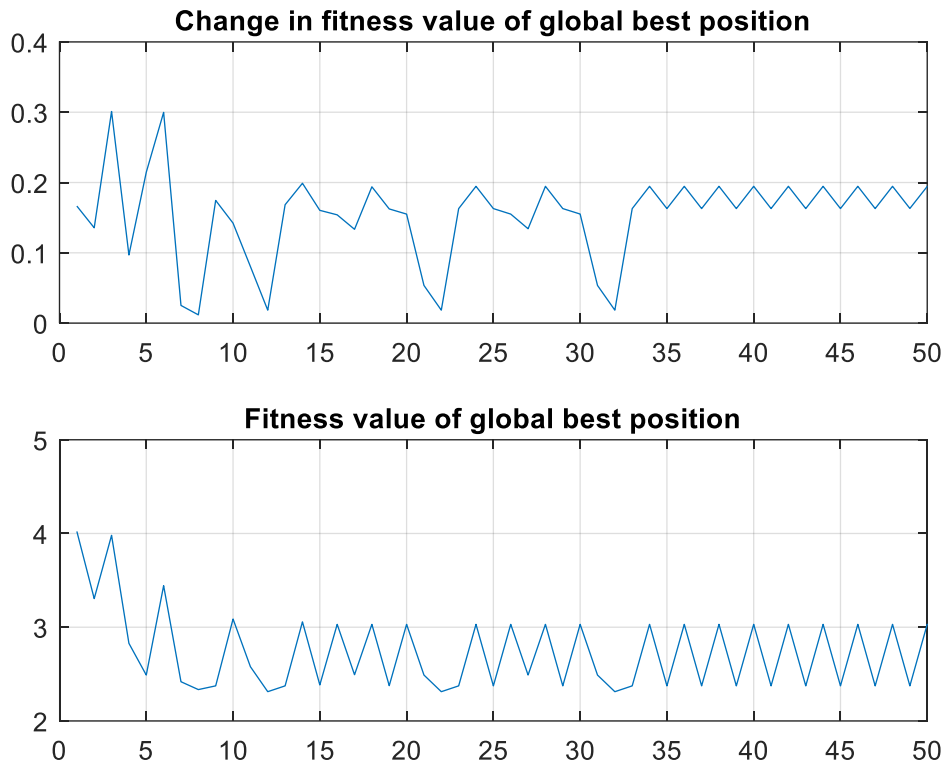

Fig. 6.20: 118-bus system convergence characteristics of the PSO algorithm, showing slightly oscillatory behaviour

The voltage profile of the 118-bus system before and after the Volt/VAR optimization is depicted in Figure (6.22) in the form of a radar chart. The pattern is similar to the preceding case studies, with the post-optimization voltage magnitudes being greater than the pre-optimization values for almost all the buses. Incidentally, the Volt/VAR optimization also alleviates some over-voltage conditions, particularly at buses 73 and 99, where the initial (i.e. pre-optimization) voltage magnitudes actually exceed the upper limit of 1.1 per-unit.

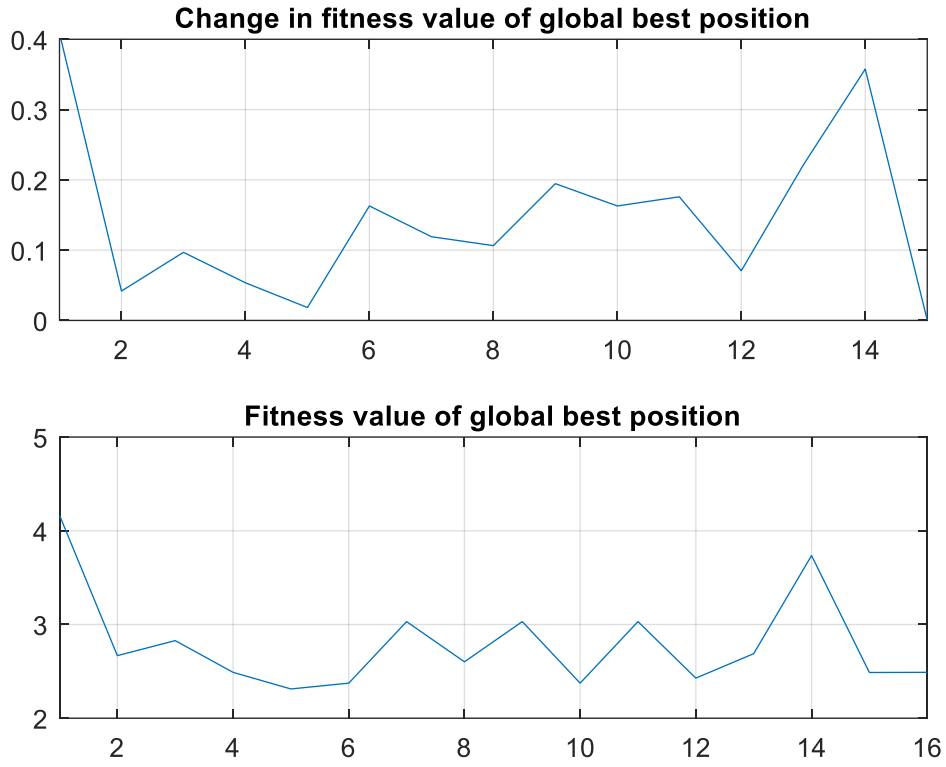

Fig. 6.21: 118-bus system convergence characteristics of the PSO algorithm after increasing termination condition tolerance of the change in the fitness value (top trace), showing successful convergence

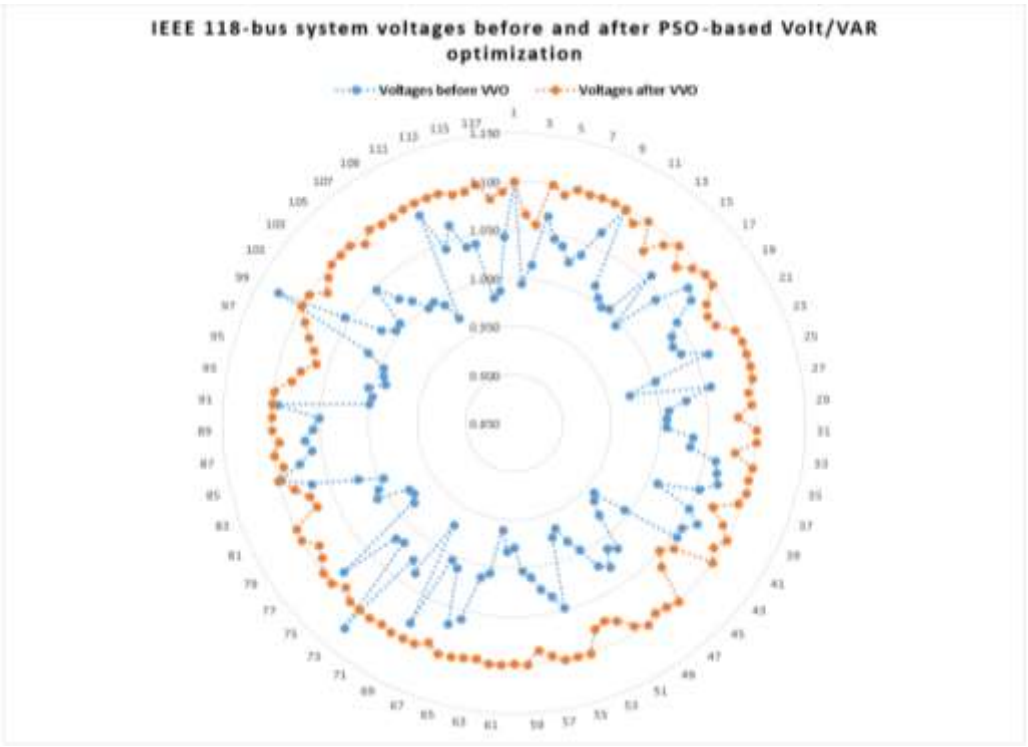

Fig. 6.22: radar chart of 118-bus system voltage profiles before and after PSO-based VVO

6.6 Comparison of PSO with PDIPM for VVO

It is interesting to compare how the two algorithms (i.e. PDIPM and PSO), developed in the previous chapter and in this chapter respectively, perform with respect to the Volt/VAR optimization problem. The comparison is made relatively easier by the fact that the same case studies have been conducted for both algorithms. The summary of the results is detailed in Table (6.7). For the PSO algorithm, since several simulations have been performed for each system case study, the "pareto-optimal" solution was selected for comparison with the PDIPM algorithm, considering the performance analysis criteria used in Table (6.7).

Table 6.7: Comparison of the PDIPM and PSO Volt/VAR optimization results

System	Initial loss (p.u.)		Final loss (p.u.)		Number of iterations		Run time (sec)		% Loss reduction	
	PDIPM	PSO	PDIPM	PSO	PDIPM	PSO	PDIPM	PSO	PDIPM	PSO
3-bus	0.0195	0.0180	0.0180	0.0156	4	11	0.0843	0.025	7.89	77.48
6-bus	0.1335	0.1378	0.1290	0.1265	13	54	0.1775	0.1060	3.37	41.62
14-bus	0.1353	0.1381	0.1296	0.1290	14	41	0.1477	0.2870	4.24	30.95
30-bus	0.1141	0.1353	0.1084	0.0925	14	58	0.3565	1.3170	5.03	80.29
118-bus	3.3939	4.6573	3.2270	2.3743	8	200	2.0120	85.78	4.92	72.06

In line with the performance analysis for the individual algorithms, the comparative performance analysis is made on the basis of the computational efficiency, in terms of the number of iterations taken by the algorithm to converge and the execution time required, as well as the percentage real power loss reduction. As can be deduced from the table, the PSO algorithm far outperforms the PDIPM algorithm in terms of the percentage loss reduction in all cases, as it consistently achieves much higher percentage loss reduction, and the final per-unit loss reduction is less for the PSO algorithm in all cases. The computational efficiency of the PSO algorithm, however, is generally worse than that of the PDIPM algorithm. It requires a much higher number of iterations in order to converge in all cases, although the execution time itself is lower for the 3-bus and 6-bus systems, but much higher for the 14-bus, 30-bus and 118-bus systems. This seems to suggest that the computational cost of the PSO algorithm tends to increase significantly as the problem dimension increases. The results of this study demonstrate what is generally known about the relative performance characteristics of classical and heuristic optimization techniques. The main strength of classical optimization methods is their high computational efficiency, particularly for differentiable nonlinear systems, but have the drawback of lacking the ability to find the globally optimal solution. Heuristic optimization techniques, on the other hand, generally incur high computational cost, but have the advantage that they have the capability to find the globally optimal solution.

6.7 Conclusion

The main result of this chapter is the development and implementation of the particle swarm optimization (PSO) algorithm for solving the Volt/VAR optimization (VVO) problem. A brief presentation on the historical development of the PSO algorithm is followed by a discussion of the principle of operation and basic construction of the algorithm, after which some pertinent implementation aspects are presented. The adaptation of the algorithm to the solution of the VVO problem is then outlined, before presenting five case studies by means of which the performance analysis of the developed PSO algorithm has been analysed. The performance analysis has focused on the algorithm's ability to effect real power loss reduction without adversely impacting the voltage profile of the system, and the computational efficiency, assessed in terms of the number of iterations required by the algorithm to converge, and the corresponding execution time. From the results analysis, the PSO performs very well in terms of real power loss reduction, although the computational cost is quite high, especially for systems with high problem dimensions (particularly the 118-bus system). The chapter ends with a comparative analysis of the PDIPM algorithm (developed and presented in chapter 5) and the PSO algorithm presented in this chapter, based on the same set of case studies. The comparative analysis reveals that the PDIPM algorithm is relatively more computationally efficient, whereas the PSO algorithm far outperforms in terms of real power loss reduction.

The following chapter concludes this thesis with a summary of the achievements of this research, as well as recommendations for further research.

CHAPTER SEVEN

CONCLUSION AND RECOMMENDATIONS FOR FUTURE RESEARCH

7.1 Introduction

Reliable electrical power supply is one of the most important utilities for modern society, so much that even momentary power supply interruptions can lead to enormous disruption of essential services and normal daily activities to the extent of being considered intolerable. Moreover, recent developments in the power system, such as the deregulation and restructuring of the electrical power supply industry, the introduction of competitive electricity and power markets, and the rapid growth of distributed and decentralized electrical power generation, have led to a significant increase in the complexity of modern power systems, adding to the challenge of operating them reliably and efficiently. Thus, the need for optimal strategies for the secure, economical and efficient operation of the power system is arguably even greater now than at any other time in the history of the power system. In line with this identified need, this thesis has presented the research conducted on the theoretical design, development, and practical implementation of efficient algorithms that contribute to the secure, economical and reliable operation of modern complex power systems. Particularly, algorithms have been developed for the solution of the Volt/VAR optimization problem, a very important sub-problem of the optimal power flow (OPF) problem that is mainly concerned with the optimal coordinated dispatch of voltage-regulating devices and reactive power sources.

The foundation for the work developed and presented in this thesis was laid by means of a thorough investigation of the state-of-the-art in problem formulation and solution techniques for the Volt/VAR optimization problem, considering both classical and heuristic optimization techniques. A critical comparative analysis of the classical and heuristic optimization techniques was then performed, to establish their individual characteristics, as well as their relative strengths and weaknesses. Some of the optimization performance criteria considered in the comparative analysis included the computational efficiency, convergence characteristics, and solution quality. This work has been presented in chapter 2 of this thesis.

The theoretical development and practical implementation of the classical primal-dual interior-point method for the solution of the Volt/VAR optimization problem has been detailed in chapters 4 and 5. Chapter 6 presents the design and implementation of the heuristic particle swarm optimization algorithm, also for the solution of the Volt/VAR optimization problem. The chapter also presents a comparative analysis of the performance of the two developed methods when applied to the Volt/VAR optimization problem. The performance analysis reveals that the primal-dual interior-

point method outperforms the particle swarm optimization algorithm in terms of computational efficiency, since on average it requires fewer iterations to converge, and has a shorter running time. The particle swarm optimization, on the other hand, generally achieves a higher percentage real power loss reduction than the primal-dual interior-point method. This suggests that the two classes of methods (i.e. classical and heuristic optimization methods) have complementary performance characteristics, something which could be exploited to devise optimization strategies that seek to combine their relative strengths, and thus have a better prospect exhibiting performance that is superior to that of the individual algorithms.

This chapter presents a concise summary of the main outcomes of the research presented in this thesis. The aim and objectives of the research have been outlined in section 1.4, and are reiterated in section 7.2, followed by the thesis deliverables in section 7.3. Section 7.4 lists possible applications of the results obtained from this research, and section 7.5 provides some recommendations for the direction that further research building on the work presented in this thesis could take. Section 7.6 lists the publications that have come out of this research.

The main purpose of sections 7.2 and 7.3 is to make a correlation between the aim and objectives stated in chapter 1 (reiterated in section 7.2 for convenience) and the thesis deliverables as reported in chapters 2 to 6.

7.2 Aim and objectives of the research

7.2.1 Aim

The main focus of the research presented in this thesis has been the theoretical design and practical implementation of efficient methods for the Volt/VAR optimization problem, in recognition of the central role played by Volt/VAR optimization in ensuring the security, economy, efficiency and reliability of operation of modern complex power systems. The research encompassed a thorough and comprehensive investigation of the state-of-the-art in the problem formulation and solution methods for the Volt/VAR optimization problem, as well as the theoretical development, design and practical implementation of the algorithms for the solution of the Volt/VAR optimization problem, based on both classical/conventional and heuristic/intelligent search-based optimization techniques. The objectives that supported the realization of the main aim of the research can be outlined as follows.

7.2.2 Objectives

1. Thorough investigation of the state-of-the-art in problem formulation and solution techniques for the Volt/VAR optimization problem, considering both classical and heuristic optimization techniques.

2. Critical comparative analysis of classical and heuristic optimization techniques, based on key optimization performance criteria, such as computational efficiency, convergence characteristics, and solution quality.
3. Model development for the Volt/VAR optimization problem, considering both the polar and rectangular coordinate representation of the system voltages.
4. Theoretical development of the primal-dual interior-point method (PDIPM) as the classical optimization technique applied to the solution of the Volt/VAR optimization problem.
5. Practical implementation of the PDIPM-based Volt/VAR optimization (PDIPM-VVO) algorithm, and a comprehensive performance analysis of the developed algorithm by means of a variety of power system case studies.
6. Theoretical development of the particle swarm optimization (PSO) algorithm as the heuristic optimization technique applied to the solution of the Volt/VAR optimization problem.
7. Practical implementation of the PSO-based Volt/VAR optimization (PSO-VVO) algorithm, and a comprehensive performance analysis of the developed algorithm by means of a variety of power system case studies.
8. Comparative analysis of the performance of the PDIPM and PSO algorithms as solution methods for the Volt/VAR optimization problem.

These objectives have been achieved, and are detailed in the preceding chapters of this thesis. In the following section, the deliverables of the research as outlined in section 1.9 are presented.

7.3 Thesis deliverables

7.3.1 Comprehensive literature study and review of the main aspects of Volt/VAR optimization

The comprehensive literature covered the problem formulation for the Volt/VAR optimization problem, encompassing the objectives, constraints and decision variables, as well as the main reactive power and voltage control devices most commonly used in Volt/VAR optimization. It also covered a thorough review and critical analysis of the main classical/conventional and heuristic/intelligent search-based optimization techniques that have been applied to the Volt/VAR optimization problem over the decades. A critical comparative analysis of these two classes of optimization methods was presented, highlighting their individual characteristics, as well as their relative strengths and weaknesses. This deliverable has been presented in chapter 2 of this thesis, as well in the publication (Mataifa *et al.*, 2022)

7.3.2 Problem formulation and model development for the Volt/VAR optimization problem

The problem formulation and model development for the Volt/VAR optimization problem involved the analysis of the objective functions, constraints and decision variables of the problem. Both the polar and rectangular coordinate representations of the system voltages were considered. The analysis performed revealed that the rectangular formulation had relatively more favourable mathematical properties from the perspective of computational efficiency, particularly for (although not limited to) the primal-dual interior-point algorithm, and was thus adopted in this study. This deliverable has been presented in chapter 3 of this thesis.

7.3.3 Theoretical development and design of the algorithms used in solving the Volt/VAR optimization problem

Three main algorithms were designed and developed for the solution of the Volt/VAR optimization problem: (1) the Newton-Raphson load flow algorithm in rectangular coordinates of system voltages, which is used as part of the Volt/VAR optimization algorithm purely for load flow computation; (2) the primal-dual interior-point method, which forms the basis for the design of the solution algorithm for the Volt/VAR optimization problem based on a classical optimization method; and (3) the particle swam optimization algorithm, which forms the basis for the design of the solution algorithm for the Volt/VAR optimization algorithm based on a heuristic optimization method. The development of the primal-dual interior-point method has been presented in chapters 4 and 5, the Newton-Raphson load flow algorithm has also been detailed in chapter 5, and the particle swarm optimization algorithm has been presented in chapter 6 of this thesis.

7.3.4 Software development for the implementation of the developed algorithms

The algorithms developed as detailed in section 7.3.3 (Newton-Raphson load flow algorithm, primal-dual interior-point method and particle swarm optimization algorithm) have been implemented in the MATLAB numerical and technical computing environment. The MATLAB programs for these algorithms are presented in Appendices A-C, and are listed in Table (8.1).

Table 7.1: Software programs developed and implemented in this thesis

Chapter 4	
Program description	**Appendix**
Function that implements the interior-point method (IPM) for a general nonlinear programming problem with inequality constraints	A.1 File name: *ipm_generic_nonlinear.m*

Function that defines the objective function, constraints, and the Jacobian and Hessian of the Lagrangian of the problem	A.2 File name: *func.m*
MATLAB script that calls the IPM to implement the example problem in section 4.3.8	A.3 File name: *ipm_generic_nonlinear_test.m*
Chapter 5	
Function that computes the residues of the load flow problem	B.1 File name: *dF.m*
Function that computes the Jacobian of the power flow equations for the load flow	B.2 File name: *jacobian.m*
Function that implements the Newton-Raphson load flow algorithm	B.3 File name: *NR_load_flow.m*
MATLAB script that runs the Newton-Raphson load flow computation for the 3-bus system	B.4 File name: three_bus_system_NR_load_flow.m
Function that implements the primal-dual interior-point method-based Volt/VAR optimization (PDIPM-VVO) for the 3-bus system, incorporating the Newton-Raphson load flow	B.5 File name: *ipm_3bus.m*
Function that computes the objective function, its gradient and Hessian for the 3-bus system	B.6 File name: *loss_func_3bus.m*
Function that computes the constraint functions, their Jacobian and Hessian for the 3-	B.7 *File name: h_gradh_hessh_3bus.m*
Function that computes the Jacobian and Hessian of the Lagrangian of the VVO problem for the 3-bus system	B.8 File name: *f_3bus.m*
MATLAB script that runs the PDIPM-VVO algorithm for the 3-bus system	B.9 File name: *three_bus_system_pdipm_vvo_test.m*
Function that implements the primal-dual interior-point method-based Volt/VAR optimization (PDIPM-VVO) for the 6-bus system, incorporating the Newton-Raphson load flow computation	B.10 File name: *ipm_6bus.m*
Function that computes the objective function, its gradient and Hessian for the 6-bus system	B.11 File name: *loss_func_6bus.m*
Function that computes the constraint functions, their Jacobian and Hessian for the 6-	B.12 *File name: h_gradh_hessh_6bus.m*
Function that computes the Jacobian and Hessian of the Lagrangian of the VVO problem for the 6-bus system	B.13 File name: *f_6bus.m*
MATLAB script that runs the PDIPM-VVO algorithm for the 6_bus system	B.14 File name: *six_bus_system_pdipm_vvo_test.m*

Function that implements the primal-dual interior-point method-based Volt/VAR optimization (PDIPM-VVO) for the 14-bus system, incorporating the Newton-Raphson load flow	B.15 File name: *ipm_14bus.m*
Function that computes the objective function, its gradient and Hessian for the 14-bus	B.16 File name: *loss_func_14bus.m*
Function that computes the constraint functions, their Jacobian and Hessian for the 14-bus system	B.17 *File name: h_gradh_hessh_14bus.m*
Function that computes the Jacobian and Hessian of the Lagrangian of the VVO problem for the 14-bus system	B.18 File name: *f_14bus.m*
MATLAB script that runs the PDIPM-VVO algorithm for the 14_bus system	B.19 File name: *fourteen_bus_system_pdipm_vvo_test.m*
Function that implements the primal-dual interior-point method-based Volt/VAR optimization (PDIPM-VVO) for the 30-bus system, incorporating the Newton-Raphson load flow	B.20 File name: *ipm_30bus.m*
Function that computes the objective function, its gradient and Hessian for the 30-bus	B.21 File name: *loss_func_30bus.m*
Function that computes the constraint functions, their Jacobian and Hessian for the 30-bus system	B.22 *File name: h_gradh_hessh_30bus.m*
Function that computes the Jacobian and Hessian of the Lagrangian of the VVO problem for the 30-bus system	B.23 File name: *f_30bus.m*
MATLAB script that runs the PDIPM-VVO algorithm for the 30-bus system	B.24 File name: *thirty_bus_system_pdipm_vvo_test.m*
Function that implements the primal-dual interior-point method-based Volt/VAR optimization (PDIPM-VVO) for the 118-bus system, incorporating the Newton-Raphson load flow	B.25 File name: *ipm_118bus.m*
Function that computes the gradient of the objective function for the 118-bus system	B.26 File name: *df_118bus.m*
Function that computes the Hessian of the objective function for the 118-bus system	B.27 File name: *d2f_118bus.m*
Function that defines the constraint functions for the 118-bus system	B.28 File name: *h_118bus.m*
Function that computes the Jacobian of the constraint functions for the 118-bus system	B.29 File name: *dh_118bus.m*
Function that computes the Hessian of the constraint functions for the 118-bus system	B.30 File name: *d2ht_lami_118bus.m*
Function that computes the Jacobian and Hessian of the Lagrangian of the VVO problem for the 118-bus system	B.31 File name: *f_118bus.m*

MATLAB script that runs the PDIPM-VVO algorithm for the 118_bus system	B.32 File name: *one_hundred_eighteen_bus_system_pdipm_vvo_test.m*
Chapter 6	
MATLAB script that runs the PSO-VVO algorithm for the 3-bus system	C.1 File name: *pso_vvo_3bus_system.m*
MATLAB script that runs the PSO-VVO algorithm for the 6-bus system	C.2 File name: *pso_vvo_6bus_system.m*
MATLAB script that runs the PSO-VVO algorithm for the 14-bus system	C.3 *File name: pso_vvo_14bus_system.m*
MATLAB script that runs the PSO-VVO algorithm for the 30-bus system	C.4 File name: *pso_vvo_30bus_system.m*
MATLAB script that runs the PSO-VVO algorithm for the 118-bus system	C.5 File name: *pso_vvo_118bus_system.m*
Function that computes the personal and global best positions for the PSO algorithm; applies to all case studies	C.6 File name: *PSO_compute_pbest_gbest.m*
Function that computes the fitness value of an individual particle for the PSO algorithm; applies to all case studies	C.7 File name: *PSO_objective_evaluation.m*
Function that computes the velocity update and adjusts the particle position for the PSO algorithm; applies to all case	C.8 File name: *PSO_X_update_cc1.m*
Utility functions	
Function to compute the impedance (Y) matrix for an arbitrary power system	D.1 File name: *compute_Ybus.m*
Function to compute the generator active and reactive power outputs once the load flow computation has converged	D.2 File name: *compute_PQ.m*
Function to define the initial guess for the Newton-Raphson load flow computation	D.3 File name: *define_x0.m*
Function to define the generator voltage magnitude set-points for the Newton-Raphson load flow computation	D.4 File name: *Vgref_0.m*
Function to compute the real power loss magnitude for an arbitrary power system	D.5 File name: *loss_func.m*

7.3.5 Comprehensive performance analysis of the developed algorithms by means of a variety of power system case studies

The performance of the developed PDIPM-VVO and PSO-VVO algorithms has been analysed by means of five different power system case studies, ranging in size from a 3-bus system to a 118-bus system. The main performance analysis criteria considered were the solution quality, in terms of the magnitude of real power loss percentage reduction and the voltage profile improvement achieved; the computational efficiency of the algorithm in terms of the number of iterations and corresponding runtime required by the algorithm to converge; the scalability of the developed algorithms, analysed in terms of the increase in number of iterations and running time as the system size increases, considering power system case studies ranging in size from 3-bus to 118-bus system; for the PSO-VVO algorithm, the impact of the swarm size on the solution quality and the computational cost of the PSO algorithm. The results of these analyses have been presented in chapters 5 and 6 of this thesis. Chapter 6 also presents a comparative analysis of the PDIPM-VVO and PSO-VVO algorithms.

7.4 Possible applications of the research outputs

The methods, algorithms and software programs developed in this thesis can find application in industry as well as in academia, for example:

- Voltage profile improvement in industrial electrical networks.
- Reactive power optimization and voltage control in transmission networks, as part of the energy management system.
- Power system simulators used to train operators, to facilitate the study of the impact of reactive power optimization on system voltages and on overall power and energy management.
- Postgraduate research programs focused on the teaching of fundamental concepts in classical and heuristic optimization, and their practical implementation in software development environments such as MATLAB.

7.5 Recommendations for future research

Possible directions of further research that builds on the research presented in this thesis can be:

- Testing of the developed algorithms in a real-time simulation environment, with MATLAB as the computational engine for the optimization, and the Real Time Digital Simulator (RTDS) as the real-time environment for the modelling of the power system. Consideration of the robustness of the developed algorithm to parameter uncertainties.

- Extension of the developed algorithms and software programs to incorporate parallel programming and parallel processing, to enable the handling of larger systems in a computationally more efficient way, and to improve computational resource utilization.
- Consideration of other objectives besides real power loss minimization and voltage profile improvement, such as voltage stability maximization, and reactive power reserve margin maximization, as well the impact of generation/demand variation on the effectiveness of the developed algorithms.
- Consideration of other optimization algorithms, such as genetic algorithm, evolutionary programming, and simulated annealing, and comparing the efficiency and effectiveness of the various algorithms when applied to the Volt/VAR optimization problem.
- Consideration of newer forms of power system resources, such as FACTS devices and various distributed energy resources using type 4 wind turbines and inverter based systems which are capable of providing reactive power support in the framework of Volt/VAR optimization, and additional controls, such as under-load transformer tap changer.

7.6 Publications

1. Mataifa, H., Krishnamurthy, S. & Kriger, C., "Volt/VAR Optimization: A Survey of Classical and Heuristic Optimization Methods," *IEEE Access, vol. 10*, 2022.

2. Mataifa, H., Krishnamurthy, S. & Kriger, C., "An Efficient Primal-Dual Interior-Point Algorithm for Volt/VAR Optimization in Rectangular Coordinates," (under review, submitted to *IEEE Access*, November 2022).

3. Mataifa, H., Krishnamurthy, S. & Kriger, C., "Comparative Analysis of the Primal-Dual Interior-Point Method and Particle Swarm Optimization Algorithms for the Solution of the Volt/VAR Optimization Problem," (to be submitted to *Electric Power Systems Research*, December 2022).

7.7 Conclusion

This chapter has concluded the thesis by presenting a concise summary of the most important outcomes of the research. Section 7.2 has outlined the research aim and objectives, section 7.3 has presented a detailed description of the thesis deliverables, making the linkage clear between the work presented in the thesis chapters and their alignment to the main research outputs and deliverables. Possible applications of the research presented in this thesis are covered in section 7.4, and section 7.5 makes some recommendations for further research that builds on the work presented in this thesis. Section 7.6 has listed the publications coming out of this research.

BIBLIOGRAPHY

Abdul-Rahman, K.H. & Shahidehpour, S.M., “A fuzzy-based optimal reactive power control,” IEEE Trans. on Power Syst., vol. 8, no. 2, May 1993.

Abido M.A. & Bakhashwain, J.M., “A novel multiobjective evolutionary algorithm for optimal reactive power dispatch problem,” Proc. of 10th IEEE Intern. Conf. on Electronic Circuits and Syst., ICECS, vol. 3, pp. 1054-1057, 2003.

Ahmadi, H., Marti, J.R. & Dommel, H.W., “A framework for volt/var optimization in distribution systems,” IEEE Trans. Smart Grid, vol 6, no. 3, May 2015.

Aldahmashi, J. & Ma, X., “Advanced machine learning approach of power flow optimization in community microgrid,” *Proceedings of the 27th International Conference on Automation & Computing*, University of the West of Englad, Bristol, UK, 1-3 September 2022.

Alghamdi, A.S., “Optimal power flow of renewable-integrated power systems using a Gaussian bare-bones levy-flight firefly algorithm,” *Frontiers in Energy Research*, vol. 10, May 2022.

Alsac, O., Bright, J., Paris M., & Stott, B., “Further developments in LP-based optimal power flow,” IEEE Trans. Power Syst., vol. 5, no. 3, Aug. 1990, pp. 697-711.

Alves da Silva, A.P. & Falcao, D.M., “Fundamentals of genetic algorithms,” in K.Y. Lee and M.A. El-Sharkawi (eds.), Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems, John Wiley & Sons Inc., 2008.

Aucharimayet, S. & Sirisumrannukul, S., “Volt/VAr control in distribution systems by fuzzy multiobjective and particle swarm,” 6th Intern. Conf. on Electr. Eng./Electronics, Computer, Telecom. & Info. Tech., pp. 234-237, 2009.

Bazaraa, M., Sherali H. and Shetty, C., “Nonlinear programming: theory and algorithms,” Wiley New York, 2006.

Bekhouche, N., “Automatic generation control before and after deregulation,” Proceedings of the Thirty-Fourth Southeastern Symp. on Syst. Theory, Huntsville, AL., USA, 2002, pp. 321-323.

Bhatele, R.P., Sharma, J.D. & Thapar, O.D., “Optimal reactive power control via loss minimization and voltage control,” Electric Power & Energy Syst., vol. 7, no. 4, Oct. 1985.

Billinton, R. & Sachdeva, S.S., “Real and reactive power optimization by suboptimal techniques,” IEEE Trans. Power Appar. & Syst., PAS-92, No. 3, pp. 950-956, May 1973.

Bjelogrlic, M., Calovic, M.S. & Babic, B.S., “Application of Newton’s optimal power flow in voltage/reactive power control,” IEEE Trans. Power Syst., vol. 5, no. 4, Nov. 1990.

Boyd, R. & Richerson, P.J., “Culture and the evolutionary process,” Univ. of Chicago Press, Chicago, 1985.

Braun, M., “Technological control capabilities of DER to provide future ancillary services,” Intern. Journal of Distrib. Energy Res., vol. 3, no. 3, pp. 191-206., 2007.

Brown, C.E. & O’Leary, D.E., “Introduction to artificial intelligence and expert systems,” Intern. Journal of Intelligent Syst., 1995.

Cai, G., Ren, Z. & Yu, T., “Optimal reactive power dispatch based on modified particle swarm optimization considering voltage stability,” IEEE Power Eng. Society Gen. Meet., pp. 1-5, 2007.

Cain, M.B., O’Neill, P.R. & Castillo, A., “History of optimal power flow and formulations (OPF Paper 1),” Tech. Rep., US FERC, December 2012.

Capitanescu, F., Glavic, M. & Wehenkel, L., “Interior-point based algorithms for the solution of optimal power flow problems,” Electr. Power Syst. Res., 77, pp. 508-517, 2007.

Capitanescu, F., Glavic, M. & Wehenkel, L., 2005, “An interior-point based optimal power flow,” *3rd*

ACOMEN Conference, Gand, Belgium, June 2005.

Carpentier, J., “Contribution a l’etude du dispatching economique,” Bulletin de la Societe Francaise des Electriciens, vol. 3, ser. 8, pp. 431-447, 1962.

Cha, M., Yazdani, A., Harkness, J., Melendrez, D., Vang, N. & Lopez, J., “Smart Inverters and Volt Var Optimization,” 2020 52nd North American Power Symposium (NAPS), 2020.

Chebbo, A.M., “Security constrained reactive power dispatch in electrical power systems,” Ph.D. dissertation, Sch. of Eng. & Appl. Science, Durham Univ., Durham, UK, 1990.

Chebbo, A.M., Irving M.R. & Sterling, M.J.H., “Reactive power dispatch incorporating voltage stability,” IEE Proceedings-C, vol. 139, no. 3, May 1992.

Cheng, S.J., Malik, O.P. & Hope, G.S., “An expert system for voltage and reactive power control of a power system,” IEEE Trans. on Power Syst., vol. 3, no. 4, Nov. 1988.

Choden, K., Sonam, C., Tenzin, S., Karchung, J. & Seldon, Y., “Enhancement of voltage stability in the power system using genetic algorithm,” *Journal of Applied Engineering, Technology and Management* (JAETM), vol. II, iss. I, June 2022.

Clausen, J., “Branch and bound algorithms-principles and examples,” Tech. Rep., Univ. Copenhagen, 1999.

Clerc, M. & Kennedy, J., “The particle swarm – explosion, stability and convergence in a multidimensional complex space,” *IEEE Transactions on Evolutionary Computation, vol. 6, no. 1*, pp. 58-73, 2002.

Coello, C.A.C., Pullido, G.T. & Lechuga, M.S., “Handling multiple objectives with particle swarm optimization,” IEEE Trans. on Evolutionary Computation, vol. 8, no. 3, pp. 256-279, June 2004.

Colorni, A., Dorigo M. & Maniezzo, V., “Distributed optimization by ant colonies,” *Proc. of First Euro. Conf. on Artificial Life*, pp. 134-142, MIT Press, Cambridge, 1991.

Csany, E., 2014, “An example of transformer tap-changer correct adjustment,” [online]. Available at: https://electrical-engineering-portal.com/example-of-transformer-tap-changer-correct-adjustment. Accessed June 13, 2022.

da Costa, G.R.M., “Optimal reactive dispatch through primal-dual method,” IEEE Trans. Power Syst., vol. 12, no. 2, May 1997.

de Carvalho, E., dos Santos, A. & Mac, T., “Reduced gradient method combined with augmented Lagrangian and barrier for the optimal power flow problem,” Appl. Math. Comput. vol. 200, pp. 529-536, 2008.

De Jong, K., Fogel, D.B. & Schwefel, H.-P., “A history of evolutionary computation,” in T. Baeck, D.B. Fogel and Z. Michalewicz (eds.), Handbook of Evolutionary Computation, CRC Press, 1997.

De M. & Goswami, S.K., “Optimal reactive power procurement with voltage stability consideration in deregulated power system,” IEEE Trans. Power Syst. vol. 29, no. 5, Sept. 2014.

de Sousa, V.A., Nunes, L.C.T., Belati, E.A. & da Costa, G.R.M., “A new approach type-Newton for optimal reactive power dispatch,” IEEE Power Eng. Soc. Gen. Meet., vol. 1, pp. 252-257, 2003.

de Souza, B.A. & de Almeida, A.M.F., “Multiobjective optimization and fuzzy logic applied to planning of the volt/var problem in distributions systems,” IEEE Trans. Power Syst., vol. 25, no. 3, Aug. 2010.

Deb, K., “Multi-objective optimization using evolutionary algorithms, John Wiley & Sons, Ltd, Chichester, England, 2001.

Deeb N., & Shahidehpour, S.M., “Linear reactive power optimization in a large power network using the decomposition approach,” IEEE Trans. Power Syst., vol. 5, no. 2, May 1990.

Ding, F., Zhang, Y., Simpson, J., Bernstein A. & Vadari, S., “Optimal Energy Dispatch of Distributed PVs for the Next Generation of Distribution Management Systems,” IEEE Open Access Journal of Power and Energy, Aug. 2020.

Ding, Q., Li, N. & Wang, X., “Implementation of Interior Point Method Based on Voltage/Reactive Power Optimization,” IEEE Power Eng. Society Winter Meeting, Conf. Proc., vol. 12, pp. 1197-1201, 2000.

Dommel, H.W. & Tinney, W.F., “Optimal power flow solutions, IEEE Trans. Power Appar. & Syst., Oct. 1968.

Dulau, L-L. & Bica, D., “Optimal power flow analysis of a power system with distributed generators and storage considering seasons,” Tehnicki vjesnik 29, 6 (2022), 1819-1826.

Eberhart, R.C. & Shi, Y., “Comparing inertia weights and constriction factors in particle swarm optimization,” Proceedings of the IEEE Conference on Evolutionary Computation (CEC), pp. 84-88, San Diego, CA., Piscataway

EPRI, “Optimization of Reactive Volt-Ampere (VAR) Sources in System Planning: Volume 1, Solution Techniques, Computing Methods, and Results,” EPRI, Scientific Systems, Inc., Cambridge, Massachusetts, Nov. 1984.

Exposito, A.G., Ramos, J.L.M., Macias, J.L.R. & Salinas, Y.C., “Sensitivity-based reactive power control for voltage profile improvement,” IEEE Trans. on Power Syst., vol. 8, no. 3, Aug. 1993.

Feng X., & Peterson, W., “Volt/VAR optimization reduces losses, peak demands,” EET&D Magazine, vol. 14, no. 1, pp. 22-25, Feb. 2010.

FERC, “Reactive Power Supply and Consumption,” Federal Energy Regulatory Commision Staff Report No. AD05-1-000, Feb. 4, 2005.

Fernandes, R.A., Happ H.H. & Wirgau, K.A., “Optimal reactive power flow for improved system operations,” Electric Power & Energy Syst., vol. 2, no. 3, July 1980.

Ferreira, E.C., Neto, M.S.I. & Asada, E.N., “Metaheuristic strategies for solving the optimal reactive power dispatch with discrete variables,” 12th IEEE Intern. Conf. on Industr. Appl., INDUSCON, pp. 1-6, 2016.

Fletcher R. & Reeves, C.M., “Function minimization by conjugate gradients,” Computer Journal, iss. 7, pp. 149-154, 1964.

Forsgren, A., Gill P.E. & Wright, M.H., “Interior Methods for Nonlinear Optimization,” SIAM Review, vol. 44, no. 4, pp. 525-597, 2002.

Frank, S. Rebennack, S., “An introduction to optimal power flow: theory, formulation, and examples,” IIE Trans., vol. 48, no. 12, pp. 1172-1197, August 2016.

Frank, S., Steponavice, I. & Rebennack, S., “Optimal power flow: a bibliographic survey I, formulations and deterministic methods,” Energy Systems, vol. 3, pp. 221-258, 2012a.

Frank, S., Steponavice, I. & Rebennack, S., “Optimal power flow: a bibliographic survey II, non-determistic and hybrid methods”, *Energy Systems,* vol. 3, pp. 259-289, 2012b.

Freitas, D., Lopes, L.G. & Morgado-Dias, F., “Particle Swarm Optimisation: A Historical Review Up to the Current Developments,” *Entropy* 2020, *22*, 362.

Fukuyama, Y., “Fundamentals of Particle Swarm Optimization,” in K.Y. Lee and M.A. El-Sharkawi (eds.), Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems, John Wiley & Sons Inc., 2008.

Gad, A.G., “Particle swarm optimization algorithm and its applications: A systematic review,” *Archives of Computational Methods in Engineering* (2022) 29:2531-2561.

Gandoman, F.H., Ahmadi, A., Sharaf, A.M., Siano, P., Pou, J., Hredzak, B. & Agelidis, V.G., “Review of FACTS technologies and applications for power quality in smart grids with renewable energy systems,” Renewable and Sustainable Energy Reviews 82 (2018) 502-514.

Geoffrion, A.M., “Generalized Benders decomposition,” Journal Optim. Theory & Appl., vol. 10, no. 4, 1972.

Ghodrati, M., Piri, M., & Sadr, S.M., “Probabilistic Multi-Objective Reactive Power Planning Considering

Large-Scale Wind Integration," 34th International Power System Conference (PSC2019), 9-11 Dec., Tehran, Iran, 2019.

Glover, F. & Laguna, M., "Tabu search," Kluwer Academic Press, Boston, 1997.

Goldberg, D.E., "Genetic algorithms in search, optimization and machine learning," Addison-Wesley Publishing Company, Inc., 1989.

Golkar, M.A. & Rajabzadeh, M., "Optimum allocation of reactive power in real-time operation under de-regulated electricity market," 2009 Australasian Univ. Power Eng. Conf., Adelaide, SA., 2009, pp. 1-6.

Gomes J.R. & Saavedra, O.R., "Optimal reactive power dispatch using evolutionary computation: extended algorithms," IEE Pro.-Gener. Transm. Distrib., vol. 146, no. 6, Nov. 1999.

Gonzalez-Longatt, F.M., 2014, "IEEE 30 bus Test," [online]. Available at: https://www.fglongatt.org/Test_Systems/IEEE_30bus.html. Accessed August 30, 2022.

Gonzalez-Longatt, F.M., 2015, "IEEE 14 bus Test," [online]. Available at: https://www.fglongatt.org/Test_Systems/IEEE_14bus.html. Accessed August 30, 2022.

Grant, L., Venayagamoorthy, G.K., Krost, G. & Bakare, G.A., "Swarm intelligence and evolutionary approaches for reactive power and voltage control," IEEE Swarm Intelligence Symposium, St. Louis MO, USA, Sept. 21-23 2008.

Granville, S., "Optimal Reactive Dispatch Through Interior Point Methods," IEEE Trans. Power Syst., vol. 9, no. 1, Feb. 1994.

Grudinin, N., "Reactive Power Optimization Using Successive Quadratic Programming Method," IEEE Trans. Power Syst., vol. 13, no. 4, Nov. 1998.

Gupta, Y., Chatterjee, K. & Doolla, S., "Volt-Var optimization and reconfiguration: reducing power demand and losses in a droop-based microgrid," IEEE Trans. on Industry Applications, vol. 57, no. 3, May/June 2021.

Haida, T. & Akimoto, Y., "Genetic algorithms approach to voltage optimization," Proc. of First Intern. Forum on Appl. of Neural Networks to Power Syst., pp. 139-143, 1991.

Hano, I., Tamura, Y., Narita, S. & Matsumoto, K., "Real time control of system voltage and reactive power," IEEE Trans. Power Appar. & Syst., pp. 1544-1559, Oct. 1969.

Happ, H.H., "Optimal power dispatch: a comprehensive survey," IEEE Trans. Power Appar. and Syst., PAS-90, May 1977, pp. 841-854.

Hemmecke, R., Koeppe, M., Lee, J. & Weismantel, R., "Nonlinear integer programming," in 50 Years of Integer Programming 1958-2008: The Early Years and State-of-the-Art Surveys, Springer-Verlag, 2009.

Heppner, F. & Grenander, U., "A stochastic nonlinear model for coordinated bird flocks," In Krasner, S., Ed., "The ubiquity of chaos," AAAS Publications, Washington D.C., 1990.

Hestenes M.R. & Stiefel, E., "Methods of conjugate gradients for solving linear systems," Journal Res. National Bureau of Standards, vol. 49, pp. 409-436, 1952.

Hobson, E., "Network Constrained Reactive Power Control Using Linear Programming," IEEE Trans. Power Appar. & Syst., vol. PAS-99, no. 3, 1980.

Holland, J.H., "Adaptation in natural and artificial systems," ANN Harbor, MI, Michigan Univ. Press, 1975.

Hongji, C., Daobing, L., Shichun, L., Zitong, J., Ye, Y. & Songlin, X., "Optimal power flow calculation of active distribution network based on improved comprehensive technology," *Wuhan University Journal of Natural Sciences*, vol. 27, no. 2, pp. 142-152, 2022.

Hu, X., Shi, Y. & Eberhart, R., "Recent advances in particle swarm," *Proceedings of the 2004 Congress on Evolutionary Computation*, vol. 1, pp. 90-97.

Huneault M & Galiana, F.D., "A survey of the optimal power flow literature," IEEE Trans. Power Syst., vol.

6, 1991, pp. 762-770.

Iba, K., “Reactive power optimization by genetic algorithm,” IEEE Trans. on Power Syst., vol. 9, no. 2, May 1994.

Iyer, R.S., Ramachandran, K. & Hariharan , S., “New technique for optimal reactive-power allocation for loss minimization in power systems,” IEE Proceedings, vol.130, no.4, July 1983

Jenkins, N., Ekanayake, J.B. & Strbac, G. 2010. “Distributed Generation”, IET, London.

Jha R.R., & Dubey, A., “Coordinated voltage control for conservation voltage reduction in power distribution systems,” IEEE Power & Energy Society General Meeting (PESGM), 2020.

Ji, Y., Liu, K., Geng, G., Sheng, W., Meng, X., Jia D. & He, K., “A novel reactive power optimization in distribution network based on typical scenarios partitioning and load distribution matching method,” MDPI, Appl. Sciences, Aug. 2017

Jin, D., Chiang, H.-D. & Li, P., “Two-timescale multi-objective coordinated volt/var optimization for active distribution networks,” IEEE Trans. Power Syst., vol. 34, no. 6, Nov. 2019.

Joos, G., Ooi, B.T., McGillis, D., Galiana, F.D. & Marceau, R., “The potential of distributed generation to provide ancillary services,” Power Eng. Society Summer Meeting, Seattle, WA, 2000, vol. 3, pp. 1762-1767.

Karmakar, N., Raj, S. & Bhattacharyya, B., “Hybrid Intelligence Technique for Reactive Power Planning using FACTS devices,” 2020 International Conference on Emerging Frontiers in Electrical and Electronic Technologies (ICEFEET), pp. 1-6, 2020.

Karmarkar, N., “A new polynomial time algorithm for linear programming,” Combinatorica, iss. 4, pp. 373-395, 1984.

Katuri, R., Jayalaxmi, A., Yesuratnam, G. & Yeddanapalli, D., “Genetic algorithm optimization of generator reactive power,” AASRI Conf. Power & Energy Syst., pp. 192-198, 2012.

Kaur, D., Lie, T.T., Nair, N.K.C. & Valles, B., “An optimal reactive power dispatch (ORPD) for voltage security using particle swarm optimization (PSO) in graph theory,” IEEE Intern. Conf. on Sustainable Energy Technologies (ICSET), pp. 25-30, 2016.

Kennedy, J. & Eberhart, R., “Particle swarm optimization,” Proc. of IEEE Intern. Conf. on Neural Networks (ICNN’95), vol. iv, pp. 1942-1948, Perth, Australia.

Kishore A. and Hill, E.F., “Static Optimization of Reactive Power Sources by use of Sensitivity Parameters, IEEE Transactions on Power Apparatus and Systems,” PAS-90, no. 3, pp. 1166-1173, May 1971.

Krishnamurthy, S. & Tzoneva, R., “Investigation of the methods for single area and multi area optimization of a power system dispatch problem,” Intern. Review of Electr. Eng. (IREE), vol. 7, no. 1, Jan.-Feb. 2012.

Kundur, P., “Power System Stability and Control,” McGraw-Hill, Inc., 1994.

Land A.H. & Doig, A., “An automatic method for solving discrete programming problems,” Econometrica, vol. 28, pp. 497-520, 1960.

Lee, K.Y., Bai, X. & Park, Y-M., “Optimization method for reactive power planning by using a modified simple genetic algorithm,” IEEE Trans. on Power Syst., vol. 10, no. 4, Nov. 1995.

Lewis, S.M., “Creating a smart power-delivery system,” Transm. & Distrib., vol. 52, no. 1, Jan. 2000, pp. 34-41.

Li A. & Zhong, J., “Market-Based Volt-Var Optimization and It’s Applications on Bottom-Up Load Modeling Method,” IEEE Transactions on Power Systems, Vol. 36, No. 3, May 2021.

Li, C. & Coster, D.C., “Improved particle swarm optimization algorithms for optimal designs with various decision criteria,” *Mathematics, MDPI*, 2022.

Li, J., Huang, H., Lou, B., Peng, Y., Huang, Q. & Xia, K., “Wind Farm Reactive Power and Voltage Control Strategy Based on Adaptive Discrete Binary Particle Swarm Optimization Algorithm,” 2019 IEEE Asia

Power and Energy Engineering Conference (APEEC), 2019, pp. 99-102.

Lin, M., Rayudu, R.K. & Samarasinghe, S., “A review of voltage/var control,” Australasian Univ. Power Eng. Conf. (AUPEC2003), Christchurch, New Zealand, October 2003.

Liu, C.-C. & Tomsovic, K., “An expert system assisting decision-making of reactive power/voltage control,” IEEE Trans. on Power Syst., vol. PWRS-1, no. 3, Aug. 1986.

Liu, H., Zhang, C., Chai, Q., Meng, K., Guo, G. & Yang Dong, Z., “Robust Regional Coordination of Inverter-Based Volt/Var Control via Multi-Agent Deep Reinforcement Learning,” DOI 10.1109/TSG.2021.3104139, IEEE Transactions on Smart Grid, 2021.

Lomi A. & Limpraptono, F.Y., “Implementation of expert system for power system voltage stability improvement,” Intern. Journal of Electr. Eng., vol. 5, no. 1, June 2017.

Lu, F.-C. & Hsu, Y.-Y., “Fuzzy dynamic programming approach to reactive power/voltage control in a distribution substation,” IEEE Trans. on Power Syst., vol. 12, no. 2, May 1997.

Mahapatra, S., Badi, M. & Raj, S., “Implementation of PSO, its variants and hybrid GWO-PSO for improving reactive power planning,” Global Conference for Advancement in Technology (GCAT), Bangalore, India, Oct. 18-20, 2019.

Mahapatra, S., Dey, B. & Raj, S., “A novel ameliorated Harris Hawk optimizer for solving complex engineering optimization problems,” Int. J. Intell. Syst., 2021, 36, 7641-7681.

Mahapatra, S., Raj, S. & Mohan Krishna, S., “Optimal TCSC location for reactive power optimization using oppositional salp swarm algorithm,” in book: Innovation in Electrical Power Engineering, Communication, and Computing Technology, pp. 413-424, 2020.

Mamandur, K.R.C. & Chenoweth, R.D., “Optimal Control of Reactive Power Flow for Improvements in Voltage Profiles and for Real Power Loss Minimization,” IEEE Trans. Power Appar. & Syst., PAS-100, no. 7, July 1981.

Manbachi, M., “Smart grid adaptive Volt/VAR optimization in distribution networks,” PhD Thesis, Simon Fraser University, 2015.

Mangoli, M.K., Lee, K.Y. & Park, Y.M., “Optimal real and reactive power control using linear programming,” Electr. Power Syst. Res., vol. 26. pp. 1-10, 1993.

Martinez Ramos, J.L., Gomez Exposito, A. & Quintana, V.H., 2005, “Transmission power loss reduction by interior-point methods: implementation issues and practical experience,” *IEE Pro.-Gener. Transm. Distrib., Vol. 152, No. 1*, January 2005.

Mataifa, H., Krishnamurthy, S. & Kriger, C., 2022, “Volt/VAR Optimization: A Survey of Classical and Heuristic Optimization Methods,” *IEEE Access, vol. 10*, 2022, DOI: 10.1109/ACCESS.2022.3146366.

Mehra, R.K., “Hierarchical control design challenge problems for optimal VAR planning and real time voltage control in electric power systems,” Proc. 33rd Conf. Decision & Control, 1994.

Mehrotra, S., “On the implementation of a primal-dual interior-point method,” SIAM J. Optim. vol. 2, no. 4, pp. 575-601, 1992.

Miller, T.J.E., “Reactive Power Control in Electric Systems,” John Wiley and Sons, 1982.

Miranda, V., “Fundamentals of evolution strategies and evolutionary programming,” in K.Y. Lee and M.A. El-Sharkawi (eds.), Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems, John Wiley & Sons Inc., 2008.

Momoh, J.A. & Tomsovic, K., “Overview and literature survey of fuzzy set theory in power systems,” IEEE Trans. on Power Syst., vol. 10, no. 3, Aug. 1995.

Momoh, J.A., “Electric power distribution, automation, protection and control,” Taylor & Francis Group, LLC, CRC Press, Boca Raton, FL., 2007.

Momoh, J.A., 2001, “Electric power system applications of optimization,” Marcel Dekker, Inc., New York,

NY, USA.

Momoh, J.A., El-Harway M.E. & Adapa, R., “A review of selected optimal power flow literature to 1993,” parts I and II, IEEE Trans. Power Syst., vol. 14, no. 1, Feb. 1999, pp. 96-111.

Momoh, J.A., Zheng, W. & D’Arnaud, K., “Fuzzy logic control application to enhance voltage stability of electric power systems,” 15th Intern. Conf. on Intelligent Syst. Appl. to Power Syst., pp. 1-6, 2009.

Mota-Palomino R. & Quintana, V.H., “Sparse Reactive Power Scheduling by a Penalty Function-Linear Programming Technique,” IEEE Trans. Power Syst., PWRS-1, no. 3, Aug. 1986.

Myrda, P., “Modeling, simulation and optimization for the 21st century electric power grid,” Electric Power Research Institute Eds., ECI Symposium Series, 2013.

Narita S. & Hammam, M.S.A.A., “A computational algorithm for real-time control of system voltage and reactive power, part I-Problem formulation,” IEEE Winter Power Meeting, New York, January 1971.

Negnevitsky, M. & Le, T.L., “An expert system for teaching voltage control in power systems,” 11th Intern. Conf. on Appl. of Artificial Intelligence in Eng., Florida, 1996.

NERSA, 2014a, “South African Distribution Code: System Operating Code, Version 6.0,” RSA Grid Code Secretariat.

Nicholson, H. & Sterling, M.J.H., “Optimum Dispatch of Active and Reactive Generation by Quadratic Programming,” IEEE Power Eng. Society, 1972.

Nocedal J. & Wright, S.J., “Numerical Optimization,” 2nd ed., Springer Science + Business Media LLC., New York, 2006.

NRS 048-2:2003, *Electricity Supply-Quality of Supply-Part 2: Voltage characteristics, compatibility levels, limits and assessment methods*.

Padilha-Feltrin, A., Rodezno, D.A.Q. & Mantovani, J.R.S., “Volt-VAR multiobjective optimization to peak-load relief and energy efficiency in distribution systems,” IEEE Trans. on Power Deliv., vol. 30, no. 2, April 2015.

Padullaparti, H.V., Nguyen, Q. Santoso, S., “Advances in volt-var control approaches in utility distribution systems, IEEE Power & Energy Society Gen. Meet. (PESGM), pp. 1-5, 2016.

Pandya K.S. & Joshi, S.K., “A survey of optimal power flow methods,” Journal of Theoret. and Appl. Inform. Techn., vol. 4, no. 5, May 2008, pp. 452-458.

Pecas Lopez, J.A., Hatziargyriou, N., Mutale, J., Djapic, P. & Jenkins, N., “Integrating distributed generation into electric power systems: A review of drivers, challenges and opportunities,” Electric Power Syst. Res., vol. 77, pp. 1189-1203, 2007.

Peschon, J., Piercy, D.S., Tinney, W.F., Tveit, O.J. & Cuenod, M., “Optimum control of reactive power flow,” IEEE Trans. Power Appar. & Syst., PAS-87, vol. 1: pp. 40-48, 1968.

Pijarski, P. & Kacejko, P., “Methods of simulated annealing and particle swarm applied to the optimization of reactive power flow in electric power systems,” *Advances in Electrical and Computer Engineering, vol. 18, no. 4*, 2018.

Pilo *et al*., 2014, “Planning and optimization methods for active distribution systems,” Technical Report, *CIGRE Working Group C6.19*, August 2014.

Poli, R., Kennedy, J. & Blackwell, T., “Particle swarm optimization: An overview,” *Swarm Intell (2007)*, 1:33-37, Springer Science + Business Media, LLC, 2007.

Porto, V.W., “Evolutionary programming,” in T. Baeck, D.B. Fogel and Z. Michalewicz (eds.), Handbook of Evolutionary Computation, CRC Press, 1997.

Prabawa, P. & Choi, D-H., “Hierarchical Volt-VAR optimization framework considering voltage control of smart electric vehicle charging stations under uncertainty,” *IEEE Access*, 2019.

Puttgen, H., Volzka, D. & Olken, M., “Restructuring and reregulation of the U.S. electric utility industry,”

IEEE Power Eng. Rev., vol. 21, no. 2, pp. 8-10, Feb. 2001.

Quintana V.H. & Santos-Nieto, M., “Reactive-power dispatch by successive quadratic programming,” IEEE Trans. Energy Conv., vol. 4, no. 3, Sept. 1989.

Rabiee A. & Parniani, M., “Voltage security constrained multi-period optimal reactive power flow using Benders decomposition and optimality condition decompositions,” IEEE Trans. Power Syst., vol. 28, no. 2, May 2013.

Rahimi, K. *et al.*, “Dynamic control of Volt-VAr devices: an effective approach to overcome associated issues with high penetration of solar photovoltaic resources,” IEEE/PES Transm. & Distr. Conf. & Expo. (T&D), pp. 1-5, 2020.

Raj, S. & Bhattacharyya, B., “Optimal placement of TCSC and SVC for reactive power planning using Whale optimization algorithm,” Swarm and Evolutionary Computation 40 (2018), 131-143.

Rao, S.S., “Engineering optimization: theory and practice,” 3rd ed., John Wiley & Sons, Inc., 1996.

Reynolds, C., “Flocks, herds and schools: a distributed behavioral model,” Computer Graphics, 21(4), pp. 25-34, 1987.

Risi, B-G., Riganti-Fulginei, F. & Laudani, A., “Modern techniques for the optimal power flow problem: state of the art,” *Energies, MDPI*, 2022.

Roytelman, I., Wee, B.K. & Lugtu, R.L., “Volt/Var control algorithm for modern distribution management system,” IEEE Trans. Power Syst., vol. 10, no. 3, Aug. 1995.

Russell S. & Norvig, P., “Artificial Intelligence: A Modern Approach,” 3rd ed., Pearson Education, Inc., 2010.

Sahli, Z., Hamouda, A., Bekrar, A. & Trentesaux, D., “Hybrid PSO-Tabu search for the optimal reactive power dispatch problem,” 40th Annual Conf. of IEEE Industr. Electronics Society, IECON 2014, pp. 3536-3542, 2014.

Saric, A.T. & Stankovic, A.M., “A robust algorithm for volt/var control,” IEEE/PES Power Syst. Conf. & Expo., pp. 1-8, 2009.

Savvopoulos, N., Evrenosoglu, C.Y., Marinakis, A., Oudalov, A. & Hatziargyriou, N., “A Long-Term Reactive Power Planning Framework for Transmission Grids with High Shares of Variable Renewable Generation,” 2019 IEEE Milan PowerTech, 2019, pp. 1-6.

Shi, Y. & Eberhart, R., “A modified particle swarm optimizer,” Proceedings of the *IEEE International Conference on Evolutionary Computation*, pp. 69-73, 1998.

Singh, S., Kumar, S., Kalla, U.K., Chandra, A. & Saad, M., “Optimization of rooftop PV system deployment for LV distribution network,” Intern. Conf. on Sustainable Energy & Future Electr. Transp. (SeFeT), Jan. 2021, GRIET, Huderabad, India.

Sivalingam, C.M.K., Ramachandran, S. & Rajamani, P.S.S., “Reactive power optimization in a power system network through metaheuristic algorithms,” Turkish Journal of Electr. Eng. & Comp. Sciences, vol. 25, pp. 4615-4623, 2017.

Springer Verlag, 2012, “Appendix E: IEEE 118-bus test system data,” [online]. Available at: https://link.springer.com/content/pdf/bbm%3A978-1-4615-4473-9%2F1.pdf. Accessed May 05, 2022.

Storn R., & Price, K., “Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces,” Tech. Rep. TR-95-012, March 1995.

Su, C.-T. & Lin, C.-T., “A new fuzzy control approach to voltage profile enhancement for power systems,” IEEE Trans. Power Syst., vol. 11, no. 3, Aug. 1996.

Subbaraj, P. & Rajnarayanan, P.N., “Optimal reactive power dispatch using self-adaptive real-coded genetic algorithm,” Electric Power Syst. Res. vol. 79, 374-381, 2009.

Talukder, S., “Mathematical modeling and applications of particle swarm optimization,” MSc Thesis,

Blekinge Institute of Technology, Karlskrona, Sweden, 2011.

Taylor, G.A., Song, Y.-H. Irving, M.R. Bradley, M.E. & Williams, T.G., “A review of algorithmic and heuristic based methods for voltage/var control,” 5th Intern. Power Eng. Conf. (IPEC2001), vol. 1, pp. 117-122, 2001.

Tomsovic, K., “A fuzzy linear programming approach to the reactive power/voltage control problem,” IEEE Trans. on Power Syst., vol. 7, no. 1, Feb. 1992.

Torres G. & Quintana, V., “An interior-point method for nonlinear optimal power flow using voltage rectangular coordinates,” IEEE Trans. Power Syst., vol. 13, no. 4, pp. 1211-1218, 1998.

Torres, G.L., 1998, “Nonlinear Optimal Power Flow by Interior and Non-Interior Point Methods,” Ph.D. dissertation, Electrical Engineering, University of Waterloo, Canada.

Venkatesh, B., Sadasivam, G. & Abdullah Khan, M., “A new optimal reactive power scheduling method for loss minimization and voltage stability margin maximization using successive multi-objective fuzzy LP technique,” IEEE Trans. Power Syst., vol. 15, no. 2, pp. 844-851, May 2000.

Vitor T.S. & Vieira, J.C.M., “A robust volt/var control via multi-objective optimization,” 13th IEEE Intern. Conf. Indust. Appl., 2018.

Wagner, W.R., Keyhani, A., Hao, S. & Wong, T.C., “A rule-based approach to decentralized voltage control,” IEEE Trans. on Power Syst., vol. 5, no. 2, May 1990.

Wang, P., Mou, Q., Liu, X., Gu, W. & Chen, W., “Start-up control of a synchronous condenser integrated HVDC system with power electronics based static frequency converter,” *IEEE Access*, vol. 7, pp. 146914-146921, 2019.

Wilson, E.O., “Sociobiology: The new synthesis,” *Belknap Press*, Cambridge, MA, 1975.

Wood, A.J., Wollenberg, B.F. & Sheble, G.B., “Power Generation, Operation and Control,” 3rd ed., John Wiley & Sons, Inc., 2014.

Wright, S.J., “Primal-dual interior-point methods, Society for Industr. & Appl. Math., Univ. City Science Center, Philadelphia, 1997.

Wu Q.H. & Ma, J.T., “Power system optimal reactive power dispatch using evolutionary programming,” IEEE Trans. on Power Syst., vol. 10, no. 3., 1995

Xiang-jun, L., “Research and development on substation voltage and reactive power control based on expert system,” 6th IEEE Conf. on Industr. Electronics & Appl., pp. 489-492, 2011.

Xu T. & Wu, W., “Accelerated ADMM-Based Fully Distributed Inverter-Based Volt/Var Control Strategy for Active Distribution Networks,” IEEE Transactions on Industrial Informatics, Vol. 16, No. 12, Dec. 2020.

Yokoyama, R., Nimura, T. & Nakanish, Y., “A coordinated control of voltage and reactive power by heuristic modeling and approximate reasoning,” IEEE Trans. on Power Syst., vol. 8, no. 2, May 1993.

Yu, D.C., Fagan, J.E., Foote, B. & Aly, A.A., “An optimal load flow study by the generalized reduced gradient approach,” Electr. Power Syst. Res., vol. 10, 47-53, 1986.

Zhang, W., Li, F. & Tolbert, L.M., “Review of reactive power planning: objectives, constraints and algorithms,” IEEE Trans. Power Syst., vol. 22, no. 4, Nov. 2007, pp. 2177-2186.

Zhang, Y., Wang, X., Wang, J. & Zhang, Y., “Deep Reinforcement Learning Based Volt-VAR Optimization in Smart Distribution Systems,” IEEE Transactions on Smart Grid, Vol. 12, No. 1, Jan. 2021.

Zheng, W., Wu, W., Zhang, B. & Wang, Y., “Robust reactive power optimization and voltage control method for active distribution networks via dual time-scale coordination,” IET Gener. Transm. Distrib., vol. 11, Iss. 6, pp. 1461-1471, 2017

Zhou, X., Wei, K., Ma, Y. & Gao, Z., “A Review of Reactive Power Compensation Devices,” 2018 IEEE International Conference on Mechatronics and Automation (ICMA), 2018, pp. 2020-2024.

Zhu, J., “Optimization of power system operation,” John Wiley & Sons, Inc., Hoboken, New Jersey, 2009.

APPENDICES

APPENDIX A: SOFTWARE PROGRAMS FOR CHAPTER 4

A.1 Function to implement the interior-point method (IPM) for a general nonlinear programming problem with inequality constraints

```
function [x,dx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu, gradL_norm,fval,
cond]=ipm_generic_nonlinear(f3,x0,s0,mu0,h3, sigma)
zeta=.99995;
x=x0;
s=s0;
mu=mu0;
niq=length(h3(x));
e=ones(niq,1);

S=diag(s);
Si=inv(S);
lam_i=mu*Si*e;

eps_1=1e-4;
eps_2=1e-4;
eps_mu=1e-4;
maxIt=30;

Lam_i=diag(lam_i);
%mu=sigma*(s'*lam_i)/2/niq;

grad_norm=[];
X=[];
Mu=[];

[f, h, dh, gL, g2L]=f3(x, lam_i);

c1=max(h);
c2=(norm(gL,inf)/(1+norm(x)+norm(lam_i)));
c3=((s'*lam_i)/(1+norm(x)));
c4=(abs(f)/(1+abs(f)));
c5=mu;

grad_norm=[grad_norm; c2];
X=[X;x0'];
Mu=[Mu; c5];

converged=c1<=0&&c2<=eps_1&&c3<=eps_1&&c4<=eps_2&&c5<=eps_mu;

fprev=f;
i=1;
while (~converged && i<=maxIt)
A=g2L+dh'*Si*Lam_i*dh;
B=gL+dh'*Si*(mu*e+Lam_i*h);
gradL_norm=norm(gL,inf);
dx=-A\B;
ds=-h-s-dh*dx;
dlam_i=Si*(-S*lam_i+mu*e-Lam_i*ds);
```

```
k_ds=find(ds<0);
if(k_ds)
    alpha_p=min(1, zeta*min(-s(k_ds)./ds(k_ds)));
else
    alpha_p=1;
end
k_dlam_i=find(dlam_i<0);
if(k_dlam_i)
    alpha_d=min(1, zeta*min(-lam_i(k_dlam_i)./dlam_i(k_dlam_i)));
else
    alpha_d=1;
end
alpha=min(alpha_p,alpha_d);
%alpha_p=1;
x=x+alpha*dx;
s=s+alpha*ds;
lam_i=lam_i+alpha*dlam_i;
mu=sigma*(s'*lam_i)/2/niq;
S=diag(s);
Si=inv(S);
Lam_i=diag(lam_i);

[f, h, dh, gL, g2L]=f3(x, lam_i);

fval=f;
i=i+1;

c1=max(h);
c2=norm(gL,inf)/(1+norm(x)+norm(lam_i));
c3=s'*lam_i/(1+norm(x));
c4=abs(f-fprev)/(1+abs(f));
c5=mu;
converged=c1<=0&&c2<=eps_1&&c3<=eps_1&&c4<=eps_2&&c5<=eps_mu;
fprev=f;
cond.num_iterations=i;

grad_norm=[grad_norm; c2];
X=[X;x'];
Mu=[Mu; c5];

end
cond.primal_feasibility=c1;
cond.grad_condition=c2;
cond.comp_condition=c3;
cond.objective_condition=c4;
cond.barrier_condition=c5;
cond.A=A;
cond.B=B;
cond.h=h;
cond.S=S;
cond.Lami=Lam_i;
cond.grad_norm=grad_norm;
```

```
cond.X=X;
cond.Mu=Mu;
```

A.2 Function to define the objective function, constraints, and the Jacobian and Hessian of the Lagrangian of the problem

```
function [f, h, dh, gL, g2L]=func(x, lam_i)
f=x(1)*x(2);
h=[x(1)-x(2)-2; -x(1)+x(2)+1];
dh=[1 -1; -1 1];
grad_f=[x(2); x(1)];
grad_2_f=[0 1; 1 0];

grad_h_t_lam=[lam_i(1)-lam_i(2); -lam_i(1)+lam_i(2)];
grad_2_h_t_lam=0;
gL=grad_f+grad_h_t_lam;
g2L=grad_2_f+grad_2_h_t_lam;
```

A.3 MATLAB script that calls the IPM to implement the example problem in section 4.3.8

```
clear
clc
% File name: ipm_generic_nolinear_test.m
%================================================================
% Test script to test the interior-point algorithm on a generic
% nonlinear function
%================================================================
% Define initial solution estimate and IPM parameters:
% slack variables, barrier parameter, and centering parameter
x0=[2 1]';
s0=[1 1]';
mu0=10;
sigma=.15;

% Call the interior-point algorithm:
tic
[x,dx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu, gradL_norm,fval,
cond]=ipm_generic_nonlinear(@func,x0,s0,mu0,@nonlincon, sigma)
toc

% Extract some results:
t=1:length(cond.Mu);
%t=t(1:6);
x1=cond.X(:,1);
%x1=x1(1:6);
x2=cond.X(:,2);
%x2=x2(1:6);
```

```
gNorm=cond.grad_norm;
%gNorm=gNorm(1:6);
mu=cond.Mu;
%mu=mu(1:6);

% Plot the results:
figure(1)
plot(t,x1,t,x2,'r--');
title('Trajectories of x_1 and x_2 over the PDIPA''s iterations for problem
(4.29)');
legend('x_1','x_2');
xlabel('Iteration number');
ylabel('x_1,x_2');
axis([1 7 -1 2]);
grid

figure(2)
[hAx, ~, hLine2]=plotyy(t,gNorm,t,mu);
title('Gradient norm and barrier parameter trajectories for problem (4.29)')
legend('Gradient norm of the Lagrangian','Barrier parameter (\mu)')
xlabel('Iteration number')
ylabel(hAx(1),'Gradient norm of Lagrangian');
ylabel(hAx(2),'Barrier parameter (\mu)');
hLine2.LineStyle='--';
axis([1 7 0 .2]);
grid
```

APPENDIX B: SOFTWARE PROGRAMS FOR CHAPTER 5

B.1 Function that computes the residues of the load flow problem

```
function dF=dF(x, Vgref)
global E F G B bus_data nbus Cf

% Define some parameters:
bus_type=bus_data(:,1);
Pg=bus_data(:,4);
Qg=bus_data(:,5);
Pd=bus_data(:,6);
Qd=bus_data(:,7);
Pdg=Pd-Pg;
Qdg=Qd-Qg;
Vsq=E.^2+F.^2;

% Update the voltage vectors E, F from the input x
E(1)=sqrt(Vgref(1)^2-F(1)^2);

k=1;
for i=2:nbus
    E(i)=x(k);
    F(i)=x(k+1);
    k=k+2;
end

% Generate the vector of bus power and voltage residues:
dF=[];
for i=2:nbus
    Pi=G(i,i)*Vsq(i)+Pdg(i);    % Each bus has a real power balance residue

    for j=1:nbus
        if (Cf(i,j)==1 && i~=j)
            Pi=Pi+E(i)*(G(i,j)*E(j)-B(i,j)*F(j))+...
                F(i)*(G(i,j)*F(j)+B(i,j)*E(j));
        end
    end
    dF=[dF Pi];

    if (bus_type(i)==2) % PV bus, requires voltage balance residue
        dVi2=Vgref(i)^2-Vsq(i);
        dF=[dF dVi2];
    else                % PQ bus, requires reactive power balance residue
        Qi=-B(i,i)*Vsq(i)+Qdg(i);

        for j=1:nbus
            if (Cf(i,j)==1 && i~=j)
                Qi=Qi+F(i)*(G(i,j)*E(j)-B(i,j)*F(j))-...
                    E(i)*(G(i,j)*F(j)+B(i,j)*E(j));
            end
        end
        dF=[dF Qi];
    end
```

```
end
dF=dF';
```

B.2 Function that computes the Jacobian of the power flow equations for the load flow problem

```
function J=jacobian(x)
global E F G B nbus Cf bus_type

% Update the voltage vectors E, F from the input x

k=1;
for i=2:nbus
    E(i)=x(k);
    F(i)=x(k+1);
    k=k+2;
end

% Generate the Jacobian of bus power and voltage residues:
J=[];   % Initialize Jacobian to empty matrix:
for i=2:nbus

    dPi=[];    % Vector of partial derivatives of Pi:
    for j=2:nbus
        if (i==j)   % dPi/dei and dPi/dfi
            dPidei=2*G(i,i)*E(i);
            dPidfi=2*G(i,i)*F(i);

            for k=1:nbus
                if (Cf(i,k)==1 && i~=k)
                    dPidei=dPidei+G(i,k)*E(k)-B(i,k)*F(k);
                    dPidfi=dPidfi+G(i,k)*F(k)+B(i,k)*E(k);
                end
            end
            dPi=[dPi dPidei dPidfi];
        else
            dPidej=0;
            dPidfj=0;
            if (Cf(i,j)==1 && i~=j)
                dPidej=G(i,j)*E(i)+B(i,j)*F(i);
                dPidfj=G(i,j)*F(i)-B(i,j)*E(i);
            end
            dPi=[dPi dPidej dPidfj];
        end
    end
    J=[J; dPi];

    if (bus_type(i)==2) % PV bus, vector of partial derivatives of dVi^2:
        dVi2=[];
        for j=2:nbus
            dVi2dej=0;
            dVi2dfj=0;
```

```
            if (i==j)
                dVi2dej=-2*E(i);
                dVi2dfj=-2*F(i);
            end
            dVi2=[dVi2 dVi2dej dVi2dfj];
        end
        J=[J; dVi2];
    else                % PQ bus, vector of partial derivatives of dQi:
        dQi=[];
            for j=2:nbus
                if (i==j)    % dQi/dei and dQi/dfi
                    dQidei=-2*B(i,i)*E(i);
                    dQidfi=-2*B(i,i)*F(i);

                    for k=1:nbus
                        if (Cf(i,k)==1 && i~=k)
                            dQidei=dQidei-G(i,k)*F(k)-B(i,k)*E(k);
                            dQidfi=dQidfi+G(i,k)*E(k)-B(i,k)*F(k);
                        end
                    end
                    dQi=[dQi dQidei dQidfi];
                else
                    dQidej=0;
                    dQidfj=0;
                    if (Cf(i,j)==1 && i~=j)
                        dQidej=G(i,j)*F(i)-B(i,j)*E(i);
                        dQidfj=-G(i,j)*E(i)-B(i,j)*F(i);
                    end
                    dQi=[dQi dQidej dQidfj];
                end
            end
            J=[J; dQi];
    end
end
```

B.3 Function that implements the Newton-Raphson load flow algorithm

```
function [V, output]=NR_load_flow(dF, J, x0, Vgref)
global E F
x=x0;
tol=1e-3;
maxIt=10;

mismatch=max(norm(dF(x,Vgref), inf));
converged=(mismatch<=tol);

i=1;

while (~converged && i<maxIt)
    dV=-J(x)\dF(x,Vgref);
    x=x+dV;
```

```
    mismatch=max(norm(dF(x,Vgref), Inf));
    converged=(mismatch<=tol);
    i=i+1;
end
V=E+1i*F;
Vm=sqrt(E.^2+F.^2);
Va=180/pi*atan(F./E);

result='Failed to converge';
if (converged)
    result='converged successfully';
end

output.num_iter=i;
output.exit_flag=converged;
output.result=result;
output.mismatch=mismatch;
output.V=[Vm Va];
```

B.4 MATLAB script that runs the Newton-Raphson load flow computation for the 3-bus system

```
clear

close all
clc
% File name: three_bus_system_NR_load_flow.m

% List of functions needed to run this program:
% (further details are provided for each of these functions
%  where they are first called in the program):

% 1.  [G, B, Cf]=compute_Ybus(from_bus, to_bus, r, x)
% 2.  [R, X, Cf]=computeRX(from, to, r, x)
% 3.  x0=define_x0()
% 4.  Vgref=Vgref_0()
% 5.  dF=dF(x, Vgref)
% 6.  J=jacobian(x)
% 7.  [V, output]=NR_load_flow(@dF, @jacobian, x0, Vgref)
%
% Bus Data:
% Volt/VAR optimization for a 3-bus system:

% Number of buses       : 3
% Number of lines       : 3
% Number of generators  : 2
% Number of loads       : 1

global bus_data Cf E F G B bus_type nbus

% bus_data is matrix in which each row applies to a bus, and specifies
% (numbers in parentheses are column numbers of the bus_data matrix):
```

```
%
% (1) bus type (1=slack bus, 2=PV bus, 3=PQ bus)
% (2) Real component of bus voltage (E)
% (3) Imaginary component of bus voltage (F)
% (4) Generated real power (Pg)
% (5) Generated reactive power (Qg)
% (6) Real power demand (Pd)
% (7) Reactive power demand (Qd)

% Cf is the bus connectivity matrix; Cf(i,j)=1 when buses i and j are
% connected, zero otherwise

% E and F are vectors containing real and imaginary components of the
% bus voltages (in rectangular coordinates)

% G and B are conductance and susceptance matrices respectively;
% i.e. Y=G+jB, where Y is the bus admittance matrix

% bus_type is simply the first column of the bus_data matrix

% nbus is the number of buses in the system

% Values are all in per-unit

%=====================================================================
%     Bus type    V_real     V_imag  Pg      Qg     Pd          Qd
bus_data=[1         1.02         0     0       0      0          0  ;...
          3          1           0     0       0      2         .5  ;...
          2         1.03         0    1.5      0      0          0 ];
%=====================================================================

bus_type=bus_data(:,1);
nbus=length(bus_type);
% bus_data(2,end)=-.5;
% Line Data:
% Corresponding elements of the from_bus and to_bus vectors are the bus
% pairs of connected buses; a line or branch (i,j) exists between bus
% pairs (from_bus(i), to_bus(j));
% r and x are vectors of line resistance and reactance values,
% corresponding to the lines specified by (from_bus(i), to_bus(j))

from_bus=[1 1 2];
to_bus=[2 3 3];
r=[.02 .0059 .0055];
x=[.06 .0235 .0183];

% Compute the admittance (in the form G+jB) and bus connectivity (Cf)
% matrices
% Function compute_Ybus() calls function computeRX(), which computes
% the resistance (R) and reactance (X) vectors, needed by the
% function compute_Ybus(), as well as matrix Cf

[G, B, Cf]=compute_Ybus(from_bus, to_bus, r, x);
```

```
% Define some parameters:
% Extract the vectors of real and imaginary bus voltage components
% from the bus_data matrix

E=bus_data(:,2);
F=bus_data(:,3);

% Define initial input to Newton-Raphson load flow algorithm
% (initial bus voltages,in rectangular form):

x0=define_x0();

% Define the reference voltage vector for the generator voltages;
% this is required for the Volt/VAR optimization problem when running
% the load flow algorithm at each Newton method iteration

Vgref=Vgref_0();

% Perform Newton-Raphson load flow
% Newton-Raphson algorithm implemented in rectangular coordinates of
% bus voltages; assumes that bus 1 is the (only) slack bus. Calls functions
% dF() of residues (of the real and reactive power/voltage balance
% equations), as well as jacobian(), which computes the Jacobian of the
% residues, needed to compute the Newton step once the algorithm
% (hopefully) converges, the bus voltages are output in polar form
% (i.e. magnitude and phase angle)
global deltaF JJ deltaX X
deltaF=[];
JJ=[];
deltaX=[];
X=[];

format long

tic
[~, output]=NR_load_flow1(@dF, @jacobian, x0, Vgref);
toc
v=[output.V(:,1) output.V(:,2)]

% mismatch=output.dF
% J=output.jacobian
% dX=output.dX
% XX=output.X

deltaF
JJ
deltaX
X

% Computes system losses after the load flow algorithm has terminated

[losses, ~]=loss_func();
losses=-losses
```

B.5 Function that implements the primal-dual interior-point method-based Volt/VAR optimization (PDIPM-VVO) for the 3-bus system, incorporating the Newton-Raphson load flow computation

```
function [X,adx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu,fval, cond,
output]=ipm_3bus(Func,x0,s0,mu0, sigma, h, dF, J, Vgref, maxIter)
global E F
%zeta=.99995;
zeta=.9995;

[V, output]=NR_load_flow(dF, J, x0, Vgref);
E=real(V);
F=imag(V);

% Initial load flow voltage results:
Vinit=output.V(:,1);

% Initial system real power losses:
[initial_loss, ~, ~]=loss_func_3bus();

% update control vector x from generator voltage vector components E, F:
x=update_control_vector();

s=s0;
mu=mu0;
niq=length(h(x));
e=ones(niq,1);

S=diag(s);
Si=inv(S);
lam_i=mu*Si*e;
Lam_i=diag(lam_i);

eps_1=1e-3;
eps_2=1e-3;
eps_mu=1e-3;

i=1;

[f, h, dh, gL, g2L]=Func(x, lam_i);

c1=(max(h)<0 || norm(h, inf)<=1e-3);
c2=(norm(gL,inf)/(1+norm(x)+norm(lam_i)));
c3=((s'*lam_i)/(1+norm(x, inf)));
c4=(abs(f)/(1+abs(f)));
c5=mu;
c6=output.exit_flag;

converged=c1&&c2<=eps_1&&c3<=eps_2&&c4<=eps_2&&c5<=eps_mu&&c6;

fprev=f;
fval=[];
```

```
fval=[fval; f];
X=[];
X=[X x];
adx=[];
iter=[];
iter=[iter; i];
Loss=[];
Loss=[Loss; initial_loss];

Vslack=[E(1) F(1)];

while (~converged && i<=maxIter)

    A=g2L+dh'*Si*Lam_i*dh;
    b=gL+dh'*Si*(mu*e+Lam_i*h);
    dx=-A\b;
    ds=-h-s-dh*dx;
    dlam_i=Si*(-S*lam_i+mu*e-Lam_i*ds);

k_ds=find(ds<0);
if(k_ds)
    alpha_p=min(1, zeta*min(-s(k_ds)./ds(k_ds)));
else
    alpha_p=1;
end
k_dlam_i=find(dlam_i<0);
if(k_dlam_i)
    alpha_d=min(1, zeta*min(-lam_i(k_dlam_i)./dlam_i(k_dlam_i)));
else
    alpha_d=1;
end
alpha=min(alpha_p,alpha_d);
%alpha_p=1;
if (alpha_p>.1 && alpha_d>.1)
    alpha_p=alpha_d;
end

x=x+alpha_p*dx;
s=s+alpha_p*ds;
lam_i=lam_i+alpha_d*dlam_i;
mu=sigma*(s'*lam_i)/2/niq;
S=diag(s);
Si=inv(S);
Lam_i=diag(lam_i);
i=i+1;

% Define updated initial voltage vector for the NR load flow algorithm:
x0=define_updated_x0(x);

% Define generator voltage reference vector for the NR load flow algorithm
Vgref=updated_Vgref(x);

[V, output]=NR_load_flow(dF, J, x0, Vgref);
E=real(V);
```

```
F=imag(V);

% Vslack:
Vslack=[Vslack; E(1) F(1)];


[x(1), x(2), x(3)]=deal(E(1), E(3), F(3));

[f, h, dh, gL, g2L]=Func(x, lam_i);
loss=f;

fval=[fval;f];
X=[X x];
adx=[adx alpha_p*dx];
iter=[iter; i];
Loss=[Loss; loss];

c1=(max(h)<0 || norm(h, inf)<=1e-3);
c2=norm(gL,inf)/(1+norm(x)+norm(lam_i));
c3=s'*lam_i/(1+norm(x, inf));
c4=abs(f-fprev)/(1+abs(f));
c5=mu;
c6=output.exit_flag;
converged=c1&&c2<=eps_1&&c3<=eps_1&&c4<=eps_2&&c5<=eps_mu&&c6;
fprev=f;
cond.num_iterations=i;

end

cond.primal_feasibility=max(h);
cond.grad_condition=c2;
cond.comp_condition=c3;
cond.objective_condition=c4;
cond.barrier_condition=c5;
cond.A=A;
cond.b=b;
cond.h=h;
cond.S=S;
cond.Lami=Lam_i;
cond.initial_loss=initial_loss;
cond.loss=Loss;
cond.iter=iter;
cond.V1=Vslack;
cond.Vinit=Vinit;
```

B.6 Function that computes the objective function, its gradient and Hessian for the 3-bus system

```
function [f, df, d2f]=loss_func_3bus()
global E F G

f=G(1,2)*((E(1)-E(2))^2+(F(1)-F(2))^2)+...
```

```
    G(1,3)*((E(1)-E(3))^2+(F(1)-F(3))^2)+...
    G(2,3)*((E(2)-E(3))^2+(F(2)-F(3))^2);

df=2*[G(1,2)*(E(1)-E(2))+G(1,3)*(E(1)-E(3));...
    -G(1,3)*(E(1)-E(3))-G(2,3)*(E(2)-E(3));...
    -G(1,3)*(F(1)-F(3))-G(2,3)*(F(2)-F(3))];

d2f=2*[G(1,2)+G(1,3) -G(1,3) 0;...
    -G(1,3) G(1,3)+G(2,3) 0;...
    0 0 G(1,3)+G(2,3)];
```

B.7 Function that computes the constraint functions, their Jacobian and Hessian for the 3-bus system

```
function [h, dh, dht_lam_i, d2ht_lam_i]=h_gradh_hessh_3bus(x, lam_i)
global E F nbus

Vmsq=.95^2;
VMsq=1.1^2;

[E(1), E(3), F(3)]=deal(x(1), x(2), x(3));

% Define inequality constraints h(x):
h=zeros(2*nbus,1);
k=1;
for i=1:nbus
    h(k)=-(E(i)^2+F(i)^2)+Vmsq;
    h(k+1)=E(i)^2+F(i)^2-VMsq;
    k=k+2;
end

% Define the Jacobian of the inequality constraints dh(x):
% (transposed and multiplied with the Lagrangian multiplier vector lambda_i):
dh=zeros(2*nbus, length(x));
[dh(1,1), dh(2,1), dh(5,2), dh(5,3), dh(6,2), dh(6,3)]=...
    deal(-2*E(1), 2*E(1), -2*E(3), -2*F(3), 2*E(3), 2*F(3));

dht_lam_i=2*[E(1)*(lam_i(2)-lam_i(1));...
             E(3)*(lam_i(6)-lam_i(5));...
             F(3)*(lam_i(6)-lam_i(5))];

d2ht_lam_i_diag=2*[lam_i(2)-lam_i(1), lam_i(6)-lam_i(5), lam_i(6)-lam_i(5)];
d2ht_lam_i=diag(d2ht_lam_i_diag);
```

B.8 Function that computes the Jacobian and Hessian of the Lagrangian of the VVO problem for the 3-bus system

```
function [f, h, dh, gL, g2L]=f_3bus(x, lam_i)

% Define objective function, its gradient and hessian [f, df, d2f]:
```

```
[f, df, d2f]=loss_func_3bus();

% Define inequality constraints, the Jacobian and hessian [h, dht_lam_i,
d2ht_lam_i]:
[h, dh, dht_lam_i, d2ht_lam_i]=h_gradh_hessh_3bus(x, lam_i);

% Define gradient and Hessian of Lagrangian, gL, g2L:
gL=df+dht_lam_i;
g2L=d2f-d2ht_lam_i;
```

B.9 MATLAB script that runs the PDIPM-VVO algorithm for the 3-bus system

```
clear

close all
clc

% File name: three_bus_system_pdipm_vvo_test.m

% List of functions needed to run this program:
% (further details are provided for each of these functions
%  where they are first called in the program):

% 1.  [G, B, Cf]=compute_Ybus(from_bus, to_bus, r, x)
% 2.  [R, X, Cf]=computeRX(from, to, r, x)
% 3.  x0=define_x0()
% 4.  Vgref=Vgref_0()
% 5.  dF=dF(x, Vgref)
% 6.  J=jacobian(x)
% 7.  [V, output]=NR_load_flow(@dF, @jacobian, x0, Vgref)
% 8.  h=h_3bus(x)
% 9.  [f, df, d2f]=loss_func_3bus()
% 10. [h, dh, dht_lam_i, d2ht_lam_i]=h_gradh_hessh_3bus(x, lam_i)
% 11. [f, h, dh, gL, g2L]=f_3bus(x, lam_i)
% 12. [X,adx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu,fval, cond, output]=...
%         ipm_3bus(@f_3bus,x0,s0,mu0, sigma, @h_3bus, @dF, @jacobian,...
%         Vgref, maxIter)
% 13. x=update_control_vector()
% 14. update_generator_voltages(x)
% 15. Vgref=updated_Vgref(x)
%
% Bus Data:
% Volt/VAR optimization for a 3-bus system:

% Number of buses       : 3
% Number of lines       : 3
% Number of generators  : 2
% Number of loads       : 1

global bus_data Cf E F G B bus_type nbus

% bus_data is matrix in which each row applies to a bus, and specifies
```

```
% (numbers in parentheses are column numbers of the bus_data matrix):
%
% (1) bus type (1=slack bus, 2=PV bus, 3=PQ bus)
% (2) Real component of bus voltage (E)
% (3) Imaginary component of bus voltage (F)
% (4) Generated real power (Pg)
% (5) Generated reactive power (Qg)
% (6) Real power demand (Pd)
% (7) Reactive power demand (Qd)

% Cf is the bus connectivity matrix; Cf(i,j)=1 when buses i and j are
% connected, zero otherwise

% E and F are vectors containing real and imaginary components of the
% bus voltages (in rectangular coordinates)

% G and B are conductance and susceptance matrices respectively;
% i.e. Y=G+jB, where Y is the bus admittance matrix

% bus_type is simply the first column of the bus_data matrix

% nbus is the number of buses in the system

% Values are all in per-unit

%=====================================================================
%     Bus type    V_real     V_imag  Pg     Qg    Pd          Qd
bus_data=[1         1.02          0     0      0     0          0  ;...
          3          1            0     0      0     2         .5  ;...
          2         1.03          0    1.5     0     0          0 ];
%=====================================================================

bus_type=bus_data(:,1);
nbus=length(bus_type);
% bus_data(2,end)=-.5;
% Line Data:
% Corresponding elements of the from_bus and to_bus vectors are the bus
% pairs of connected buses; a line or branch (i,j) exists between bus
% pairs (from_bus(i), to_bus(j));
% r and x are vectors of line resistance and reactance values,
% corresponding to the lines specified by (from_bus(i), to_bus(j))

from_bus=[1 1 2];
to_bus=[2 3 3];
r=[.02 .0059 .0055];
x=[.06 .0235 .0183];

% Compute the admittance (in the form G+jB) and bus connectivity (Cf)
% matrices
% Function compute_Ybus() calls function computeRX(), which computes
% the resistance (R) and reactance (X) vectors, needed by the
% function compute_Ybus(), as well as matrix Cf

[G, B, Cf]=compute_Ybus(from_bus, to_bus, r, x);
```

```
% Define some parameters:
% Extract the vectors of real and imaginary bus voltage components
% from the bus_data matrix

E=bus_data(:,2);
F=bus_data(:,3);

% Define initial input to Newton-Raphson load flow algorithm
% (initial bus voltages,in rectangular form):

x0=define_x0();

% Define the reference voltage vector for the generator voltages;
% this is required for the Volt/VAR optimization problem when running
% the load flow algorithm at each Newton method iteration

Vgref=Vgref_0();

% Perform Newton-Raphson load flow
% Newton-Raphson algorithm implemented in rectangular coordinates of
% bus voltages; assumes that bus 1 is the (only) slack bus. Calls functions
% dF() of residues (of the real and reactive power/voltage balance
% equations), as well as jacobian(), which computes the Jacobian of the
% residues, needed to compute the Newton step once the algorithm
% (hopefully) converges, the bus voltages are output in polar form
% (i.e. magnitude and phase angle)
% global deltaF JJ deltaX X
% deltaF=[];
% JJ=[];
% deltaX=[];
% X=[];
%
% format long

% tic
% [~, output]=NR_load_flow(@dF, @jacobian, x0, Vgref);
% toc
% v=[output.V(:,1) output.V(:,2)]

% mismatch=output.dF
% J=output.jacobian
% dX=output.dX
% XX=output.X
%
% deltaF
% JJ
% deltaX
% X
%

% Computes system losses after the load flow algorithm has terminated

[losses, ~]=loss_func();
```

```
losses=-losses

% Perform Volt/VAR optimization
% Interior-Point Method (IPM)-based Volt/VAR optimization (VVO); applies
% the Newton method to compute the search direction for the primal-dual
% system of the VVO problem derived on the basis of the perturbed KKT
% (first-order) optimality conditions.
%
% Computation of the Newton step requires calculating the first- and
% second-order partial derivatives of the objective and constraint
% functions. The IPM algorithm only considers the inequality constraints
% (in this implementation only the bus voltage magnitude constraints),
% the equality constraints (real and reactive power balance equations)
% are handled by the (Newton-Raphson) load flow algorithm. Therefore, at
% each iteration of the Newton step of the IPM algorithm, the load flow
% algorithm is executed once the primal and dual variables have been
% updated.
%
% The IPM algorithm needs the following functions to execute:
%
% 1. f_3bus(), which computes the gradient and hessian of the Lagrangian
%      function of the VVO problem, which are required to compute the
%      Newton step;
% 2.  loss_func_3bus(), which is called by f_3bus(), to compute the
%     objective, its gradient and hessian;
% 3.  h_gradh_hessh_3bus(), which is also called by f_3bus, to compute
%     the constraint functions, the Jacobian and hessian thereof as well;
% 4.  h_3bus(), used only once at the beginning of the IPM algorithm
%     to determine the number of constraint function;
% 5.  dF(), which computes the residues needed by the Newton-Raphson
%     load flow algorithm;
% 6.  jacobian(), which computes the Jacobian of the residues, also needed
%     by the Newton-Raphson load flow algorithm;
% 7.  A number of utility functions called by the functions stated above,
%     including update_generator_voltages(), update_control_vector(),
%     define_updated_x0(), updated_Vgref();
%
% The IPM algorithm also requires a number of parameters, such as the
% barrier parameter (mu), the centering parameter (zeta), and the
% choice of initial primal and dual variables.

% Initialize some input parameters:
xc=update_control_vector();
h0=h_3bus(xc);
s0=abs(h0);
s0(s0==0)=.01;
mu0=10;
sigma=.15;
maxIt=3;

% Run the IPM algorithm on the VVO problem:

tic
[X,adx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu,fval, cond, output]=...
```

```
    ipm_3bus6(@f_3bus,x0,s0,mu0, sigma, @h_3bus, @dF, @jacobian, @compute_PQ_3bus,
Vgref, maxIt);
toc

% Output some computation results:

X
mu
cond
output
V=output.V

% Compute the loss reduction:

initial_loss_pu=-cond.loss(1)
final_loss_pu=-cond.loss(end)
loss_reduction_percentage=100*(cond.loss(1)-cond.loss(end))/cond.loss(1)

% Plot the loss reduction vs. the iteration number:
figure(1), plot(cond.V1(:,1)), hold on, plot(cond.V1(:,2),'r'),hold on,
plot(cond.V1(:,2)./cond.V1(:,1),'k--')
legend('E','F','F/E');
title('Fig.1: Slack-bus voltage (real(E) and imaginary(F) components)');
xlabel('Iteration number')
ylabel('V_{slack-pu}')
V1_mag=sqrt(cond.V1(:,1).^2+cond.V1(:,2).^2);
V1_angle=180/pi*atan(cond.V1(:,2)./cond.V1(:,1));
V1=[V1_mag V1_angle]

figure(2)
Vinit=cond.Vinit;
Vfinal=V(:,1);
plot(Vinit,'b-s'), hold on, plot(Vfinal,'r-s'),legend('V_{Initial}','V_{Final}')
grid
title('Voltage profile, final compared with initial');
xlabel('Bus number')
ylabel('V_{bus-pu}')

figure(3)
plot(cond.iter,-cond.loss, 'r',cond.iter,-cond.loss, 'b*');
grid
title('3-bus system real power loss vs. iteration number', 'FontSize', 11.0)
xlabel('Iteration number')
ylabel('P_{loss} (p.u.)')
loss_label=['Percentage loss reduction = ' num2str(loss_reduction_percentage) '%'];

ax=gca;
y_lims=ax.YLim;
text(1.5,y_lims(2)-.0001, loss_label);

loss=-cond.loss;
P1=cond.PQs(1,:)';
Q1=cond.PQs(2,:)';
Q3=cond.PQs(3,:)';
```

```
Qsum=.0015+cond.Qsum';
Vg1=cond.Vgen(1,:)';
Vg3=cond.Vgen(2,:)';

x1=0:length(loss)-1;
x2=0:length(P1)-1;

figure(4)
% subplot(221)
% [hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
% title('Fig.4(a): Slack-bus active power and power loss')
% legend('P_{loss}','P_{slack}')
% xlabel('Iteration number')
% ylabel(hAx(1),'Real power loss');
% ylabel(hAx(2),'Slack-bus real power');
% hLine2.LineStyle='--';
% grid

subplot(311)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
grid

subplot(312)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q3);
title('Bus-3 generator reactive power and power loss')
legend('P_{loss}','Q_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-3 generator reactive power');
hLine2.LineStyle='--';
grid

subplot(313)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Q3);
title('Generator reactive powers')
legend('Q_{slack}','Q_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Generator reactive powers');
hLine2.LineStyle='--';
grid

figure(5)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('Slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
```

```
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid

figure(6)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
grid

figure(7)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q3);
title('Bus-3 generator reactive power and power loss')
legend('P_{loss}','Q_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-3 generator reactive power');
hLine2.LineStyle='--';
grid

figure(8)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Q3);
title('Generator reactive powers')
legend('Q_{slack}','Q_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Bus-3 generator reactive power');
hLine2.LineStyle='--';
grid

figure(9)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Qsum);
title('Total generated reactive power and power loss')
legend('P_{loss}','Q_{gen}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Total generated reactive power');
hLine2.LineStyle='--';
grid

figure(10)
subplot(121)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Vg1);
title('Slack-bus reactive power and voltage magnitude')
legend('Q_{slack}','V_{g1}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Slack-bus voltage magnitude');
hLine2.LineStyle='--';
grid
```

```
subplot(122)
[hAx, ~, hLine2]=plotyy(x2,Q3,x2,Vg3);
title('Bus-3 generator reactive power and voltage magnitude')
legend('Q_{g3}','V_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-3 generator reactive power');
ylabel(hAx(2),'Bus-3 voltage magnitude');
hLine2.LineStyle='--';
grid

figure(11)
subplot(211)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Vg1);
title('Slack-bus reactive power and voltage magnitude')
legend('Q_{slack}','V_{g1}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Slack-bus voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(212)
[hAx, ~, hLine2]=plotyy(x2,Q3,x2,Vg3);
title('Bus-3 generator reactive power and voltage magnitude')
legend('Q_{g3}','V_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-3 generator reactive power');
ylabel(hAx(2),'Bus-3 voltage magnitude');
hLine2.LineStyle='--';
grid


figure(12)
subplot(211)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('Slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid

subplot(212)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Qsum);
title('Total generated reactive power and power loss')
legend('P_{loss}','Q_{gen}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Total generated reactive power');
hLine2.LineStyle='--';
grid
```

B.10 Function that implements the primal-dual interior-point method-based Volt/VAR optimization (PDIPM-VVO) for the 6-bus system, incorporating the Newton-Raphson load flow computation

```
function [X,adx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu,fval, cond,
output]=ipm_6bus(Func,x0,s0,mu0, sigma, h, dF, J, Vgref, maxIter)
global E F
%zeta=.99995;
zeta=.9995;

[V, output]=NR_load_flow(dF, J, x0, Vgref);
E=real(V);
F=imag(V);

% Initial load flow voltage results:
Vinit=output.V(:,1);

% Initial system real power losses:
[initial_loss, ~, ~]=loss_func_6bus();

% update control vector x from generator voltage vector components E, F:
x=update_control_vector();

s=s0;
mu=mu0;
niq=length(h(x));
e=ones(niq,1);

S=diag(s);
Si=inv(S);
lam_i=mu*Si*e;
Lam_i=diag(lam_i);

eps_1=1e-3;
eps_2=1e-3;
eps_mu=1e-3;

i=1;

[f, h, dh, gL, g2L]=Func(x, lam_i);

c1=(max(h)<0 || norm(h, inf)<=1e-3);
c2=(norm(gL,inf)/(1+norm(x)+norm(lam_i)));
c3=((s'*lam_i)/(1+norm(x, inf)));
c4=(abs(f)/(1+abs(f)));
c5=mu;
c6=output.exit_flag;

converged=c1&&c2<=eps_1&&c3<=eps_2&&c4<=eps_2&&c5<=eps_mu&&c6;

fprev=f;
fval=[];
```

```
fval=[fval; f];
X=[];
X=[X x];
adx=[];
iter=[];
iter=[iter; i];
Loss=[];
Loss=[Loss; initial_loss];

Vslack=[E(1) F(1)];

while (~converged && i<=maxIter)

    A=g2L+dh'*Si*Lam_i*dh;
    b=gL+dh'*Si*(mu*e+Lam_i*h);
    dx=-A\b;
    ds=-h-s-dh*dx;
    dlam_i=Si*(-S*lam_i+mu*e-Lam_i*ds);

k_ds=find(ds<0);
if(k_ds)
    alpha_p=min(1, zeta*min(-s(k_ds)./ds(k_ds)));
else
    alpha_p=1;
end
k_dlam_i=find(dlam_i<0);
if(k_dlam_i)
    alpha_d=min(1, zeta*min(-lam_i(k_dlam_i)./dlam_i(k_dlam_i)));
else
    alpha_d=1;
end
alpha=min(alpha_p,alpha_d);
%alpha_p=1;
if (alpha_p>.1 && alpha_d>.1)
    alpha_p=alpha_d;
end

x=x+alpha_p*dx;
s=s+alpha_p*ds;
lam_i=lam_i+alpha_d*dlam_i;
mu=sigma*(s'*lam_i)/2/niq;
S=diag(s);
Si=inv(S);
Lam_i=diag(lam_i);
i=i+1;

% Define updated initial voltage vector for the NR load flow algorithm:
x0=define_updated_x0(x);

% Define generator voltage reference vector for the NR load flow algorithm:
Vgref=updated_Vgref(x);

[V, output]=NR_load_flow(dF, J, x0, Vgref);
E=real(V);
```

```
F=imag(V);

% Vslack:
Vslack=[Vslack; E(1) F(1)];


[x(1), x(2), x(3), x(4), x(5)]=deal(E(1), E(2), F(2), E(3), F(3));

[f, h, dh, gL, g2L]=Func(x, lam_i);

loss=f;

fval=[fval;f];
X=[X x];
adx=[adx alpha_p*dx];
iter=[iter; i];
Loss=[Loss; loss];

c1=(max(h)<0 || norm(h, inf)<=1e-3);
c2=norm(gL,inf)/(1+norm(x)+norm(lam_i));
c3=s'*lam_i/(1+norm(x, inf));
c4=abs(f-fprev)/(1+abs(f));
c5=mu;
c6=output.exit_flag;
converged=c1&&c2<=eps_1&&c3<=eps_1&&c4<=eps_2&&c5<=eps_mu&&c6;
fprev=f;
cond.num_iterations=i;

end

cond.primal_feasibility=max(h);
cond.grad_condition=c2;
cond.comp_condition=c3;
cond.objective_condition=c4;
cond.barrier_condition=c5;
cond.A=A;
cond.b=b;
cond.h=h;
cond.S=S;
cond.Lami=Lam_i;
cond.initial_loss=initial_loss;
cond.loss=Loss;
cond.iter=iter;
cond.V1=Vslack;
cond.Vinit=Vinit;
```

B.11 Function that computes the objective function, its gradient and Hessian for the 6-bus system

```
function [f, df, d2f]=loss_func_6bus()
global E F G
```

```
f=G(1,2)*((E(1)-E(2))^2+(F(1)-F(2))^2)+...
    G(1,4)*((E(1)-E(4))^2+(F(1)-F(4))^2)+...
    G(1,5)*((E(1)-E(5))^2+(F(1)-F(5))^2)+...
    G(2,3)*((E(2)-E(3))^2+(F(2)-F(3))^2)+...
    G(2,4)*((E(2)-E(4))^2+(F(2)-F(4))^2)+...
    G(2,5)*((E(2)-E(5))^2+(F(2)-F(5))^2)+...
    G(2,6)*((E(2)-E(6))^2+(F(2)-F(6))^2)+...
    G(3,5)*((E(3)-E(5))^2+(F(3)-F(5))^2)+...
    G(3,6)*((E(3)-E(6))^2+(F(3)-F(6))^2)+...
    G(4,5)*((E(4)-E(5))^2+(F(4)-F(5))^2)+...
    G(5,6)*((E(5)-E(6))^2+(F(5)-F(6))^2);

df=2*[G(1,2)*(E(1)-E(2))+G(1,4)*(E(1)-E(4))+G(1,5)*(E(1)-E(5));...
    G(2,3)*(E(2)-E(3))-G(1,2)*(E(1)-E(2))+G(2,4)*(E(2)-E(4))+...
    G(2,5)*(E(2)-E(5))+G(2,6)*(E(2)-E(6));...
    G(2,3)*(F(2)-F(3))-G(1,2)*(F(1)-F(2))+G(2,4)*(F(2)-F(4))+...
    G(2,5)*(F(2)-F(5))+G(2,6)*(F(2)-F(6));...
    G(3,5)*(E(3)-E(5))-G(2,3)*(E(2)-E(3))+G(3,6)*(E(3)-E(6));...
    G(3,5)*(F(3)-F(5))-G(2,3)*(F(2)-F(3))+G(3,6)*(F(3)-F(6))];

d2f=2*[G(1,2)+G(1,4)+G(1,5) -G(1,2) 0 0 0;...
    -G(1,2) G(1,2)+G(2,3)+G(2,4)+G(2,5)+G(2,6) 0 -G(2,3) 0;...
    0 0 G(1,2)+G(2,3)+G(2,4)+G(2,5)+G(2,6) 0 -G(2,3);...
    0 -G(2,3) 0 G(2,3)+G(3,5)+G(3,6) 0;...
    0 0 -G(2,3) 0 G(2,3)+G(3,5)+G(3,6)];
```

B.12 Function that computes the constraint functions, their Jacobian and Hessian for the 6-bus system

```
function [h, dh, dht_lam_i, d2ht_lam_i]=h_gradh_hessh_6bus(x, lam_i)
global E F nbus

Vmsq=.95^2;
VMsq=1.1^2;

[E(1), E(2), F(2), E(3), F(3)]=deal(x(1), x(2), x(3), x(4), x(5));

% Define inequality constraints h(x):
h=zeros(2*nbus,1);
k=1;
for i=1:nbus
    h(k)=-(E(i)^2+F(i)^2)+Vmsq;
    h(k+1)=E(i)^2+F(i)^2-VMsq;
    k=k+2;
end

% Define the Jacobian of the inequality constraints dh(x):
% (transposed and multiplied with the Lagrangian multiplier vector lambda_i):
dh=zeros(2*nbus, length(x));

[dh(1,1), dh(2,1), dh(3,2), dh(3,3), dh(4,2), dh(4,3), dh(5,4), dh(5,5),...
```

```
    dh(6,4), dh(6,5)]=deal(-2*E(1), 2*E(1), -2*E(2), -2*F(2), 2*E(2),...
    2*F(2), -2*E(3), -2*F(3), 2*E(3), 2*F(3));

dht_lam_i=2*[E(1)*(lam_i(2)-lam_i(1));...
             E(2)*(lam_i(4)-lam_i(3));...
             F(2)*(lam_i(4)-lam_i(3));...
             E(3)*(lam_i(6)-lam_i(5));...
             F(3)*(lam_i(6)-lam_i(5))];

d2ht_lam_i_diag=2*[lam_i(2)-lam_i(1), lam_i(4)-lam_i(3),...
    lam_i(4)-lam_i(3), lam_i(6)-lam_i(5), lam_i(6)-lam_i(5)];
d2ht_lam_i=diag(d2ht_lam_i_diag);
```

B.13 Function that computes the Jacobian and Hessian of the Lagrangian of the VVO problem for the 6-bus system

```
function [f, h, dh, gL, g2L]=f_6bus(x, lam_i)

% Define objective function, its gradient and hessian [f, df, d2f]:
[f, df, d2f]=loss_func_6bus();

% Define inequality constraints, the Jacobian and hessian [h, dht_lam_i,
d2ht_lam_i]:
[h, dh, dht_lam_i, d2ht_lam_i]=h_gradh_hessh_6bus(x, lam_i);

% Define gradient and Hessian of Lagrangian, gL, g2L:
gL=df+dht_lam_i;
g2L=-d2f+d2ht_lam_i;
```

B.14 MATLAB script that runs the PDIPM-VVO algorithm for the 6_bus system

```
clear

close all
clc

% File name: six_bus_system_pdipm_vvo_test.m

% List of functions needed to run this program:
% (further details are provided for each of these functions
%  where they are first called in the program):

% 1.  [G, B, Cf]=compute_Ybus(from_bus, to_bus, r, x)
% 2.  [R, X, Cf]=computeRX(from, to, r, x)
% 3.  x0=define_x0()
% 4.  Vgref=Vgref_0()
% 5.  dF=dF(x, Vgref)
% 6.  J=jacobian(x)
% 7.  [V, output]=NR_load_flow(@dF, @jacobian, x0, Vgref)
% 8.  h=h_6bus(x)
```

```
% 9.  [f, df, d2f]=loss_func_6bus()
% 10. [h, dh, dht_lam_i, d2ht_lam_i]=h_gradh_hessh_6bus(x, lam_i)
% 11. [f, h, dh, gL, g2L]=f_6bus(x, lam_i)
% 12. [X,adx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu,fval, cond, output]=...
%         ipm_6bus(@f_6bus,x0,s0,mu0, sigma, @h_6bus, @dF, @jacobian,...
%         Vgref, maxIter)
% 13. x=update_control_vector()
% 14. update_generator_voltages(x)
% 15. Vgref=updated_Vgref(x)
%
% Bus Data:
% Volt/VAR optimization for a 6-bus system:
%
% Number of buses       : 6;
% Number of lines       : 11;
% Number of generators  : 3;
% Number of loads       : 3

global bus_data Cf E F G B bus_type nbus

% bus_data is matrix in which each row applies to a bus, and specifies
% (numbers in parentheses are column numbers of the bus_data matrix):
%
% (1) bus type (1=slack bus, 2=PV bus, 3=PQ bus)
% (2) Real component of bus voltage (E)
% (3) Imaginary component of bus voltage (F)
% (4) Generated real power (Pg)
% (5) Generated reactive power (Qg)
% (6) Real power demand (Pd)
% (7) Reactive power demand (Qd)

% Cf is the bus connectivity matrix; Cf(i,j)=1 when buses i and j are
% connected, zero otherwise

% E and F are vectors containing real and imaginary components of the
% bus voltages (in rectangular coordinates)

% G and B are conductance and susceptance matrices respectively;
% i.e. Y=G+jB, where Y is the bus admittance matrix

% bus_type is simply the first column of the bus_data matrix

% nbus is the number of buses in the system

% Values are all in per-unit

%=====================================================================
%     Bus type    V_real     V_imag  Pg     Qg    Pd      Qd
bus_data=[1         1.07         0     0      0     0       0  ;...
          2         1.05         0    .5      0     0       0  ;...
          2         1.05         0    .5      0     0       0  ;...
          3          1           0     0      0     1      .15 ;...
          3          1           0     0      0     1      .15 ;...
          3          1           0     0      0     1      .15];
```

```
%=====================================================================

bus_type=bus_data(:,1);
nbus=length(bus_type);

% Line Data:
% Corresponding elements of the from_bus and to_bus vectors are the bus
% pairs of connected buses; a line or branch (i,j) exists between bus
% pairs (from_bus(i), to_bus(j));
% r and x are vectors of line resistance and reactance values,
% corresponding to the lines specified by (from_bus(i), to_bus(j))

from_bus=[1 1 1 2 2 2 2 3 3 4 5]';
to_bus=[2 4 5 3 4 5 6 5 6 5 6]';
r=[.1 .05 .08 .05 .05 .1 .07 .12 .02 .2 .1];
x=[.2 .2 .3 .25 .1 .3 .2 .26 .1 .4 .3];

% Compute the admittance (in the form G+jB) and bus connectivity (Cf)
% matrices
% Function compute_Ybus() calls function computeRX(), which computes
% the resistance (R) and reactance (X) vectors, needed by the
% function compute_Ybus(), as well as matrix Cf

[G, B, Cf]=compute_Ybus(from_bus, to_bus, r, x);

% Define some parameters:

% Extract the vectors of real and imaginary bus voltage components
% from the bus_data matrix

E=bus_data(:,2);
F=bus_data(:,3);

% Define initial input to Newton-Raphson load flow algorithm
% (initial bus voltages,in rectangular form):

x0=define_x0();

% Define the reference voltage vector for the generator voltages;
% this is required for the Volt/VAR optimization problem when running
% the load flow algorithm at each Newton method iteration

Vgref=Vgref_0();

% Perform Newton-Raphson load flow
% Newton-Raphson algorithm implemented in rectangular coordinates of
% bus voltages; assumes that bus 1 is the (only) slack bus. Calls functions
% dF() of residues (of the real and reactive power/voltage balance
% equations), as well as jacobian(), which computes the Jacobian of the
% residues, needed to compute the Newton step once the algorithm
% (hopefully) converges, the bus voltages are output in polar form
% (i.e. magnitude and phase angle)

tic
```

```
[~, output]=NR_load_flow(@dF, @jacobian, x0, Vgref);
toc
v=[output.V(:,1) output.V(:,2)]

% Computes system losses after the load flow algorithm has terminated

[losses, ~, ~]=loss_func_6bus()
%losses=-losses;

% Perform Volt/VAR optimization:
% Interior-Point Method (IPM)-based Volt/VAR optimization (VVO); applies
% the Newton method to compute the search direction for the primal-dual
% system of the VVO problem derived on the basis of the perturbed KKT
% (first-order) optimality conditions.
%
% Computation of the Newton step requires calculating the first- and
% second-order partial derivatives of the objective and constraint
% functions. The IPM algorithm only considers the inequality constraints
% (in this implementation only the bus voltage magnitude constraints),
% the equality constraints (real and reactive power balance equations)
% are handled by the (Newton-Raphson) load flow algorithm. Therefore, at
% each iteration of the Newton step of the IPM algorithm, the load flow
% algorithm is executed once the primal and dual variables have been
% updated.
%
% The IPM algorithm needs the following functions to execute:
%
% 1. f_6bus(), which computes the gradient and hessian of the Lagrangian
%      function of the VVO problem, which are required to compute the
%      Newton step;
% 2.  loss_func_6bus(), which is called by f_6bus(), to compute the
%     objective, its gradient and hessian;
% 3.  h_gradh_hessh_6bus(), which is also called by f_6bus, to compute
%     the constraint functions, the Jacobian and hessian thereof as well;
% 4.  h_6bus(), used only once at the beginning of the IPM algorithm
%     to determine the number of constraint function;
% 5.  dF(), which computes the residues needed by the Newton-Raphson
%     load flow algorithm;
% 6.  jacobian(), which computes the Jacobian of the residues, also needed
%     by the Newton-Raphson load flow algorithm;
% 7.  A number of utility functions called by the functions stated above,
%     including update_generator_voltages(), update_control_vector(),
%     define_updated_x0(), updated_Vgref();
%
% The IPM algorithm also requires a number of parameters, such as the
% barrier parameter (mu), the centering parameter (zeta), and the
% choice of initial primal and dual variables.

% Initialize some input parameters:

h0=h_6bus(x0);
s0=abs(h0);
s0(s0==0)=.01;
mu0=10;
```

```
sigma=.15;
maxIt=12;

% Run the IPM algorithm on the VVO problem:

tic
[X,adx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu,fval, cond, output]=...
    ipm_6bus6(@f_6bus,x0,s0,mu0, sigma, @h_6bus, @dF, @jacobian, Vgref, maxIt);
toc

% Output some computation results:

X
mu
cond
output
V=output.V

% Compute the loss reduction:

initial_loss_pu=-cond.loss(1)
final_loss_pu=-cond.loss(end)
loss_reduction_percentage=100*(cond.loss(1)-cond.loss(end))/cond.loss(1)

% Plot the loss reduction vs. the iteration number:

figure(1), plot(cond.V1(:,1)), hold on, plot(cond.V1(:,2),'r'),hold on,
plot(cond.V1(:,2)./cond.V1(:,1),'k--')
legend('E','F','F/E');
title('Slack-bus voltage (real(E) and imaginary(F) components)');
xlabel('Iteration number')
ylabel('V_{slack-pu}')
V1_mag=sqrt(cond.V1(:,1).^2+cond.V1(:,2).^2);
V1_angle=180/pi*atan(cond.V1(:,2)./cond.V1(:,1));
V1=[V1_mag V1_angle]

figure(2)
Vinit=cond.Vinit;
Vfinal=V(:,1);
plot(Vinit,'b-s'), hold on, plot(Vfinal,'r-s'),legend('V_{Initial}','V_{Final}')
grid
title('Voltage profile, final compared with initial');
xlabel('Bus number')
ylabel('V_{bus-pu}')

% xlswrite('six_bus_system_voltage.xlsx', [Vinit Vfinal])

figure(3)
plot(cond.iter,-cond.loss, 'r',cond.iter,-cond.loss, 'b*');
grid
title('6-bus system real power loss vs. iteration number', 'FontSize', 10.0)
xlabel('Iteration number')
ylabel('P_{loss} (p.u.)')
loss_label=['Percentage loss reduction = ' num2str(loss_reduction_percentage) '%'];
```

```
ax=gca;
y_lims=ax.YLim;
text(6,y_lims(2)-.0025, loss_label);

loss=-cond.loss;
P1=cond.PQs(1,:)';
Q1=cond.PQs(2,:)';
Q2=cond.PQs(3,:)';
Q3=cond.PQs(4,:)';
Qsum=cond.Qsum';
Vg1=cond.Vgen(1,:)';
Vg2=cond.Vgen(2,:)';
Vg3=cond.Vgen(3,:)';

x1=[0:length(loss)-1]';
x2=[0:length(P1)-1]';

figure(4)
subplot(221)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('Slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid

subplot(222)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
grid

subplot(223)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q2);
title('Bus-2 generator reactive power and power loss')
legend('P_{loss}','Q_{g2}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-2 generator reactive power');
hLine2.LineStyle='--';
grid

subplot(224)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Q3);
title('Generator reactive powers')
legend('Q_{slack}','Q_{g3}')
xlabel('Iteration number')
```

```
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Generator reactive powers');
hLine2.LineStyle='--';
grid

figure(5)
subplot(211)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('6-bus system: slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid
ax=gca;
y_lims=ax.YLim;
text(5,y_lims(2)-.0025, loss_label);

subplot(212)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Qsum);
title('6-bus system: total generated reactive power and power loss')
legend('P_{loss}','Q_{gen}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Total generated reactive power');
hLine2.LineStyle='--';
grid

figure(6)
[hAx6, ~, hLine6]=plotyy(x1,loss,[x2,x2,x2],[Q1,Q2,Q3]);
title('Generator reactive powers and power loss')
legend('P_{loss}','Q_{slack}','Q_{g2}','Q_{g3}')
xlabel('Iteration number')
ylabel(hAx6(1),'Real power loss');
ylabel(hAx6(2),'Generator reactive power outputs');
hLine6(1).LineStyle='--';
hLine6(2).LineStyle=':';
hLine6(2).Color='r';
hLine6(3).LineStyle='-.';
grid

figure(7)
subplot(311)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Vg1);
title('Slack-bus reactive power and voltage magnitude')
legend('Q_{slack}','V_{g1}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Slack-bus voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(312)
```

```
[hAx, ~, hLine2]=plotyy(x2,Q2,x2,Vg2);
title('Bus-2 generator reactive power and voltage magnitude')
legend('Q_{g2}','V_{g2}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-2 generator reactive power');
ylabel(hAx(2),'Bus-2 voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(313)
[hAx, ~, hLine2]=plotyy(x2,Q3,x2,Vg3);
title('Bus-3 generator reactive power and voltage magnitude')
legend('Q_{g3}','V_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-3 generator reactive power');
ylabel(hAx(2),'Bus-3 voltage magnitude');
hLine2.LineStyle='--';
grid
```

B.15 Function that implements the primal-dual interior-point method-based Volt/VAR optimization (PDIPM-VVO) for the 14-bus system, incorporating the Newton-Raphson load flow computation

```
function [X,adx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu,fval, cond,
output]=ipm_14bus(Func,x0,s0,mu0, sigma, h, dF, J, Vgref, maxIter)
global E F PQs Qsum Vgens
%zeta=.99995;
zeta=.9995;

% Added computation of slack-bus active and generator reactive powers

[V, output]=NR_load_flow(dF, J, x0, Vgref);
E=real(V);
F=imag(V);

% Initialize slack-bus active, and generator reactive power outputs
% matrix to empty matrix:
PQs=[];

% Compute slack-bus active, and generator reactive power outputs
compute_PQ();

% Vector to hold generator bus voltages:
Vgens=[];

% Extract generator bus voltages;
get_Vgen();

% Initial load flow voltage results:
Vinit=output.V(:,1);
```

```
% Initial system real power losses:
[initial_loss, ~, ~]=loss_func_14bus();

% update control vector x from generator voltage vector components E, F:
x=update_control_vector();

s=s0;
mu=mu0;
niq=length(h(x));
e=ones(niq,1);

S=diag(s);
Si=inv(S);
lam_i=mu*Si*e;
Lam_i=diag(lam_i);

eps_1=1e-3;
eps_2=1e-3;
eps_mu=1e-3;

i=1;

[f, h, dh, gL, g2L]=Func(x, lam_i);

c1=(max(h)<0 || norm(h, inf)<=1e-3);
c2=(norm(gL,inf)/(1+norm(x)+norm(lam_i)));
c3=((s'*lam_i)/(1+norm(x, inf)));
c4=(abs(f)/(1+abs(f)));
c5=mu;
c6=output.exit_flag;

converged=c1&&c2<=eps_1&&c3<=eps_2&&c4<=eps_2&&c5<=eps_mu&&c6;

fprev=f;
fval=[];
fval=[fval; f];
X=[];
X=[X x];
adx=[];
iter=[];
iter=[iter; i];
Loss=[];
Loss=[Loss; initial_loss];

Vslack=[E(1) F(1)];

while (~converged && i<=maxIter)

    A=g2L+dh'*Si*Lam_i*dh;
    b=gL+dh'*Si*(mu*e+Lam_i*h);
    dx=-A\b;
    ds=-h-s-dh*dx;
    dlam_i=Si*(-S*lam_i+mu*e-Lam_i*ds);
```

```
k_ds=find(ds<0);
if(k_ds)
    alpha_p=min(1, zeta*min(-s(k_ds)./ds(k_ds)));
else
    alpha_p=1;
end
k_dlam_i=find(dlam_i<0);
if(k_dlam_i)
    alpha_d=min(1, zeta*min(-lam_i(k_dlam_i)./dlam_i(k_dlam_i)));
else
    alpha_d=1;
end
alpha=min(alpha_p,alpha_d);
%alpha_p=1;
if (alpha_p>.1 && alpha_d>.1)
    alpha_p=alpha_d;
end

x=x+alpha_p*dx;
s=s+alpha_p*ds;
lam_i=lam_i+alpha_d*dlam_i;
mu=sigma*(s'*lam_i)/2/niq;
S=diag(s);
Si=inv(S);
Lam_i=diag(lam_i);
i=i+1;

% Define updated initial voltage vector for the NR load flow algorithm:
x0=define_updated_x0(x);

% Define generator voltage reference vector for the NR load flow algorithm:
Vgref=updated_Vgref(x);

[V, output]=NR_load_flow(dF, J, x0, Vgref);
E=real(V);
F=imag(V);

% Compute slack-bus active, and generator reactive power outputs
compute_PQ();

% Extract generator bus voltages;
get_Vgen();

% Vslack:
Vslack=[Vslack; E(1) F(1)];

[x(1), x(2), x(3), x(4), x(5), x(6), x(7), x(8), x(9)]=...
    deal(E(1), E(2), F(2), E(3), F(3), E(6), F(6), E(8), F(8));

[f, h, dh, gL, g2L]=Func(x, lam_i);

loss=f;

fval=[fval;f];
```

```
X=[X x];
adx=[adx alpha_p*dx];
iter=[iter; i];
Loss=[Loss; loss];

c1=(max(h)<0 || norm(h, inf)<=1e-3);
c2=norm(gL,inf)/(1+norm(x)+norm(lam_i));
c3=s'*lam_i/(1+norm(x, inf));
c4=abs(f-fprev)/(1+abs(f));
c5=mu;
c6=output.exit_flag;
converged=c1&&c2<=eps_1&&c3<=eps_1&&c4<=eps_2&&c5<=eps_mu&&c6;
fprev=f;
cond.num_iterations=i;

end

cond.primal_feasibility=max(h);
cond.grad_condition=c2;
cond.comp_condition=c3;
cond.objective_condition=c4;
cond.barrier_condition=c5;
cond.A=A;
cond.b=b;
cond.h=h;
cond.S=S;
cond.Lami=Lam_i;
cond.initial_loss=initial_loss;
cond.loss=Loss;
cond.iter=iter;
cond.V1=Vslack;
cond.Vinit=Vinit;
cond.PQs=PQs;
cond.Qsum=Qsum;
cond.Vgen=Vgens;
```

B.16 Function that computes the objective function, its gradient and Hessian for the 14-bus system

```
function [f, df, d2f]=loss_func_14bus()
global E F G

f=abs(G(1,2))*((E(1)-E(2))^2+(F(1)-F(2))^2)+abs(G(1,5))*((E(1)-E(5))^2+(F(1)-
F(5))^2)+...
    abs(G(2,3))*((E(2)-E(3))^2+(F(2)-F(3))^2)+abs(G(2,4))*((E(2)-E(4))^2+(F(2)-
F(4))^2)+...
    abs(G(2,5))*((E(2)-E(5))^2+(F(2)-F(5))^2)+abs(G(3,4))*((E(3)-E(4))^2+(F(3)-
F(4))^2)+...
    abs(G(4,5))*((E(4)-E(5))^2+(F(4)-F(5))^2)+abs(G(6,11))*((E(6)-E(11))^2+(F(6)-
F(11))^2)+...
    abs(G(6,12))*((E(6)-E(12))^2+(F(6)-F(12))^2)+abs(G(6,13))*((E(6)-
E(13))^2+(F(6)-F(13))^2)+...
```

```
    abs(G(9,10))*((E(9)-E(10))^2+(F(9)-F(10))^2)+abs(G(9,14))*((E(9)-
E(14))^2+(F(9)-F(14))^2)+...
    abs(G(10,11))*((E(10)-E(11))^2+(F(10)-F(11))^2)+abs(G(12,13))*((E(12)-
E(13))^2+(F(12)-F(13))^2)+...
    abs(G(13,14))*((E(13)-E(14))^2+(F(13)-F(14))^2);


df=2*[abs(G(1,2))*(E(1)-E(2))+abs(G(1,5))*(E(1)-E(5));...
    abs(G(2,3))*(E(2)-E(3))-abs(G(1,2))*(E(1)-E(2))+abs(G(2,4))*(E(2)-E(4))+...
    abs(G(2,5))*(E(2)-E(5));...
    abs(G(2,3))*(F(2)-F(3))-abs(G(1,2))*(F(1)-F(2))+abs(G(2,4))*(F(2)-F(4))+...
    abs(G(2,5))*(F(2)-F(5));...
    abs(G(3,4))*(E(3)-E(4))-abs(G(2,3))*(E(2)-E(3));...
    abs(G(3,4))*(F(3)-F(4))-abs(G(2,3))*(F(2)-F(3));...
    abs(G(6,11))*(E(6)-E(11))+abs(G(6,12))*(E(6)-E(12))+...
    abs(G(6,13))*(E(6)-E(13));...
    abs(G(6,11))*(F(6)-F(11))+abs(G(6,12))*(F(6)-F(12))+...
    abs(G(6,13))*(F(6)-F(13));...
    0;...
    0];

d2f=zeros(length(df));

d2f(1,1)=2*abs(G(1,2)+G(1,5));
d2f(1,2)=-2*abs(G(1,2));
d2f(2,1)=d2f(1,2);
d2f(2,2)=2*abs(G(1,2)+G(2,3)+G(2,4)+G(2,5));
d2f(2,4)=-2*abs(G(2,3));
d2f(3,3)=2*abs(G(1,2)+G(2,3)+G(2,4)+G(2,5));
d2f(3,5)=-2*abs(G(2,3));
d2f(4,2)=d2f(2,4);
d2f(4,4)=2*abs(G(2,3)+G(3,4));
d2f(5,3)=d2f(3,5);
d2f(5,5)=2*abs(G(2,3)+G(3,4));
d2f(6,6)=2*abs(G(6,11)+G(6,12)+G(6,13));
d2f(7,7)=2*abs(G(6,11)+G(6,12)+G(6,13));
```

B.17 Function that computes the constraint functions, their Jacobian and Hessian for the 14-bus system

```
function [h, dh, dht_lam_i, d2ht_lam_i]=h_gradh_hessh_14bus(x, lam_i)
global E F nbus

Vmsq=.95^2;
VMsq=1.1^2;

[E(1), E(2), F(2), E(3), F(3), E(6), F(6), E(8), F(8)]=...
    deal(x(1), x(2), x(3), x(4), x(5), x(6), x(7), x(8), x(9));


% Define inequality constraints h(x):
h=zeros(2*nbus,1);
```

```
k=1;
for i=1:nbus
    h(k)=-(E(i)^2+F(i)^2)+Vmsq;
    h(k+1)=E(i)^2+F(i)^2-VMsq;
    k=k+2;
end

% Define the Jacobian of the inequality constraints dh(x):
% (transposed and multiplied with the Lagrangian multiplier vector lambda_i):
dh=zeros(2*nbus, length(x));
[dh(1,1), dh(2,1), dh(3,2), dh(3,3), dh(4,2), dh(4,3), dh(5,4), dh(5,5)]=...
    deal(-2*E(1), 2*E(1), -2*E(2), -2*F(2), 2*E(2), 2*F(2), -2*E(3), -2*F(3));
[dh(6,4), dh(6,5), dh(11,6), dh(11,7), dh(12,6), dh(12,7), dh(15,8), dh(15,9),...
    dh(16,8), dh(16,9)]=deal(2*E(3), 2*F(3),-2*E(6), -2*F(6), 2*E(6), 2*F(6),...
    -2*E(8), -2*F(8), 2*E(8), 2*F(8));

dht_lam_i=2*[E(1)*(lam_i(2)-lam_i(1));...
    E(2)*(lam_i(4)-lam_i(3));...
    F(2)*(lam_i(4)-lam_i(3));...
    E(3)*(lam_i(6)-lam_i(5));...
    F(3)*(lam_i(6)-lam_i(5));...
    E(6)*(lam_i(12)-lam_i(11));...
    F(6)*(lam_i(12)-lam_i(11));...
    E(8)*(lam_i(16)-lam_i(15));...
    F(8)*(lam_i(16)-lam_i(15))];
d2ht_lam_i_diag=2*[lam_i(2)-lam_i(1), lam_i(4)-lam_i(3), lam_i(4)-lam_i(3),...
    lam_i(6)-lam_i(5), lam_i(6)-lam_i(5), lam_i(12)-lam_i(11), lam_i(12)-
lam_i(11),...
    lam_i(16)-lam_i(15), lam_i(16)-lam_i(15)];
d2ht_lam_i=diag(d2ht_lam_i_diag);
```

B.18 Function that computes the Jacobian and Hessian of the Lagrangian of the VVO problem for the 14-bus system

```
function [f, h, dh, gL, g2L]=f_14bus(x, lam_i)

% Define objective function, its gradient and hessian [f, df, d2f]:
[f, df, d2f]=loss_func_14bus();

% Define inequality constraints, the Jacobian and hessian [h, dht_lam_i,
d2ht_lam_i]:
[h, dh, dht_lam_i, d2ht_lam_i]=h_gradh_hessh_14bus(x, lam_i);

% Define gradient and Hessian of Lagrangian, gL, g2L:
% gL=-df+dht_lam_i;
% g2L=d2f-d2ht_lam_i;

% This seems to give better results
gL=-df+dht_lam_i;
g2L=-(d2f+d2ht_lam_i);
```

B.19 MATLAB script that runs the PDIPM-VVO algorithm for the 14_bus system

```
clear

close all
clc

% File name: fourteen_bus_system_pdipm_vvo_test.m

% List of functions needed to run this program:
% (further details are provided for each of these functions
%  where they are first called in the program):

% 1.  [G, B, Cf]=compute_Ybus(from_bus, to_bus, r, x)
% 2.  [R, X, Cf]=computeRX(from, to, r, x)
% 3.  x0=define_x0()
% 4.  Vgref=Vgref_0()
% 5.  dF=dF(x, Vgref)
% 6.  J=jacobian(x)
% 7.  [V, output]=NR_load_flow(@dF, @jacobian, x0, Vgref)
% 8.  h=h_14bus(x)
% 9.  [f, df, d2f]=loss_func_14bus()
% 10. [h, dh, dht_lam_i, d2ht_lam_i]=h_gradh_hessh_14bus(x, lam_i)
% 11. [f, h, dh, gL, g2L]=f_14bus(x, lam_i)
% 12. [X,adx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu,fval, cond, output]=...
%         ipm_14bus(@f_14bus,x0,s0,mu0, sigma, @h_14bus, @dF, @jacobian,...
%         Vgref, maxIter)
% 13. x=update_control_vector()
% 14. update_generator_voltages(x)
% 15. Vgref=updated_Vgref(x)
%
% Bus Data:
% Volt/VAR optimization for the IEEE 14-bus system:
%
% Number of buses       : 14;
% Number of lines       : 20;
% Number of generators  : 5;
% Number of loads       : 11;

global bus_data Cf E F G B bus_type nbus

% bus_data is matrix in which each row applies to a bus, and specifies:
% (1) bus type (1=slack bus, 2=PV bus, 3=PQ bus)
% (2) Real component of bus voltage (E)
% (3) Imaginary component of bus voltage (F)
% (4) Generated real power (Pg)
% (5) Generated reactive power (Qg)
% (6) Real power demand (Pd)
% (7) Reactive power demand (Qd)

% Cf is the bus connectivity matrix; Cf(i,j)=1 when buses i and j are
% connected, zero otherwise

% E and F are vectors containing real and imaginary components of the
```

```
% bus voltages (in rectangular coordinates)

% G and B are conductance and susceptance matrices respectively;
% i.e. Y=G+jB, where Y is the bus admittance matrix

% bus_type is simply the first column of the bus_data matrix

% nbus is the number of buses in the system

% Values are all in per-unit

%=========================================================
%     Bus type    V_real     V_imag  Pg   Qg    Pd     Qd
bus_data=[1       1.06         0     0    0      0      0  ;...
          2       1.045        0    .4    0    .217   .127 ;...
          2       1.01         0     0    0    .942   .19  ;...
          3         1          0     0    0    .478  -.039 ;...
          3         1          0     0    0    .076   .016 ;...
          2       1.07         0     0    0    .112   .075 ;...
          3         1          0     0    0      0      0  ;...
          2       1.09         0     0    0      0      0  ;...
          3         1          0     0    0     .295  .166 ;...
          3         1          0     0    0     .09   .058 ;...
          3         1          0     0    0     .035  .018 ;...
          3         1          0     0    0     .061  .016 ;...
          3         1          0     0    0     .135  .058 ;...
          3         1          0     0    0     .149  .05];
%=========================================================

bus_type=bus_data(:,1);
nbus=length(bus_type);

global from_bus to_bus r x

% Uniformly distributed generated power:
% for i=2:nbus
%     if (bus_type(i)==2)
%         bus_data(i,4)=.15;
%     end
% end
% figure(4), plot(bus_data(:,4)), title('Uniformly distributed power generation');


% Line Data:
% Corresponding elements of the from_bus and to_bus vectors are the bus
% pairs of connected buses; a line or branch (i,j) exists between bus
% pairs (from_bus(i), to_bus(j));
% r and x are vectors of line resistance and reactance values,
% corresponding to the lines specified by (from_bus(i), to_bus(j))

from_bus=[1 1 2 2 2 3 4 4 4 5 6 6 6 7 7 9 9 10 12 13]';
to_bus=[2 5 3 4 5 4 5 7 9 6 11 12 13 8 9 10 14 11 13 14]';

r=[.01938 .05403 .04699 .05811 .05695 .06701 .01335 0 0 0 ...
```

```
    .09498 .12291 .06615 0 0 .03181 .12711 .08205 .22092 .17093];
x=[.05917 .22304 .19797 .17632 .17388 .17103 .04211 .20912 ...
    .55618 .25202 .1989 .25581 .13027 .17615 .11001 .0845 ...
    .27038 .19207 .19988 .34802];

% Compute the admittance (in the form G+jB) and bus connectivity (Cf)
% matrices
% Function compute_Ybus() calls function computeRX(), which computes
% the resistance (R) and reactance (X) vectors, needed by the
% function compute_Ybus(), as well as matrix Cf

[G, B, Cf]=compute_Ybus(from_bus, to_bus, r, x);

% Define some parameters:

% Extract the vectors of real and imaginary bus voltage components
% from the bus_data matrix

E=bus_data(:,2);
F=bus_data(:,3);

% Define initial input to Newton-Raphson load flow algorithm
% (initial bus voltages,in rectangular form):

x0=define_x0();

% Define the reference voltage vector for the generator voltages;
% this is required for the Volt/VAR optimization problem when running
% the load flow algorithm at each Newton method iteration

Vgref=Vgref_0();

% Perform Newton-Raphson load flow
% Newton-Raphson algorithm implemented in rectangular coordinates of
% bus voltages; assumes that bus 1 is the (only) slack bus. Calls functions
% dF() of residues (of the real and reactive power/voltage balance
% equations), as well as jacobian(), which computes the Jacobian of the
% residues, needed to compute the Newton step once the algorithm
% (hopefully) converges, the bus voltages are output in polar form
% (i.e. magnitude and phase angle)

tic
[~, output]=NR_load_flow(@dF, @jacobian, x0, Vgref);
toc

v=[output.V(:,1) output.V(:,2)]

% Computes system losses after the load flow algorithm has terminated

[losses, ~, ~]=loss_func_14bus()

% Perform Volt/VAR optimization:
% Interior-Point Method (IPM)-based Volt/VAR optimization (VVO); applies
% the Newton method to compute the search direction for the primal-dual
```

```
% system of the VVO problem derived on the basis of the perturbed KKT
% (first-order) optimality conditions.
%
% Computation of the Newton step requires calculating the first- and
% second-order partial derivatives of the objective and constraint
% functions. The IPM algorithm only considers the inequality constraints
% (in this implementation only the bus voltage magnitude constraints),
% the equality constraints (real and reactive power balance equations)
% are handled by the (Newton-Raphson) load flow algorithm. Therefore, at
% each iteration of the Newton step of the IPM algorithm, the load flow
% algorithm is executed once the primal and dual variables have been
% updated.
%
% The IPM algorithm needs the following functions to execute:
%
% 1. f_14bus(), which computes the gradient and hessian of the Lagrangian
%      function of the VVO problem, which are required to compute the
%      Newton step;
% 2.  loss_func_14bus(), which is called by f_14bus(), to compute the
%     objective, its gradient and hessian;
% 3.  h_gradh_hessh_14bus(), which is also called by f_14bus, to compute
%     the constraint functions, the Jacobian and hessian thereof as well;
% 4.  h_14bus(), used only once at the beginning of the IPM algorithm
%     to determine the number of constraint function;
% 5.  dF(), which computes the residues needed by the Newton-Raphson
%     load flow algorithm;
% 6.  jacobian(), which computes the Jacobian of the residues, also needed
%     by the Newton-Raphson load flow algorithm;
% 7.  A number of utility functions called by the functions stated above,
%     including update_generator_voltages(), update_control_vector(),
%     define_updated_x0(), updated_Vgref();
%
% The IPM algorithm also requires a number of parameters, such as the
% barrier parameter (mu), the centering parameter (zeta), and the
% choice of initial primal and dual variables.

% Initialize some input parameters:

h0=h_14bus(x0);
s0=abs(h0);
s0(s0==0)=.015;
mu0=10;
sigma=.15;
maxIt=13;

% Run the IPM algorithm on the VVO problem:

tic
[X,adx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu,fval, cond, output]=...
    ipm_14bus6(@f_14bus,x0,s0,mu0, sigma, @h_14bus, @dF, @jacobian, Vgref, maxIt);
toc

% Output some computation results:
```

```
X
mu
cond
output
V=output.V

% Compute the loss reduction:

initial_loss_pu=cond.loss(1)
final_loss_pu=cond.loss(end)
loss_reduction_percentage=100*(cond.loss(1)-cond.loss(end))/cond.loss(1)

% Plot the loss reduction vs. the iteration number:

figure(1), plot(cond.V1(:,1)), hold on, plot(cond.V1(:,2),'r'),hold on,
plot(cond.V1(:,2)./cond.V1(:,1),'k--')
legend('E','F','F/E');
title('Slack-bus voltage (real(E) and imaginary(F) components)');
xlabel('Iteration number')
ylabel('V_{slack-pu}')
V1_mag=sqrt(cond.V1(:,1).^2+cond.V1(:,2).^2);
V1_angle=180/pi*atan(cond.V1(:,2)./cond.V1(:,1));
V1=[V1_mag V1_angle]

figure(2)
Vinit=cond.Vinit;
Vfinal=V(:,1);
plot(Vinit,'b-s'), hold on, plot(Vfinal,'r-s'),legend('V_{Initial}','V_{Final}')
grid
title('Voltage profile, final compared with initial');
xlabel('Bus number')
ylabel('V_{bus-pu}')

% xlswrite('fourteen_bus_system_voltage.xlsx', [Vinit Vfinal])

figure(3)
plot(cond.iter,cond.loss, 'r',cond.iter,cond.loss, 'b*');
grid
title('IEEE 14-bus system real power loss vs. iteration number')
xlabel('Iteration number')
ylabel('P_{loss} (p.u.)')
loss_label=['Percentage loss reduction = ' num2str(loss_reduction_percentage) '%'];

ax=gca;
y_lims=ax.YLim;
text(.5,y_lims(2)-.0025, loss_label);

loss=cond.loss;
P1=cond.PQs(1,:)';
Q1=cond.PQs(2,:)';
Q2=cond.PQs(3,:)';
Q3=cond.PQs(4,:)';
Q6=cond.PQs(5,:)';
Q8=cond.PQs(6,:)';
```

```
Qsum=cond.Qsum';
Vg1=cond.Vgen(1,:)';
Vg2=cond.Vgen(2,:)';
Vg3=cond.Vgen(3,:)';
Vg6=cond.Vgen(4,:)';
Vg8=cond.Vgen(5,:)';

x1=[0:length(loss)-1]';
x2=[0:length(P1)-1]';

figure(4)
subplot(221)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('Slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid

subplot(222)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
grid

subplot(223)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q3);
title('Bus-3 generator reactive power and power loss')
legend('P_{loss}','Q_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-3 generator reactive power');
hLine2.LineStyle='--';
grid

subplot(224)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Q2);
title('Generator reactive powers')
legend('Q_{slack}','Q_{g2}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Generator reactive powers');
hLine2.LineStyle='--';
grid


% figure(5)
% [hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
```

```
% title('Fig.5: Slack-bus active power and power loss')
% legend('P_{loss}','P_{slack}')
% xlabel('Iteration number')
% ylabel(hAx(1),'Real power loss');
% ylabel(hAx(2),'Slack-bus real power');
% hLine2.LineStyle='--';
% grid

figure(5)
subplot(211)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('14-bus system: slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid
ax=gca;
y_lims=ax.YLim;
text(.5,y_lims(2)-.0025, loss_label);


subplot(212)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Qsum);
title('14-bus system: total generated reactive power and power loss')
legend('P_{loss}','Q_{gen}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Total generated reactive power');
hLine2.LineStyle='--';
grid

figure(6)
[hAx6, ~, hLine6]=plotyy(x1,loss,[x2,x2,x2,x2,x2],[Q1, Q2, Q3, Q6, Q8]);
title('Generator reactive powers and power loss')
legend('P_{loss}','Q_{slack}','Q_{g2}','Q_{g3}','Q_{g6}','Q_{g8}')
xlabel('Iteration number')
ylabel(hAx6(1),'Real power loss');
ylabel(hAx6(2),'Generator reactive power outputs');
hLine6(1).LineStyle='--';
hLine6(2).LineStyle=':';
hLine6(2).Color='r';
hLine6(3).LineStyle='-.';
grid



% figure(6)
% [hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
% title('Fig.6: Slack-bus reactive power and power loss')
% legend('P_{loss}','Q_{slack}')
% xlabel('Iteration number')
% ylabel(hAx(1),'Real power loss');
```

```
% ylabel(hAx(2),'Slack-bus reactive power');
% hLine2.LineStyle='--';
% grid

figure(7)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q2);
title('Bus-2 generator reactive power and power loss')
legend('P_{loss}','Q_{g2}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-2 generator reactive power');
hLine2.LineStyle='--';
grid

figure(8)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q3);
title('Bus-3 generator reactive power and power loss')
legend('P_{loss}','Q_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-3 generator reactive power');
hLine2.LineStyle='--';
grid

figure(9)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q6);
title('Bus-6 generator reactive power and power loss')
legend('P_{loss}','Q_{g6}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-6 generator reactive power');
hLine2.LineStyle='--';
grid

figure(10)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q8);
title('Bus-6 generator reactive power and power loss')
legend('P_{loss}','Q_{g8}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-8 generator reactive power');
hLine2.LineStyle='--';
grid

figure(11)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Qsum);
title('Total generated reactive power and power loss')
legend('P_{loss}','Q_{gen}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Total generated reactive power');
hLine2.LineStyle='--';
grid
```

```
figure(12)
plot(x2,Q1,x2,Q2,'r--',x2,Q3,'m-.',x2,Q6,'c-+',x2,Q8,'k-*')
title('Generator reactive powers')
legend('Q_{slack}','Q_{g2}','Q_{g3}','Q_{g6}','Q_{g8}')
grid

figure(13)
subplot(311)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Vg1);
title('Slack-bus reactive power and voltage magnitude')
legend('Q_{slack}','V_{g1}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Slack-bus voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(312)
[hAx, ~, hLine2]=plotyy(x2,Q2,x2,Vg2);
title('Bus-2 generator reactive power and voltage magnitude')
legend('Q_{g2}','V_{g2}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-2 generator reactive power');
ylabel(hAx(2),'Bus-2 voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(313)
[hAx, ~, hLine2]=plotyy(x2,Q3,x2,Vg3);
title('Bus-3 generator reactive power and voltage magnitude')
legend('Q_{g3}','V_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-3 generator reactive power');
ylabel(hAx(2),'Bus-3 voltage magnitude');
hLine2.LineStyle='--';
grid

figure(14)
subplot(211)
[hAx, ~, hLine2]=plotyy(x2,Q6,x2,Vg6);
title('Bus-6 generator reactive power and voltage magnitude')
legend('Q_{g6}','V_{g6}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-6 generator reactive power');
ylabel(hAx(2),'Bus-6 voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(212)
[hAx, ~, hLine2]=plotyy(x2,Q8,x2,Vg8);
title('Bus-8 generator reactive power and voltage magnitude')
legend('Q_{g8}','V_{g8}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-8 generator reactive power');
```

```
ylabel(hAx(2),'Bus-8 voltage magnitude');
hLine2.LineStyle='--';
grid
```

B.20 Function that implements the primal-dual interior-point method-based Volt/VAR optimization (PDIPM-VVO) for the 30-bus system, incorporating the Newton-Raphson load flow computation

```
function [X,adx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu,fval, cond,
output]=ipm_30bus(Func,x0,s0,mu0, sigma, h, dF, J, Vgref, maxIter)
global E F PQs Qsum Vgens
%zeta=.99995;
zeta=.9995;

% Added computation of slack-bus active and generator reactive powers

[V, output]=NR_load_flow(dF, J, x0, Vgref);
E=real(V);
F=imag(V);

% Initialize slack-bus active, and generator reactive power outputs
% matrix to empty matrix:
PQs=[];

% Compute slack-bus active, and generator reactive power outputs
compute_PQ();

% Vector to hold generator bus voltages:
Vgens=[];

% Extract generator bus voltages;
get_Vgen();

Vinit=output.V(:,1);

[initial_loss, ~, ~]=loss_func_30bus();

% update control vector x from generator voltage vector components E, F:
x=update_control_vector();

s=s0;
mu=mu0;
niq=length(h(x));
e=ones(niq,1);

S=diag(s);
Si=inv(S);
lam_i=mu*Si*e;
Lam_i=diag(lam_i);

eps_1=1e-3;
eps_2=1e-3;
```

```
eps_mu=1e-3;

i=1;

[f, h, dh, gL, g2L]=Func(x, lam_i);

c1=(max(h)<0 || norm(h, inf)<=1e-3);
c2=(norm(gL,inf)/(1+norm(x)+norm(lam_i)));
c3=((s'*lam_i)/(1+norm(x, inf)));
c4=(abs(f)/(1+abs(f)));
c5=mu;
c6=output.exit_flag;

converged=c1&&c2<=eps_1&&c3<=eps_2&&c4<=eps_2&&c5<=eps_mu&&c6;

Vslack=[E(1) F(1)];

fprev=f;
fval=[];
fval=[fval; f];
X=[];
X=[X x];
adx=[];
iter=[];
iter=[iter; i];
Loss=[];
Loss=[Loss; initial_loss];

while (~converged && i<=maxIter)

    A=g2L+dh'*Si*Lam_i*dh;
    b=gL+dh'*Si*(mu*e+Lam_i*h);
    dx=-A\b;
    ds=-h-s-dh*dx;
    dlam_i=Si*(-S*lam_i+mu*e-Lam_i*ds);

k_ds=find(ds<0);
if(k_ds)
    alpha_p=min(1, zeta*min(-s(k_ds)./ds(k_ds)));
else
    alpha_p=1;
end
k_dlam_i=find(dlam_i<0);
if(k_dlam_i)
    alpha_d=min(1, zeta*min(-lam_i(k_dlam_i)./dlam_i(k_dlam_i)));
else
    alpha_d=1;
end
alpha=min(alpha_p,alpha_d);
%alpha_p=1;
if (alpha_p>.1 && alpha_d>.1)
    alpha_p=alpha_d;
end
```

```
x=x+alpha_p*dx;
s=s+alpha_p*ds;
lam_i=lam_i+alpha_d*dlam_i;
mu=sigma*(s'*lam_i)/2/niq;
S=diag(s);
Si=inv(S);
Lam_i=diag(lam_i);
i=i+1;

% Define updated initial voltage vector for the NR load flow algorithm:
x0=define_updated_x0(x);

% Define generator voltage reference vector for the NR load flow algorithm:
Vgref=updated_Vgref(x);

[V, output]=NR_load_flow(dF, J, x0, Vgref);
E=real(V);
F=imag(V);

% Compute slack-bus active, and generator reactive power outputs
compute_PQ();

% Extract generator bus voltages;
get_Vgen();

% Vslack:
Vslack=[Vslack; E(1) F(1)];

[x(1), x(2), x(3), x(4), x(5), x(6), x(7), x(8), x(9), x(10), x(11)]=...
    deal(E(1), E(2), F(2), E(5), F(5), E(8), F(8), E(11), F(11), E(13), F(13));

[f, h, dh, gL, g2L]=Func(x, lam_i);

loss=f;

fval=[fval;f];
X=[X x];
adx=[adx alpha_p*dx];
iter=[iter; i];
Loss=[Loss; loss];

c1=(max(h)<0 || norm(h, inf)<=1e-3);
c2=norm(gL,inf)/(1+norm(x)+norm(lam_i));
c3=s'*lam_i/(1+norm(x, inf));
c4=abs(f-fprev)/(1+abs(f));
c5=mu;
c6=output.exit_flag;
converged=c1&&c2<=eps_1&&c3<=eps_1&&c4<=eps_2&&c5<=eps_mu&&c6;
fprev=f;
cond.num_iterations=i;

end

cond.primal_feasibility=max(h);
```

```
cond.grad_condition=c2;
cond.comp_condition=c3;
cond.objective_condition=c4;
cond.barrier_condition=c5;
cond.A=A;
cond.b=b;
cond.h=h;
cond.S=S;
cond.Lami=Lam_i;
cond.initial_loss=initial_loss;
cond.loss=Loss;
cond.iter=iter;
cond.V1=Vslack;
cond.Vinit=Vinit;
cond.PQs=PQs;
cond.Qsum=Qsum;
cond.Vgen=Vgens;
```

B.21 Function that computes the objective function, its gradient and Hessian for the 30-bus system

```
function [f, df, d2f]=loss_func_30bus()
global E F G

[f, ~]=loss_func();

df=2*[G(1,2)*(E(1)-E(2))+G(1,3)*(E(1)-E(3));...
    G(2,4)*(E(2)-E(4))-G(1,2)*(E(1)-E(2))+G(2,5)*(E(2)-E(5))+G(2,6)*(E(2)-E(6));...
    G(2,4)*(F(2)-F(4))-G(1,2)*(F(1)-F(2))+G(2,5)*(F(2)-F(5))+G(2,6)*(F(2)-F(6));...
    G(5,7)*(E(5)-E(7))-G(2,5)*(E(2)-E(5));...
    G(5,7)*(F(5)-F(7))-G(2,5)*(F(2)-F(5));...
    G(8,28)*(E(8)-E(28))-G(6,8)*(E(6)-E(8));...
    G(8,28)*(F(8)-F(28))-G(6,8)*(F(6)-F(8));...
    -G(9,11)*(E(9)-E(11));...
    -G(9,11)*(F(9)-F(11));...
    -G(12,13)*(E(12)-E(13));...
    -G(12,13)*(F(12)-F(13))];

d2f=zeros(length(df));
d2f(1,1)=2*(G(1,2)+G(1,3));
d2f(1,2)=-2*G(1,2);
d2f(2,1)=d2f(1,2);
d2f(2,2)=2*(G(1,2)+G(2,4)+G(2,5)+G(2,6));
d2f(2,4)=-2*G(2,5);
d2f(3,3)=2*(G(1,2)+G(2,4)+G(2,5)+G(2,6));
d2f(3,5)=-2*G(2,5);
d2f(4,2)=d2f(2,4);
d2f(4,4)=2*(G(2,5)+G(5,7));
d2f(5,3)=d2f(3,5);
d2f(5,5)=2*(G(2,5)+G(5,7));
d2f(6,6)=2*(G(6,8)+G(8,28));
d2f(7,7)=2*(G(6,8)+G(8,28));
```

```
d2f(8,8)=2*G(9,11);
d2f(9,9)=2*G(9,11);
d2f(10,10)=2*G(12,13);
d2f(11,11)=2*G(12,13);
```

B.22 Function that computes the constraint functions, their Jacobian and Hessian for the 30-bus system

```
function [h, dh, dht_lam_i, d2ht_lam_i]=h_gradh_hessh_30bus(x, lam_i)
global E F nbus

Vmsq=.95^2;
VMsq=1.1^2;

[E(1), E(2), F(2), E(5), F(5), E(8), F(8), E(11), F(11), E(13), F(13)]=...
    deal(x(1), x(2), x(3), x(4), x(5), x(6), x(7), x(8), x(9), x(10), x(11));


% Define inequality constraints h(x):
h=zeros(2*nbus,1);
k=1;
for i=1:nbus
    h(k)=-(E(i)^2+F(i)^2)+Vmsq;
    h(k+1)=E(i)^2+F(i)^2-VMsq;
    k=k+2;
end

% Define the Jacobian of the inequality constraints dh(x):
% (transposed and multiplied with the Lagrangian multiplier vector lambda_i):
dh=zeros(2*nbus, length(x));
[dh(1,1), dh(2,1), dh(3,2), dh(3,3), dh(4,2), dh(4,3), dh(9,4), dh(9,5),...
    dh(10,4), dh(10,5), dh(15,6), dh(15,7), dh(16,6), dh(16,7), dh(21,8),...
    dh(21,9), dh(22,8), dh(22,9), dh(25,10), dh(25,11), dh(26,10), dh(26,11)]=...
    deal(-2*E(1), 2*E(1), -2*E(2), -2*F(2), 2*E(2), 2*F(2), -2*E(5), -2*F(5),...
    2*E(5), 2*F(5), -2*E(8), -2*F(8), 2*E(8), 2*F(8), -2*E(11), -2*F(11),...
    2*E(11), 2*F(11), -2*E(13), -2*F(13), 2*E(13), 2*F(13));

dht_lam_i=2*[E(1)*(lam_i(2)-lam_i(1));...
    E(2)*(lam_i(4)-lam_i(3));...
    F(2)*(lam_i(4)-lam_i(3));...
    E(5)*(lam_i(10)-lam_i(9));...
    F(5)*(lam_i(10)-lam_i(9));...
    E(8)*(lam_i(16)-lam_i(15));...
    F(8)*(lam_i(16)-lam_i(15));...
    E(11)*(lam_i(22)-lam_i(21));...
    F(11)*(lam_i(22)-lam_i(21));...
    E(13)*(lam_i(26)-lam_i(25));...
    F(13)*(lam_i(26)-lam_i(25))];
d2ht_lam_i_diag=2*[lam_i(2)-lam_i(1), lam_i(4)-lam_i(3), lam_i(4)-lam_i(3),...
    lam_i(10)-lam_i(9), lam_i(10)-lam_i(9), lam_i(16)-lam_i(15), lam_i(16)-
lam_i(15),...
```

```
    lam_i(22)-lam_i(21), lam_i(22)-lam_i(21), lam_i(26)-lam_i(25), lam_i(26)-
lam_i(25)];
d2ht_lam_i=diag(d2ht_lam_i_diag);
```

B.23 Function that computes the Jacobian and Hessian of the Lagrangian of the VVO problem for the 30-bus system

```
function [f, h, dh, gL, g2L]=f_30bus(x, lam_i)

% Define objective function, its gradient and hessian [f, df, d2f]:

[f, df, d2f]=loss_func_30bus();

%[~, f, df, d2f, ~, ~]=loss_func_sym_30bus();

% Define inequality constraints, the Jacobian and hessian [h, dht_lam_i,
d2ht_lam_i]:
[h, dh, dht_lam_i, d2ht_lam_i]=h_gradh_hessh_30bus(x, lam_i);

% Define gradient and Hessian of Lagrangian, gL, g2L:
gL=df+dht_lam_i;
g2L=-d2f+d2ht_lam_i;
```

B.24 MATLAB script that runs the PDIPM-VVO algorithm for the 30-bus system

```
clear

close all
clc

% File name: thirty_bus_system_pdipm_vvo_test.m

% List of functions needed to run this program:
% (further details are provided for each of these functions
%  where they are first called in the program):

% 1.  [G, B, Cf]=compute_Ybus(from_bus, to_bus, r, x)
% 2.  [R, X, Cf]=computeRX(from, to, r, x)
% 3.  x0=define_x0()
% 4.  Vgref=Vgref_0()
% 5.  dF=dF(x, Vgref)
% 6.  J=jacobian(x)
% 7.  [V, output]=NR_load_flow(@dF, @jacobian, x0, Vgref)
% 8.  h=h_30bus(x)
% 9.  [f, df, d2f]=loss_func_30bus()
% 10. [h, dh, dht_lam_i, d2ht_lam_i]=h_gradh_hessh_30bus(x, lam_i)
% 11. [f, h, dh, gL, g2L]=f_30bus(x, lam_i)
% 12. [X,adx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu,fval, cond, output]=...
%         ipm_30bus(@f_30bus,x0,s0,mu0, sigma, @h_30bus, @dF, @jacobian,...
```

```
%         Vgref, maxIter)
% 13. x=update_control_vector()
% 14. update_generator_voltages(x)
% 15. Vgref=updated_Vgref(x)
%
% Bus Data:
%
% Volt/VAR optimization for the IEEE 30-bus system:
% Number of buses       : 30;
% Number of lines       : 41;
% Number of generators  : 6;
% Number of loads       : 21

global bus_data Cf E F G B bus_type nbus

% bus_data is matrix in which each row applies to a bus, and specifies:
% (1) bus type (1=slack bus, 2=PV bus, 3=PQ bus)
% (2) Real component of bus voltage (E)
% (3) Imaginary component of bus voltage (F)
% (4) Generated real power (Pg)
% (5) Generated reactive power (Qg)
% (6) Real power demand (Pd)
% (7) Reactive power demand (Qd)

% Cf is the bus connectivity matrix; Cf(i,j)=1 when buses i and j are
% connected, zero otherwise

% E and F are vectors containing real and imaginary components of the
% bus voltages (in rectangular coordinates)

% G and B are conductance and susceptance matrices respectively;
% i.e. Y=G+jB, where Y is the bus admittance matrix

% bus_type is simply the first column of the bus_data matrix

% nbus is the number of buses in the system

% Values are all in per-unit

%=====================================================================
%     Bus type    V_real     V_imag  Pg      Qg     Pd          Qd
bus_data=[1         1          0     0       0      0          0  ;...
          2         1          0    .4882    0     .217       .127;...
          3         1          0     0       0     .024       .012;...
          3         1          0     0       0     .076       .016;...
          2         1          0    .2151    0     .942       .19 ;...
          3         1          0     0       0      0          0  ;...
          3         1          0     0       0     .228       .109;...
          2         1          0    .2215    0      .3        .3  ;...
          3         1          0     0       0      0          0  ;...
          3         1          0     0       0     .058       .02 ;...
          2         1          0    .1214    0      0          0  ;...
          3         1          0     0       0     .112       .075;...
          2         1          0    .12      0      0          0  ;...
```

```
          3         1         0     0       0    .062       .016;...
          3         1         0     0       0    .082       .025;...
          3         1         0     0       0    .035       .016;...
          3         1         0     0       0    .09        .058;...
          3         1         0     0       0    .032       .009;...
          3         1         0     0       0    .095       .034;...
          3         1         0     0       0    .022       .007;...
          3         1         0     0       0    .175       .112;...
          3         1         0     0       0      0         0  ;...
          3         1         0     0       0    .032       .016;...
          3         1         0     0       0    .087       .067;...
          3         1         0     0       0      0         0  ;...
          3         1         0     0       0    .035       .023;...
          3         1         0     0       0     0         0  ;...
          3         1         0     0       0     0         0  ;...
          3         1         0     0       0    .024       .009;...
          3         1         0     0       0    .106       .019];
%=====================================================================

bus_type=bus_data(:,1);
nbus=length(bus_type);

% Line Data:
% Corresponding elements of the from_bus and to_bus vectors are the bus
% pairs of connected buses; a line or branch (i,j) exists between bus
% pairs (from_bus(i), to_bus(j));
% r and x are vectors of line resistance and reactance values,
% corresponding to the lines specified by (from_bus(i), to_bus(j))

from_bus=[1, 1, 2, 2, 2, 3, 4, 4, 5, 6, 6, 6, 6, 6, 8, 9, 9,...
    10, 10, 10, 10, 12, 12, 12, 12, 14, 15, 15, 16, 18, 19, 21,...
    22, 23, 24, 25, 25, 27, 27, 28, 29];

to_bus=[2, 3, 4, 5, 6, 4, 6, 12, 7, 7, 8, 9, 10, 28, 28, 10,...
    11, 17, 20, 21, 22, 13, 14, 15, 16, 15, 18, 23, 17, 19, 20,...
    22, 24, 24, 25, 26, 27, 29, 30, 27, 30];

r=[.0192, .0452, .057, .0472, .0581, .0132, .0119, 0, .046, .0267,...
    .012, 0, 0, .0169, .0636, 0, 0, .0324, .0936, .0348, .0727, 0,...
    .1231, .0662, .0945, .221, .107, .1, .0824, .0639, .034, .0116,...
    .115, .132, .1885, .2544, .1093, .2198, .3202, 0, .2399];

x=[.0575, .1852, .1737, .1983, .1763, .0379, .0414, .256, .116, .082,...
    .042, .208, .556, .0599, .2, .208, .11, .0845, .209, .0749, .1499,...
    .14, .2559, .1304, .1987, .1997, .2185, .202, .1932, .1292, .068,...
    .0236, .179, .27, .3292, .38, .2087, .4153, .6027, .396, .4533];

% Compute the admittance (in the form G+jB) and bus connectivity (Cf)
% matrices
% Function compute_Ybus() calls function computeRX(), which computes
% the resistance (R) and reactance (X) vectors, needed by the
% function compute_Ybus(), as well as matrix Cf

[G, B, Cf]=compute_Ybus(from_bus, to_bus, r, x);
```

```
% Define some parameters:

% Extract the vectors of real and imaginary bus voltage components
% from the bus_data matrix

E=bus_data(:,2);
F=bus_data(:,3);

% Define initial input to Newton-Raphson load flow algorithm
% (initial bus voltages,in rectangular form):

x0=define_x0();

% Define the reference voltage vector for the generator voltages;
% this is required for the Volt/VAR optimization problem when running
% the load flow algorithm at each Newton method iteration

Vgref=Vgref_0();

% Perform Newton-Raphson load flow
% Newton-Raphson algorithm implemented in rectangular coordinates of
% bus voltages; assumes that bus 1 is the (only) slack bus. Calls functions
% dF() of residues (of the real and reactive power/voltage balance
% equations), as well as jacobian(), which computes the Jacobian of the
% residues, needed to compute the Newton step once the algorithm
% (hopefully) converges, the bus voltages are output in polar form
% (i.e. magnitude and phase angle)

% tic
% [~, output]=NR_load_flow(@dF, @jacobian, x0, Vgref);
% toc
%
% v=[output.V(:,1) output.V(:,2)]
%
% % Computes system losses after the load flow algorithm has terminated
%
% [losses, ~, ~]=loss_func_30bus()

% Perform Volt/VAR optimization:
% Interior-Point Method (IPM)-based Volt/VAR optimization (VVO); applies
% the Newton method to compute the search direction for the primal-dual
% system of the VVO problem derived on the basis of the perturbed KKT
% (first-order) optimality conditions.
%
% Computation of the Newton step requires calculating the first- and
% second-order partial derivatives of the objective and constraint
% functions. The IPM algorithm only considers the inequality constraints
% (in this implementation only the bus voltage magnitude constraints),
% the equality constraints (real and reactive power balance equations)
% are handled by the (Newton-Raphson) load flow algorithm. Therefore, at
% each iteration of the Newton step of the IPM algorithm, the load flow
% algorithm is executed once the primal and dual variables have been
% updated.
```

```
%
% The IPM algorithm needs the following functions to execute:
%
% 1. f_30bus(), which computes the gradient and hessian of the Lagrangian
%      function of the VVO problem, which are required to compute the
%      Newton step;
% 2.  loss_func_30bus(), which is called by f_30bus(), to compute the
%     objective, its gradient and hessian;
% 3.  h_gradh_hessh_30bus(), which is also called by f_30bus, to compute
%     the constraint functions, the Jacobian and hessian thereof as well;
% 4.  h_30bus(), used only once at the beginning of the IPM algorithm
%     to determine the number of constraint function;
% 5.  dF(), which computes the residues needed by the Newton-Raphson
%     load flow algorithm;
% 6.  jacobian(), which computes the Jacobian of the residues, also needed
%     by the Newton-Raphson load flow algorithm;
% 7.  A number of utility functions called by the functions stated above,
%     including update_generator_voltages(), update_control_vector(),
%     define_updated_x0(), updated_Vgref();
%
% The IPM algorithm also requires a number of parameters, such as the
% barrier parameter (mu), the centering parameter (zeta), and the
% choice of initial primal and dual variables.

% Initialize some input parameters:

h0=h_30bus(x0);
s0=abs(h0);
s0(s0==0)=.01;
mu0=10;
sigma=.15;
maxIt=13;

% Run the IPM algorithm on the VVO problem:

tic
[X,adx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu,fval, cond, output]=...
    ipm_30bus6(@f_30bus,x0,s0,mu0, sigma, @h_30bus, @dF, @jacobian, Vgref, maxIt);
toc

% Output some computation results:

X
mu
cond
output
V=output.V

% Compute the loss reduction:

initial_loss_pu=-cond.loss(1)
final_loss_pu=-cond.loss(end)
loss_reduction_percentage=100*(cond.loss(1)-cond.loss(end))/cond.loss(1)
```

```
% Plot the loss reduction vs. the iteration number:

figure(1), plot(cond.V1(:,1)), hold on, plot(cond.V1(:,2),'r'),hold on,
plot(cond.V1(:,2)./cond.V1(:,1),'k--')
legend('E','F','F/E');
title('Slack-bus voltage (real(E) and imaginary(F) components)');
xlabel('Iteration number')
ylabel('V_{slack-pu}')
V1_mag=sqrt(cond.V1(:,1).^2+cond.V1(:,2).^2);
V1_angle=180/pi*atan(cond.V1(:,2)./cond.V1(:,1));
V1=[V1_mag V1_angle]

figure(2)
Vinit=cond.Vinit;
Vfinal=V(:,1);
plot(Vinit,'b-s'), hold on, plot(Vfinal,'r-s'),legend('V_{Initial}','V_{Final}')
grid
title('Voltage profile, final compared with initial');
xlabel('Bus number')
ylabel('V_{bus-pu}')

% xlswrite('thirty_bus_system_voltage.xlsx', [Vinit Vfinal])

figure(3)
plot(cond.iter,-cond.loss, 'r',cond.iter,-cond.loss, 'b*');
grid
title('IEEE 30-bus system real power loss vs. iteration number', 'FontSize', 10.0)
xlabel('Iteration number')
ylabel('P_{loss} (p.u.)')
loss_label=['Percentage loss reduction = ' num2str(loss_reduction_percentage) '%'];

ax=gca;
y_lims=ax.YLim;
text(.5,y_lims(2)-.0005, loss_label);

loss=-cond.loss;
% Adding 2.2e-3 to P1 makes slack-bus active power
% and power loss coincide; this (value of 2.2e-3)
% seems to only represent a discrepancy due to
% differences in scale of the two quantities
P1=2.2e-3+cond.PQs(1,:)';
Q1=cond.PQs(2,:)';
Q2=cond.PQs(3,:)';
Q5=cond.PQs(4,:)';
Q8=cond.PQs(5,:)';
Q11=cond.PQs(6,:)';
Q13=cond.PQs(7,:)';
Qsum=cond.Qsum';
Vg1=cond.Vgen(1,:)';
Vg2=cond.Vgen(2,:)';
Vg5=cond.Vgen(3,:)';
Vg8=cond.Vgen(4,:)';
Vg11=cond.Vgen(5,:)';
Vg13=cond.Vgen(6,:)';
```

```
x1=[0:length(loss)-1]';
x2=[0:length(P1)-1]';

figure(4)
subplot(221)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('Slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid

subplot(222)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
grid

subplot(223)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q5);
title('Bus-5 generator reactive power and power loss')
legend('P_{loss}','Q_{g5}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-5 generator reactive power');
hLine2.LineStyle='--';
grid

subplot(224)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Q5);
title('Generator reactive powers')
legend('Q_{slack}','Q_{g5}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Generator reactive powers');
hLine2.LineStyle='--';
grid

figure(5)
subplot(211)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('30-bus system: slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
```

```
grid
ax=gca;
y_lims=ax.YLim;
text(.5,y_lims(2)-.0005, loss_label);

subplot(212)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Qsum);
title('30-bus system: total generated reactive power and power loss')
legend('P_{loss}','Q_{gen}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Total generated reactive power');
hLine2.LineStyle='--';
grid

figure(6)
[hAx6, ~, hLine6]=plotyy(x1,loss,[x2,x2,x2,x2,x2 x2],[Q1, Q2, Q5, Q8, Q11, Q13]);
title('Generator reactive powers and power loss')
legend('P_{loss}','Q_{slack}','Q_{g2}','Q_{g5}','Q_{g8}','Q_{g11}','Q_{g13}')
xlabel('Iteration number')
ylabel(hAx6(1),'Real power loss');
ylabel(hAx6(2),'Generator reactive power outputs');
hLine6(1).LineStyle='--';
hLine6(2).LineStyle=':';
hLine6(2).Color='r';
hLine6(3).LineStyle='-.';
grid

figure(14)
subplot(311)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Vg1);
title('Slack-bus reactive power and voltage magnitude')
legend('Q_{slack}','V_{g1}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Slack-bus voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(312)
[hAx, ~, hLine2]=plotyy(x2,Q2,x2,Vg2);
title('Bus-2 generator reactive power and voltage magnitude')
legend('Q_{g2}','V_{g2}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-2 generator reactive power');
ylabel(hAx(2),'Bus-2 voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(313)
[hAx, ~, hLine2]=plotyy(x2,Q5,x2,Vg5);
title('Bus-5 generator reactive power and voltage magnitude')
legend('Q_{g5}','V_{g5}')
xlabel('Iteration number')
```

```
ylabel(hAx(1),'Bus-5 generator reactive power');
ylabel(hAx(2),'Bus-5 voltage magnitude');
hLine2.LineStyle='--';
grid

figure(15)
subplot(311)
[hAx, ~, hLine2]=plotyy(x2,Q8,x2,Vg8);
title('Bus-8 generator reactive power and voltage magnitude')
legend('Q_{g8}','V_{g8}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-8 generator reactive power');
ylabel(hAx(2),'Bus-8 voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(312)
[hAx, ~, hLine2]=plotyy(x2,Q11,x2,Vg11);
title('Bus-11 generator reactive power and voltage magnitude')
legend('Q_{g11}','V_{g11}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-11 generator reactive power');
ylabel(hAx(2),'Bus-11 voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(313)
[hAx, ~, hLine2]=plotyy(x2,Q13,x2,Vg13);
title('Bus-13 generator reactive power and voltage magnitude')
legend('Q_{g13}','V_{g13}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-13 generator reactive power');
ylabel(hAx(2),'Bus-13 voltage magnitude');
hLine2.LineStyle='--';
grid
```

B.25 Function that implements the primal-dual interior-point method-based Volt/VAR optimization (PDIPM-VVO) for the 118-bus system, incorporating the Newton-Raphson load flow computation

```
function [X,adx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu,fval, cond,
output]=ipm_118bus(Func,x0,s0,mu0, sigma, h, dF, J, Vgref, maxIter, V1_angle_ratio)
global E F  PQs Qsum

%zeta=.9995;
zeta=.99995;

% Added computation of slack-bus active and generator reactive powers

% Perform initial load flow:
[V, output]=NR_load_flow(dF, J, x0, Vgref);
E=real(V);
```

```
F=imag(V);

% Initialize slack-bus active, and generator reactive power matrix to
% empty array:

PQs=[];

% Compute slack-bus active, and generator reactive power outputs
compute_PQ();

% Initial load flow voltage results:
Vinit=output.V(:,1);

% Initial system real power losses:
[initial_loss, ~]=loss_func();

% update control vector x from generator voltage vector components E, F:
x=update_control_vector();

s=s0;
mu=mu0;
niq=length(h(x));
e=ones(niq,1);

S=diag(s);
Si=inv(S);
lam_i=mu*Si*e;
Lam_i=diag(lam_i);

eps_1=1e-3;
eps_2=1e-3;
eps_mu=1e-3;

i=1;

[f, h, dh, gL, g2L]=Func(x, lam_i);

c1=(max(h)<0 || norm(h, inf)<=1e-3);
c2=(norm(gL,inf)/(1+norm(x)+norm(lam_i)));
c3=((s'*lam_i)/(1+norm(x, inf)));
c4=(abs(f)/(1+abs(f)));
c5=mu;
c6=output.exit_flag;

converged=c1&&c2<=eps_1&&c3<=eps_2&&c4<=eps_2&&c5<=eps_mu&&c6;

Vslack=[E(1) F(1)];

fprev=f;
fval=[];
fval=[fval; f];
X=[];
X=[X x];
adx=[];
```

```
iter=[];
iter=[iter; i];
Loss=[];
Loss=[Loss; initial_loss];

while (~converged && i<=maxIter)

    A=g2L+dh'*Si*Lam_i*dh;
    b=gL+dh'*Si*(mu*e+Lam_i*h);
    dx=-A\b;
    ds=-h-s-dh*dx;
    dlam_i=Si*(-S*lam_i+mu*e-Lam_i*ds);

k_ds=find(ds<0);
if(k_ds)
    alpha_p=min(1, zeta*min(-s(k_ds)./ds(k_ds)));
else
    alpha_p=1;
end
k_dlam_i=find(dlam_i<0);
if(k_dlam_i)
    alpha_d=min(1, zeta*min(-lam_i(k_dlam_i)./dlam_i(k_dlam_i)));
else
    alpha_d=1;
end
alpha=min(alpha_p,alpha_d);
%alpha_p=1;
if (alpha_p>.1 && alpha_d>.1)
    alpha_p=alpha_d;
end

x=x+alpha_p*dx;
s=s+alpha_p*ds;
lam_i=lam_i+alpha_d*dlam_i;
mu=sigma*(s'*lam_i)/2/niq;
S=diag(s);
Si=inv(S);
Lam_i=diag(lam_i);
i=i+1;

% Define updated initial voltage vector for the NR load flow algorithm:
x0=define_updated_x0(x);

% To maintain slack-bus voltage angle:
F(1)=V1_angle_ratio*x(1);

% Define generator voltage reference vector for the NR load flow algorithm
%Vgref=Vgref_0();
Vgref=updated_Vgref1(x);

% Run the NR load flow algorithm:
[V, output]=NR_load_flow(dF, J, x0, Vgref);
E=real(V);
F=imag(V);
```

```
% Compute slack-bus active, and generator reactive power outputs
compute_PQ();

% Vslack:
Vslack=[Vslack; E(1) F(1)];

% update control vector x from generator voltage vector components E, F:
x=update_control_vector();

[f, h, dh, gL, g2L]=Func(x, lam_i);

loss=f;

fval=[fval;f];
X=[X x];
adx=[adx alpha_p*dx];
iter=[iter; i];
Loss=[Loss; loss];

c1=(max(h)<0 || norm(h, inf)<=1e-3);
c2=norm(gL,inf)/(1+norm(x)+norm(lam_i));
c3=s'*lam_i/(1+norm(x, inf));
c4=abs(f-fprev)/(1+abs(f));
c5=mu;
c6=output.exit_flag;
converged=c1&&c2<=eps_1&&c3<=eps_1&&c4<=eps_2&&c5<=eps_mu&&c6;
fprev=f;
cond.num_iterations=i;

end

cond.primal_feasibility=max(h);
cond.grad_condition=c2;
cond.comp_condition=c3;
cond.objective_condition=c4;
cond.barrier_condition=c5;
cond.A=A;
cond.b=b;
cond.h=h;
cond.S=S;
cond.Lami=Lam_i;
cond.initial_loss=initial_loss;
cond.loss=Loss;
cond.iter=iter;
cond.V1=Vslack;
cond.Vinit=Vinit;cond.PQs=PQs;
cond.Qsum=Qsum;
```

B.26 Function that computes the gradient of the objective function for the 118-bus system

```
function df=df_118bus()
global E F G

df=[G(1,2)*(2*E(1)-2*E(2))+G(1,3)*(2*E(1)-2*E(3));...
    G(4,5)*(2*E(4)-2*E(5))+G(4,11)*(2*E(4)-2*E(11));...
    G(4,5)*(2*F(4)-2*F(5))+G(4,11)*(2*F(4)-2*F(11));...
    G(6,7)*(2*E(6)-2*E(7))-G(5,6)*(2*E(5)-2*E(6));...
    G(6,7)*(2*F(6)-2*F(7))-G(5,6)*(2*F(5)-2*F(6));...
    G(8,9)*(2*E(8)-2*E(9))-G(5,8)*(2*E(5)-2*E(8))+...
    G(8,30)*(2*E(8)-2*E(30));...
    G(8,9)*(2*F(8)-2*F(9))-G(5,8)*(2*F(5)-2*F(8))+...
    G(8,30)*(2*F(8)-2*F(30));...
    -G(9,10)*(2*E(9)-2*E(10));...
    -G(9,10)*(2*F(9)-2*F(10));...
    G(12,14)*(2*E(12)-2*E(14))-G(11,12)*(2*E(11)-2*E(12))+...
    G(12,16)*(2*E(12)-2*E(16))+G(12,117)*(2*E(12)-2*E(117))-...
    G(2,12)*(2*E(2)-2*E(12))-G(3,12)*(2*E(3)-2*E(12))-...
    G(7,12)*(2*E(7)-2*E(12));...
    G(12,14)*(2*F(12)-2*F(14))-G(11,12)*(2*F(11)-2*F(12))+...
    G(12,16)*(2*F(12)-2*F(16))+G(12,117)*(2*F(12)-2*F(117))-...
    G(2,12)*(2*F(2)-2*F(12))-G(3,12)*(2*F(3)-2*F(12))-...
    G(7,12)*(2*F(7)-2*F(12));...
    G(15,17)*(2*E(15)-2*E(17))-G(14,15)*(2*E(14)-2*E(15))-...
    G(13,15)*(2*E(13)-2*E(15))+G(15,19)*(2*E(15)-2*E(19))+...
    G(15,33)*(2*E(15)-2*E(33));...
    G(15,17)*(2*F(15)-2*F(17))-G(14,15)*(2*F(14)-2*F(15))-...
    G(13,15)*(2*F(13)-2*F(15))+G(15,19)*(2*F(15)-2*F(19))+...
    G(15,33)*(2*F(15)-2*F(33));...
    G(18,19)*(2*E(18)-2*E(19))-G(17,18)*(2*E(17)-2*E(18));...
    G(18,19)*(2*F(18)-2*F(19))-G(17,18)*(2*F(17)-2*F(18));...
    G(19,20)*(2*E(19)-2*E(20))-G(18,19)*(2*E(18)-2*E(19)) - ...
    G(15,19)*(2*E(15)-2*E(19))+G(19,34)*(2*E(19)-2*E(34));...
    G(19,20)*(2*F(19)-2*F(20))-G(18,19)*(2*F(18)-2*F(19)) - ...
    G(15,19)*(2*F(15)-2*F(19))+G(19,34)*(2*F(19)-2*F(34));...
    G(24,70)*(2*E(24)-2*E(70))-G(23,24)*(2*E(23)-2*E(24))+...
    G(24,72)*(2*E(24)-2*E(72));...
    G(24,70)*(2*F(24)-2*F(70))-G(23,24)*(2*F(23)-2*F(24))+...
    G(24,72)*(2*F(24)-2*F(72));...
    G(25,26)*(2*E(25)-2*E(26))-G(23,25)*(2*E(23)-2*E(25))+...
    G(25,27)*(2*E(25)-2*E(27));...
    G(25,26)*(2*F(25)-2*F(26))-G(23,25)*(2*F(23)-2*F(25))+...
    G(25,27)*(2*F(25)-2*F(27));...
    G(26,30)*(2*E(26)-2*E(30))-G(25,26)*(2*E(25)-2*E(26));...
    G(26,30)*(2*F(26)-2*F(30))-G(25,26)*(2*F(25)-2*F(26));...
    G(27,28)*(2*E(27)-2*E(28))-G(25,27)*(2*E(25)-2*E(27))+...
    G(27,32)*(2*E(27)-2*E(32))+...
    G(27,115)*(2*E(27)-2*E(115));...
    G(27,28)*(2*F(27)-2*F(28))-G(25,27)*(2*F(25)-2*F(27))+...
    G(27,32)*(2*F(27)-2*F(32))+...
    G(27,115)*(2*F(27)-2*F(115));...
    G(31,32)*(2*E(31)-2*E(32))-G(29,31)*(2*E(29)-2*E(31))-...
    G(17,31)*(2*E(17)-2*E(31));...
    G(31,32)*(2*F(31)-2*F(32))-G(29,31)*(2*F(29)-2*F(31))-...
    G(17,31)*(2*F(17)-2*F(31));...
```

```
G(32,113)*(2*E(32)-2*E(113))-G(27,32)*(2*E(27)-2*E(32))-...
G(31,32)*(2*E(31)-2*E(32))-...
G(23,32)*(2*E(23)-2*E(32))+G(32,114)*(2*E(32)-2*E(114));...
G(32,113)*(2*F(32)-2*F(113))-G(27,32)*(2*F(27)-2*F(32))-...
G(31,32)*(2*F(31)-2*F(32))-...
G(23,32)*(2*F(23)-2*F(32))+G(32,114)*(2*F(32)-2*F(114));...
G(34,36)*(2*E(34)-2*E(36))-G(19,34)*(2*E(19)-2*E(34))+...
G(34,37)*(2*E(34)-2*E(37))+...
G(34,43)*(2*E(34)-2*E(43));...
G(34,36)*(2*F(34)-2*F(36))-G(19,34)*(2*F(19)-2*F(34))+...
G(34,37)*(2*F(34)-2*F(37))+...
G(34,43)*(2*F(34)-2*F(43));...
- G(34,36)*(2*E(34)-2*E(36))-G(35,36)*(2*E(35)-2*E(36));...
- G(34,36)*(2*F(34)-2*F(36))-G(35,36)*(2*F(35)-2*F(36));...
G(40,41)*(2*E(40)-2*E(41))-G(39,40)*(2*E(39)-2*E(40))-...
G(37,40)*(2*E(37)-2*E(40))+G(40,42)*(2*E(40)-2*E(42));...
G(40,41)*(2*F(40)-2*F(41))-G(39,40)*(2*F(39)-2*F(40))-...
G(37,40)*(2*F(37)-2*F(40))+G(40,42)*(2*F(40)-2*F(42));...
G(42,49)*(2*E(42)-2*E(49))-G(41,42)*(2*E(41)-2*E(42))-...
G(40,42)*(2*E(40)-2*E(42));...
G(42,49)*(2*F(42)-2*F(49))-G(41,42)*(2*F(41)-2*F(42))-...
G(40,42)*(2*F(40)-2*F(42));...
G(46,47)*(2*E(46)-2*E(47))-G(45,46)*(2*E(45)-2*E(46))+...
G(46,48)*(2*E(46)-2*E(48));...
G(46,47)*(2*F(46)-2*F(47))-G(45,46)*(2*F(45)-2*F(46))+...
G(46,48)*(2*F(46)-2*F(48));...
G(49,50)*(2*E(49)-2*E(50))-G(45,49)*(2*E(45)-2*E(49))-...
G(47,49)*(2*E(47)-2*E(49))-...
G(48,49)*(2*E(48)-2*E(49))-G(42,49)*(2*E(42)-2*E(49))+...
G(49,51)*(2*E(49)-2*E(51))+...
G(49,54)*(2*E(49)-2*E(54))+G(49,66)*(2*E(49)-2*E(66))+...
G(49,69)*(2*E(49)-2*E(69));...
G(49,50)*(2*F(49)-2*F(50))-G(45,49)*(2*F(45)-2*F(49))-...
G(47,49)*(2*F(47)-2*F(49))-G(48,49)*(2*F(48)-2*F(49))-...
G(42,49)*(2*F(42)-2*F(49))+G(49,51)*(2*F(49)-2*F(51))+...
G(49,54)*(2*F(49)-2*F(54))+G(49,66)*(2*F(49)-2*F(66))+...
G(49,69)*(2*F(49)-2*F(69));...
G(54,55)*(2*E(54)-2*E(55))-G(53,54)*(2*E(53)-2*E(54))-...
G(49,54)*(2*E(49)-2*E(54))+G(54,56)*(2*E(54)-2*E(56))+...
G(54,59)*(2*E(54)-2*E(59));...
G(54,55)*(2*F(54)-2*F(55))-G(53,54)*(2*F(53)-2*F(54))-...
G(49,54)*(2*F(49)-2*F(54))+G(54,56)*(2*F(54)-2*F(56))+...
G(54,59)*(2*F(54)-2*F(59));...
G(55,56)*(2*E(55)-2*E(56))-G(54,55)*(2*E(54)-2*E(55))+...
G(55,59)*(2*E(55)-2*E(59));...
G(55,56)*(2*F(55)-2*F(56))-G(54,55)*(2*F(54)-2*F(55))+...
G(55,59)*(2*F(55)-2*F(59));...
G(56,57)*(2*E(56)-2*E(57))-G(55,56)*(2*E(55)-2*E(56))-...
G(54,56)*(2*E(54)-2*E(56))+G(56,58)*(2*E(56)-2*E(58))+...
G(56,59)*(2*E(56)-2*E(59));...
G(56,57)*(2*F(56)-2*F(57))-G(55,56)*(2*F(55)-2*F(56))-...
G(54,56)*(2*F(54)-2*F(56))+G(56,58)*(2*F(56)-2*F(58))+...
G(56,59)*(2*F(56)-2*F(59));...
G(59,60)*(2*E(59)-2*E(60))-G(55,59)*(2*E(55)-2*E(59))-...
```

```
G(56,59)*(2*E(56)-2*E(59))-G(54,59)*(2*E(54)-2*E(59))+...
G(59,61)*(2*E(59)-2*E(61))+G(59,63)*(2*E(59)-2*E(63));...
G(59,60)*(2*F(59)-2*F(60))-G(55,59)*(2*F(55)-2*F(59))-...
G(56,59)*(2*F(56)-2*F(59))-G(54,59)*(2*F(54)-2*F(59))+...
G(59,61)*(2*F(59)-2*F(61))+G(59,63)*(2*F(59)-2*F(63));...
G(61,62)*(2*E(61)-2*E(62))-G(60,61)*(2*E(60)-2*E(61))-...
G(59,61)*(2*E(59)-2*E(61))+G(61,64)*(2*E(61)-2*E(64));...
G(61,62)*(2*F(61)-2*F(62))-G(60,61)*(2*F(60)-2*F(61))-...
G(59,61)*(2*F(59)-2*F(61))+G(61,64)*(2*F(61)-2*F(64));...
G(62,66)*(2*E(62)-2*E(66))-G(61,62)*(2*E(61)-2*E(62))-...
G(60,62)*(2*E(60)-2*E(62))+G(62,67)*(2*E(62)-2*E(67));...
G(62,66)*(2*F(62)-2*F(66))-G(61,62)*(2*F(61)-2*F(62))-...
G(60,62)*(2*F(60)-2*F(62))+G(62,67)*(2*F(62)-2*F(67));...
G(65,66)*(2*E(65)-2*E(66))-G(64,65)*(2*E(64)-2*E(65))-...
G(38,65)*(2*E(38)-2*E(65))+G(65,68)*(2*E(65)-2*E(68));...
G(65,66)*(2*F(65)-2*F(66))-G(64,65)*(2*F(64)-2*F(65))-...
G(38,65)*(2*F(38)-2*F(65))+G(65,68)*(2*F(65)-2*F(68));...
G(66,67)*(2*E(66)-2*E(67))-G(62,66)*(2*E(62)-2*E(66))-...
G(65,66)*(2*E(65)-2*E(66))-G(49,66)*(2*E(49)-2*E(66));...
G(66,67)*(2*F(66)-2*F(67))-G(62,66)*(2*F(62)-2*F(66))-...
G(65,66)*(2*F(65)-2*F(66))-G(49,66)*(2*F(49)-2*F(66));...
G(69,70)*(2*E(69)-2*E(70))-G(49,69)*(2*E(49)-2*E(69))-...
G(68,69)*(2*E(68)-2*E(69))-G(47,69)*(2*E(47)-2*E(69))+...
G(69,75)*(2*E(69)-2*E(75))+G(69,77)*(2*E(69)-2*E(77));...
G(69,70)*(2*F(69)-2*F(70))-G(49,69)*(2*F(49)-2*F(69))-...
G(68,69)*(2*F(68)-2*F(69))-G(47,69)*(2*F(47)-2*F(69))+...
G(69,75)*(2*F(69)-2*F(75))+G(69,77)*(2*F(69)-2*F(77));...
G(70,71)*(2*E(70)-2*E(71))-G(69,70)*(2*E(69)-2*E(70))-...
G(24,70)*(2*E(24)-2*E(70))+G(70,74)*(2*E(70)-2*E(74))+...
G(70,75)*(2*E(70)-2*E(75));...
G(70,71)*(2*F(70)-2*F(71))-G(69,70)*(2*F(69)-2*F(70))-...
G(24,70)*(2*F(24)-2*F(70))+G(70,74)*(2*F(70)-2*F(74))+...
G(70,75)*(2*F(70)-2*F(75));...
- G(24,72)*(2*E(24)-2*E(72))-G(71,72)*(2*E(71)-2*E(72));...
- G(24,72)*(2*F(24)-2*F(72))-G(71,72)*(2*F(71)-2*F(72));...
-G(71,73)*(2*E(71)-2*E(73));...
-G(71,73)*(2*F(71)-2*F(73));...
G(74,75)*(2*E(74)-2*E(75))-G(70,74)*(2*E(70)-2*E(74));...
G(74,75)*(2*F(74)-2*F(75))-G(70,74)*(2*F(70)-2*F(74));...
G(76,77)*(2*E(76)-2*E(77))+G(76,118)*(2*E(76)-2*E(118));...
G(76,77)*(2*F(76)-2*F(77))+G(76,118)*(2*F(76)-2*F(118));...
G(77,78)*(2*E(77)-2*E(78))-G(75,77)*(2*E(75)-2*E(77))-...
G(76,77)*(2*E(76)-2*E(77))-G(69,77)*(2*E(69)-2*E(77))+...
G(77,80)*(2*E(77)-2*E(80))+G(77,82)*(2*E(77)-2*E(82));...
G(77,78)*(2*F(77)-2*F(78))-G(75,77)*(2*F(75)-2*F(77))-...
G(76,77)*(2*F(76)-2*F(77))-G(69,77)*(2*F(69)-2*F(77))+...
G(77,80)*(2*F(77)-2*F(80))+G(77,82)*(2*F(77)-2*F(82));...
G(80,81)*(2*E(80)-2*E(81))-G(79,80)*(2*E(79)-2*E(80))-...
G(77,80)*(2*E(77)-2*E(80))+G(80,96)*(2*E(80)-2*E(96))+...
G(80,97)*(2*E(80)-2*E(97))+G(80,98)*(2*E(80)-2*E(98))+...
G(80,99)*(2*E(80)-2*E(99));...
G(80,81)*(2*F(80)-2*F(81))-G(79,80)*(2*F(79)-2*F(80))-...
G(77,80)*(2*F(77)-2*F(80))+G(80,96)*(2*F(80)-2*F(96))+...
G(80,97)*(2*F(80)-2*F(97))+G(80,98)*(2*F(80)-2*F(98))+...
```

```
G(80,99)*(2*F(80)-2*F(99));...
G(85,86)*(2*E(85)-2*E(86))-G(84,85)*(2*E(84)-2*E(85))-...
G(83,85)*(2*E(83)-2*E(85))+G(85,88)*(2*E(85)-2*E(88))+...
G(85,89)*(2*E(85)-2*E(89));...
G(85,86)*(2*F(85)-2*F(86))-G(84,85)*(2*F(84)-2*F(85))-...
G(83,85)*(2*F(83)-2*F(85))+G(85,88)*(2*F(85)-2*F(88))+...
G(85,89)*(2*F(85)-2*F(89));...
-G(86,87)*(2*E(86)-2*E(87));...
-G(86,87)*(2*F(86)-2*F(87));...
G(89,90)*(2*E(89)-2*E(90))-G(88,89)*(2*E(88)-2*E(89))-...
G(85,89)*(2*E(85)-2*E(89))+G(89,92)*(2*E(89)-2*E(92));...
G(89,90)*(2*F(89)-2*F(90))-G(88,89)*(2*F(88)-2*F(89))-...
G(85,89)*(2*F(85)-2*F(89))+G(89,92)*(2*F(89)-2*F(92));...
G(90,91)*(2*E(90)-2*E(91))-G(89,90)*(2*E(89)-2*E(90));...
G(90,91)*(2*F(90)-2*F(91))-G(89,90)*(2*F(89)-2*F(90));...
G(91,92)*(2*E(91)-2*E(92))-G(90,91)*(2*E(90)-2*E(91));...
G(91,92)*(2*F(91)-2*F(92))-G(90,91)*(2*F(90)-2*F(91));...
G(92,93)*(2*E(92)-2*E(93))-G(91,92)*(2*E(91)-2*E(92))-...
G(89,92)*(2*E(89)-2*E(92))+G(92,94)*(2*E(92)-2*E(94))+...
G(92,100)*(2*E(92)-2*E(100))+G(92,102)*(2*E(92)-2*E(102));...
G(92,93)*(2*F(92)-2*F(93))-G(91,92)*(2*F(91)-2*F(92))-...
G(89,92)*(2*F(89)-2*F(92))+G(92,94)*(2*F(92)-2*F(94))+...
G(92,100)*(2*F(92)-2*F(100))+G(92,102)*(2*F(92)-2*F(102));...
G(99,100)*(2*E(99)-2*E(100))-G(80,99)*(2*E(80)-2*E(99));...
G(99,100)*(2*F(99)-2*F(100))-G(80,99)*(2*F(80)-2*F(99));...
G(100,101)*(2*E(100)-2*E(101))-G(94,100)*(2*E(94)-2*E(100))-...
G(98,100)*(2*E(98)-2*E(100))-G(99,100)*(2*E(99)-2*E(100))-...
G(92,100)*(2*E(92)-2*E(100))+G(100,103)*(2*E(100)-2*E(103))+...
G(100,104)*(2*E(100)-2*E(104))+G(100,106)*(2*E(100)-2*E(106));...
G(100,101)*(2*F(100)-2*F(101))-G(94,100)*(2*F(94)-2*F(100))-...
G(98,100)*(2*F(98)-2*F(100))-G(99,100)*(2*F(99)-2*F(100))-...
G(92,100)*(2*F(92)-2*F(100))+G(100,103)*(2*F(100)-2*F(103))+...
G(100,104)*(2*F(100)-2*F(104))+G(100,106)*(2*F(100)-2*F(106));...
G(103,104)*(2*E(103)-2*E(104))-G(100,103)*(2*E(100)-2*E(103)) + ...
G(103,105)*(2*E(103)-2*E(105))+G(103,110)*(2*E(103)-2*E(110));...
G(103,104)*(2*F(103)-2*F(104))-G(100,103)*(2*F(100)-2*F(103)) + ...
G(103,105)*(2*F(103)-2*F(105))+G(103,110)*(2*F(103)-2*F(110));...
G(104,105)*(2*E(104)-2*E(105))-G(103,104)*(2*E(103)-2*E(104))-...
G(100,104)*(2*E(100)-2*E(104));...
G(104,105)*(2*F(104)-2*F(105))-G(103,104)*(2*F(103)-2*F(104))-...
G(100,104)*(2*F(100)-2*F(104));...
G(105,106)*(2*E(105)-2*E(106))-G(104,105)*(2*E(104)-2*E(105))-...
G(103,105)*(2*E(103)-2*E(105))+G(105,107)*(2*E(105)-2*E(107)) + ...
G(105,108)*(2*E(105)-2*E(108));...
G(105,106)*(2*F(105)-2*F(106))-G(104,105)*(2*F(104)-2*F(105))-...
G(103,105)*(2*F(103)-2*F(105))+G(105,107)*(2*F(105)-2*F(107))+...
G(105,108)*(2*F(105)-2*F(108));...
- G(105,107)*(2*E(105)-2*E(107))-G(106,107)*(2*E(106)-2*E(107));...
- G(105,107)*(2*F(105)-2*F(107))-G(106,107)*(2*F(106)-2*F(107));...
G(110,111)*(2*E(110)-2*E(111))-G(109,110)*(2*E(109)-2*E(110))-...
G(103,110)*(2*E(103)-2*E(110))+G(110,112)*(2*E(110)-2*E(112));...
G(110,111)*(2*F(110)-2*F(111))-G(109,110)*(2*F(109)-2*F(110))-...
G(103,110)*(2*F(103)-2*F(110))+G(110,112)*(2*F(110)-2*F(112));...
-G(110,111)*(2*E(110)-2*E(111));...
```

```
    -G(110,111)*(2*F(110)-2*F(111));...
    -G(110,112)*(2*E(110)-2*E(112));...
    -G(110,112)*(2*F(110)-2*F(112));...
    - G(17,113)*(2*E(17)-2*E(113))-G(32,113)*(2*E(32)-2*E(113));...
    - G(17,113)*(2*F(17)-2*F(113))-G(32,113)*(2*F(32)-2*F(113));...
    -G(68,116)*(2*E(68)-2*E(116));...
    -G(68,116)*(2*F(68)-2*F(116))];
```

B.27 Function that computes the Hessian of the objective function for the 118-bus system

```
function d2f=d2f_118bus()
global G

d2f=zeros(107);

d2f(1,1)=2*G(1,2) + 2*G(1,3);
d2f(2,2)=2*G(4,5) + 2*G(4,11);
d2f(3,3)=2*G(4,5) + 2*G(4,11);
d2f(4,4)=2*G(5,6) + 2*G(6,7);
d2f(5,5)=2*G(5,6) + 2*G(6,7);
d2f(6,6)=2*G(5,8) + 2*G(8,9) + 2*G(8,30);
d2f(7,7)=2*G(5,8) + 2*G(8,9) + 2*G(8,30);
d2f(8,8)=2*G(9,10);
d2f(9,9)=2*G(9,10);
d2f(10,10)=2*G(11,12) + 2*G(12,14) + 2*G(12,16)+...
    2*G(12,117) + 2*G(2,12) + 2*G(3,12) + 2*G(7,12);
d2f(11,11)=2*G(11,12) + 2*G(12,14) + 2*G(12,16) + 2*G(12,117) + 2*G(2,12) +
2*G(3,12) + 2*G(7,12);
d2f(12,12)=2*G(13,15) + 2*G(14,15) + 2*G(15,17) + 2*G(15,19) + 2*G(15,33);
d2f(12,16)=-2*G(15,19);
d2f(13,13)=2*G(13,15) + 2*G(14,15) + 2*G(15,17) + 2*G(15,19) + 2*G(15,33);
d2f(13,17)=-2*G(15,19);
d2f(14,14)=2*G(17,18) + 2*G(18,19);
d2f(14,16)=-2*G(18,19);
d2f(15,15)=2*G(17,18) + 2*G(18,19);
d2f(15,17)=-2*G(18,19);
d2f(16,12)=-2*G(15,19);
d2f(16,14)=-2*G(18,19);
d2f(16,16)=2*G(15,19) + 2*G(18,19) + 2*G(19,20) + 2*G(19,34);
d2f(16,30)=-2*G(19,34);
d2f(17,13)=-2*G(15,19);
d2f(17,15)=-2*G(18,19);
d2f(17,17)=2*G(15,19) + 2*G(18,19) + 2*G(19,20) + 2*G(19,34);
d2f(17,31)=-2*G(19,34);
d2f(18,18)=2*G(23,24) + 2*G(24,70) + 2*G(24,72);
d2f(18,60)=-2*G(24,70);
d2f(18,62)=-2*G(24,72);
d2f(19,19)=2*G(23,24) + 2*G(24,70) + 2*G(24,72);
d2f(19,61)=-2*G(24,70);
d2f(19,63)=-2*G(24,72);
d2f(20,20)=2*G(23,25) + 2*G(25,26) + 2*G(25,27);
```

```
d2f(20,22)=-2*G(25,26);
d2f(20,24)=-2*G(25,27);
d2f(21,21)=2*G(23,25) + 2*G(25,26) + 2*G(25,27);
d2f(21,23)=-2*G(25,26);
d2f(21,25)=-2*G(25,27);
d2f(22,20)=-2*G(25,26);
d2f(22,22)=2*G(25,26) + 2*G(26,30);
d2f(23,21)=-2*G(25,26);
d2f(23,23)=2*G(25,26) + 2*G(26,30);
d2f(24,20)=-2*G(25,27);
d2f(24,24)=2*G(25,27) + 2*G(27,28) + 2*G(27,32) + 2*G(27,115);
d2f(24,28)=-2*G(27,32);
d2f(25,21)=-2*G(25,27);
d2f(25,25)=2*G(25,27) + 2*G(27,28) + 2*G(27,32) + 2*G(27,115);
d2f(25,29)=-2*G(27,32);
d2f(26,26)=2*G(17,31) + 2*G(29,31) + 2*G(31,32);
d2f(26,28)=-2*G(31,32);
d2f(27,27)=2*G(17,31) + 2*G(29,31) + 2*G(31,32);
d2f(27,29)=-2*G(31,32);
d2f(28,24)=-2*G(27,32);
d2f(28,26)=-2*G(31,32);
d2f(28,28)=2*G(23,32) + 2*G(27,32) + 2*G(31,32) + 2*G(32,113) + 2*G(32,114);
d2f(28,104)=-2*G(32,113);
d2f(29,25)=-2*G(27,32);
d2f(29,27)=-2*G(31,32);
d2f(29,29)=2*G(23,32) + 2*G(27,32) + 2*G(31,32) + 2*G(32,113) + 2*G(32,114);
d2f(29,105)=-2*G(32,113);
d2f(30,16)=-2*G(19,34);
d2f(30,30)=2*G(19,34) + 2*G(34,36) + 2*G(34,37) + 2*G(34,43);
d2f(30,32)=-2*G(34,36);
d2f(31,17)=-2*G(19,34);
d2f(31,31)=2*G(19,34) + 2*G(34,36) + 2*G(34,37) + 2*G(34,43);
d2f(31,33)=-2*G(34,36);
d2f(32,30)=-2*G(34,36);
d2f(32,32)=2*G(34,36) + 2*G(35,36);
d2f(33,31)=-2*G(34,36);
d2f(33,33)=2*G(34,36) + 2*G(35,3);
d2f(34,34)=2*G(37,40) + 2*G(39,40) + 2*G(40,41) + 2*G(40,42);
d2f(34,36)=-2*G(40,42);
d2f(35,35)=2*G(37,40) + 2*G(39,40) + 2*G(40,41) + 2*G(40,42);
d2f(35,37)=-2*G(40,42);
d2f(36,34)=-2*G(40,42);
d2f(36,36)=2*G(40,42) + 2*G(41,42) + 2*G(42,49);
d2f(36,40)=-2*G(42,49);
d2f(37,35)=-2*G(40,42);
d2f(37,37)=2*G(40,42) + 2*G(41,42) + 2*G(42,49);
d2f(37,41)=-2*G(42,49);
d2f(38,38)=2*G(45,46) + 2*G(46,47) + 2*G(46,48);
d2f(39,39)=2*G(45,46) + 2*G(46,47) + 2*G(46,48);
d2f(40,36)=-2*G(42,49);
d2f(40,40)=2*G(42,49) + 2*G(45,49) + 2*G(47,49) + ...
    2*G(48,49) + 2*G(49,50) + 2*G(49,51) + 2*G(49,54) + ...
    2*G(49,66) + 2*G(49,69);
d2f(40,42)=-2*G(49,54);
```

```
d2f(40,56)=-2*G(49,66);
d2f(40,58)=-2*G(49,69);
d2f(41,37)=-2*G(42,49);
d2f(41,41)=2*G(42,49) + 2*G(45,49) + 2*G(47,49) + ...
    2*G(48,49) + 2*G(49,50) + 2*G(49,51) + 2*G(49,54) + ...
    2*G(49,66) + 2*G(49,69);
d2f(41,43)=-2*G(49,54);
d2f(41,57)=-2*G(49,66);
d2f(41,59)=-2*G(49,69);
d2f(42,40)=-2*G(49,54);
d2f(42,42)=2*G(49,54) + 2*G(53,54) + 2*G(54,55) + 2*G(54,56) + 2*G(54,59);
d2f(42,44)=-2*G(54,55);
d2f(42,46)=-2*G(54,56);
d2f(42,48)=-2*G(54,59);
d2f(43,41)=-2*G(49,54);
d2f(43,43)=2*G(49,54) + 2*G(53,54) + 2*G(54,55) + 2*G(54,56) + 2*G(54,59);
d2f(43,45)=-2*G(54,55);
d2f(43,47)=-2*G(54,56);
d2f(43,49)=-2*G(54,59);
d2f(44,42)=-2*G(54,55);
d2f(44,44)=2*G(54,55) + 2*G(55,56) + 2*G(55,59);
d2f(44,46)=-2*G(55,56);
d2f(44,48)=-2*G(55,59);
d2f(45,43)=-2*G(54,55);
d2f(45,45)=2*G(54,55) + 2*G(55,56) + 2*G(55,59);
d2f(45,47)=-2*G(55,56);
d2f(45,49)=-2*G(55,59);
d2f(46,42)=-2*G(54,56);
d2f(46,44)=-2*G(55,56);
d2f(46,46)=2*G(54,56) + 2*G(55,56) + 2*G(56,57) + 2*G(56,58) + 2*G(56,59);
d2f(46,48)=-2*G(56,59);
d2f(47,43)=-2*G(54,56);
d2f(47,45)=-2*G(55,56);
d2f(47,47)=2*G(54,56) + 2*G(55,56) + 2*G(56,57) + 2*G(56,58) + 2*G(56,59);
d2f(47,49)=-2*G(56,59);
d2f(48,42)=-2*G(54,59);
d2f(48,44)=-2*G(55,59);
d2f(48,46)=-2*G(56,59);
d2f(48,48)=2*G(54,59) + 2*G(55,59) + 2*G(56,59) + 2*G(59,60) + 2*G(59,61) +
2*G(59,63);
d2f(48,50)=-2*G(59,61);
d2f(49,43)=-2*G(54,59);
d2f(49,45)=-2*G(55,59);
d2f(49,47)=-2*G(56,59);
d2f(49,49)=2*G(54,59) + 2*G(55,59) + 2*G(56,59) + 2*G(59,60) + 2*G(59,61) +
2*G(59,63);
d2f(49,51)=-2*G(59,61);
d2f(50,48)=-2*G(59,61);
d2f(50,50)=2*G(59,61) + 2*G(60,61) + 2*G(61,62) + 2*G(61,64);
d2f(50,52)=-2*G(61,62);
d2f(51,49)=-2*G(59,61);
d2f(51,51)=2*G(59,61) + 2*G(60,61) + 2*G(61,62) + 2*G(61,64);
d2f(51,53)=-2*G(61,62);
d2f(52,50)=-2*G(61,62);
```

```
d2f(52,52)=2*G(60,62) + 2*G(61,62) + 2*G(62,66) + 2*G(62,67);
d2f(52,56)=-2*G(62,66);
d2f(53,51)=-2*G(61,62);
d2f(53,53)=2*G(60,62) + 2*G(61,62) + 2*G(62,66) + 2*G(62,67);
d2f(53,57)=-2*G(62,66);
d2f(54,54)=2*G(38,65) + 2*G(64,65) + 2*G(65,66) + 2*G(65,68);
d2f(54,56)=-2*G(65,66);
d2f(55,55)=2*G(38,65) + 2*G(64,65) + 2*G(65,66) + 2*G(65,68);
d2f(55,57)=-2*G(65,66);
d2f(56,40)=-2*G(49,66);
d2f(56,52)=-2*G(62,66);
d2f(56,54)=-2*G(65,66);
d2f(56,56)=2*G(49,66) + 2*G(62,66) + 2*G(65,66) + 2*G(66,67);
d2f(57,41)=-2*G(49,66);
d2f(57,53)=-2*G(62,66);
d2f(57,55)=-2*G(65,66);
d2f(57,57)=2*G(49,66) + 2*G(62,66) + 2*G(65,66) + 2*G(66,67);
d2f(58,40)=-2*G(49,69);
d2f(58,58)=2*G(47,69) + 2*G(49,69) + 2*G(68,69) + 2*G(69,70) + 2*G(69,75) +
2*G(69,77);
d2f(58,60)=-2*G(69,70);
d2f(58,70)=-2*G(69,77);
d2f(59,41)=-2*G(49,69);
d2f(59,59)=2*G(47,69) + 2*G(49,69) + 2*G(68,69) + 2*G(69,70) + 2*G(69,75) +
2*G(69,77);
d2f(59,61)=-2*G(69,70);
d2f(59,71)=-2*G(69,77);
d2f(60,18)=-2*G(24,70);
d2f(60,58)=-2*G(69,70);
d2f(60,60)=2*G(24,70) + 2*G(69,70) + 2*G(70,71) + 2*G(70,74) + 2*G(70,75);
d2f(60,66)=-2*G(70,74);
d2f(61,19)=-2*G(24,70);
d2f(61,59)=-2*G(69,70);
d2f(61,61)=2*G(24,70) + 2*G(69,70) + 2*G(70,71) + 2*G(70,74) + 2*G(70,75);
d2f(61,67)=-2*G(70,74);
d2f(62,18)=-2*G(24,72);
d2f(62,62)=2*G(24,72) + 2*G(71,72);
d2f(63,19)=-2*G(24,72);
d2f(63,63)=2*G(24,72) + 2*G(71,72);
d2f(64,64)=2*G(71,73);
d2f(65,65)=2*G(71,73);
d2f(66,60)=-2*G(70,74);
d2f(66,66)=2*G(70,74) + 2*G(74,75);
d2f(67,61)=-2*G(70,74);
d2f(67,67)=2*G(70,74) + 2*G(74,75);
d2f(68,68)=2*G(76,77) + 2*G(76,118);
d2f(68,70)=-2*G(76,77);
d2f(69,69)=2*G(76,77) + 2*G(76,118);
d2f(69,71)=-2*G(76,77);
d2f(70,58)=-2*G(69,77);
d2f(70,68)=-2*G(76,77);
d2f(70,70)=2*G(69,77) + 2*G(75,77) + ...
    2*G(76,77) + 2*G(77,78) + 2*G(77,80) + 2*G(77,82);
d2f(70,72)=-2*G(77,80);
```

```
d2f(71,59)=-2*G(69,77);
d2f(71,69)=-2*G(76,77);
d2f(71,71)=2*G(69,77) + 2*G(75,77) + 2*G(76,77) + ...
    2*G(77,78) + 2*G(77,80) + 2*G(77,82);
d2f(71,73)=-2*G(77,80);
d2f(72,70)=-2*G(77,80);
d2f(72,72)=2*G(77,80) + 2*G(79,80) + 2*G(80,81) + ...
    2*G(80,96) + 2*G(80,97) + 2*G(80,98) + 2*G(80,99);
d2f(72,86)=-2*G(80,99);
d2f(73,71)=-2*G(77,80);
d2f(73,73)=2*G(77,80) + 2*G(79,80) + 2*G(80,81) + 2*G(80,96) + 2*G(80,97) +
2*G(80,98) + 2*G(80,99);
d2f(73,87)=-2*G(80,99);
d2f(74,74)=2*G(83,85) + 2*G(84,85) + 2*G(85,86) + 2*G(85,88) + 2*G(85,89);
d2f(74,78)=-2*G(85,89);
d2f(75,75)=2*G(83,85) + 2*G(84,85) + 2*G(85,86) + 2*G(85,88) + 2*G(85,89);
d2f(75,79)=-2*G(85,89);
d2f(76,76)=2*G(86,87);
d2f(77,77)=2*G(86,87);
d2f(78,74)=-2*G(85,89);
d2f(78,78)=2*G(85,89) + 2*G(88,89) + 2*G(89,90) + 2*G(89,92);
d2f(78,80)=-2*G(89,90);
d2f(78,84)=-2*G(89,92);
d2f(79,75)=-2*G(85,89);
d2f(79,79)=2*G(85,89) + 2*G(88,89) + 2*G(89,90) + 2*G(89,92);
d2f(79,81)=-2*G(89,90);
d2f(79,85)=-2*G(89,92);
d2f(80,78)=-2*G(89,90);
d2f(80,80)=2*G(89,90) + 2*G(90,91);
d2f(80,82)=-2*G(90,91);
d2f(81,79)=-2*G(89,90);
d2f(81,81)=2*G(89,90) + 2*G(90,91);
d2f(81,83)=-2*G(90,91);
d2f(82,80)=-2*G(90,91);
d2f(82,82)=2*G(90,91) + 2*G(91,92);
d2f(82,84)=-2*G(91,92);
d2f(83,81)=-2*G(90,91);
d2f(83,83)=2*G(90,91) + 2*G(91,92);
d2f(83,85)=-2*G(91,92);
d2f(84,78)=-2*G(89,92);
d2f(84,82)=-2*G(91,92);
d2f(84,84)=2*G(89,92) + 2*G(91,92) + 2*G(92,93) + 2*G(92,94) + ...
    2*G(92,100) + 2*G(92,102);
d2f(84,88)=-2*G(92,100);
d2f(85,79)=-2*G(89,92);
d2f(85,83)=-2*G(91,92);
d2f(85,85)=2*G(89,92) + 2*G(91,92) + 2*G(92,93) + 2*G(92,94) + ...
    2*G(92,100) + 2*G(92,102);
d2f(85,89)=-2*G(92,100);
d2f(86,72)=-2*G(80,99);
d2f(86,86)=2*G(80,99) + 2*G(99,100);
d2f(86,88)=-2*G(99,100);
d2f(87,73)=-2*G(80,99);
d2f(87,87)=2*G(80,99) + 2*G(99,100);
```

```
d2f(87,89)=-2*G(99,100);
d2f(88,84)=-2*G(92,100);
d2f(88,86)=-2*G(99,100);
d2f(88,88)=2*G(92,100) + 2*G(94,100) + 2*G(98,100) + 2*G(99,100) + ...
    2*G(100,101) + 2*G(100,103) + 2*G(100,104) + 2*G(100,106);
d2f(88,90)=-2*G(100,103);
d2f(88,92)=-2*G(100,104);
d2f(89,85)=-2*G(92,100);
d2f(89,87)=-2*G(99,100);
d2f(89,89)=2*G(92,100) + 2*G(94,100) + 2*G(98,100) + 2*G(99,100) + ...
    2*G(100,101) + 2*G(100,103) + 2*G(100,104) + 2*G(100,106);
d2f(89,91)=-2*G(100,103);
d2f(89,93)=-2*G(100,104);
d2f(90,88)=-2*G(100,103);
d2f(90,90)=2*G(100,103) + 2*G(103,104) + 2*G(103,105) + 2*G(103,110);
d2f(90,92)=-2*G(103,104);
d2f(90,94)=-2*G(103,105);
d2f(90,98)=-2*G(103,110);
d2f(91,89)=-2*G(100,103);
d2f(91,91)=2*G(100,103) + 2*G(103,104) + 2*G(103,105) + 2*G(103,110);
d2f(91,93)=-2*G(103,104);
d2f(91,95)=-2*G(103,105);
d2f(91,99)=-2*G(103,110);
d2f(92,88)=-2*G(100,104);
d2f(92,90)=-2*G(103,104);
d2f(92,92)=2*G(100,104) + 2*G(103,104) + 2*G(104,105);
d2f(92,94)=-2*G(104,105);
d2f(93,89)=-2*G(100,104);
d2f(93,91)=-2*G(103,104);
d2f(93,93)=2*G(100,104) + 2*G(103,104) + 2*G(104,105);
d2f(93,95)=-2*G(104,105);
d2f(94,90)=-2*G(103,105);
d2f(94,92)=-2*G(104,105);
d2f(94,94)=2*G(103,105) + 2*G(104,105) + 2*G(105,106) + ...
    2*G(105,107) + 2*G(105,108);
d2f(94,96)=-2*G(105,107);
d2f(95,91)=-2*G(103,105);
d2f(95,93)=-2*G(104,105);
d2f(95,95)=2*G(103,105) + 2*G(104,105) + 2*G(105,106) + 2*G(105,107) +
2*G(105,108);
d2f(95,97)=-2*G(105,107);
d2f(96,94)=-2*G(105,107);
d2f(96,96)=2*G(105,107) + 2*G(106,107);
d2f(97,95)=-2*G(105,107);
d2f(97,97)=2*G(105,107) + 2*G(106,107);
d2f(98,90)=-2*G(103,110);
d2f(98,98)=2*G(103,110) + 2*G(109,110) + 2*G(110,111) + 2*G(110,112);
d2f(98,100)=-2*G(110,111);
d2f(98,102)=-2*G(110,112);
d2f(99,91)=-2*G(103,110);
d2f(99,99)=2*G(103,110) + 2*G(109,110) + 2*G(110,111) + 2*G(110,112);
d2f(99,101)=-2*G(110,111);
d2f(99,103)=-2*G(110,112);
d2f(100,98)=-2*G(110,111);
```

```
d2f(100,100)=2*G(110,111);
d2f(101,99)=-2*G(110,111);
d2f(101,101)=2*G(110,111);
d2f(102,98)=-2*G(110,112);
d2f(102,102)=2*G(110,112);
d2f(103,99)=-2*G(110,112);
d2f(103,103)=2*G(110,112);
d2f(104,28)=-2*G(32,113);
d2f(104,104)=2*G(17,113) + 2*G(32,113);
d2f(105,29)=-2*G(32,113);
d2f(105,105)=2*G(17,113) + 2*G(32,113);
d2f(106,106)=2*G(68,116);
d2f(107,107)=2*G(68,116);
```

B.28 Function that defines the constraint functions for the 118-bus system

```
function h=h_118bus(x)
global E F nbus

Vmsq=.95^2;
VMsq=1.1^2;

% update generator voltage vector components
% in E, F from input control vector x:
update_generator_voltages(x);

% Define inequality constraints h(x):
h=zeros(2*nbus,1);
k=1;
for i=1:nbus
    h(k)=-(E(i)^2+F(i)^2)+Vmsq;
    h(k+1)=E(i)^2+F(i)^2-VMsq;
    k=k+2;
end
```

B.29 Function that computes the Jacobian of the constraint functions for the 118-bus system

```
function dh=dh_118bus()
global E F nbus

dh=zeros(2*nbus, 107);

dh(1,1)=-2*E(1);
dh(2,1)=2*E(1);
dh(7,2)=-2*E(4);
dh(7,3)=-2*F(4);
dh(8,2)=2*E(4);
dh(8,3)=2*F(4);
dh(11,4)=-2*E(6);
dh(11,5)=-2*F(6);
```

```
dh(12,4)=2*E(6);
dh(12,5)=2*F(6);
dh(15,6)=-2*E(8);
dh(15,7)=-2*F(8);
dh(16,6)=2*E(8);
dh(16,7)=2*F(8);
dh(19,8)=-2*E(10);
dh(19,9)=-2*F(10);
dh(20,8)=2*E(10);
dh(20,9)=2*F(10);
dh(23,10)=-2*E(12);
dh(23,11)=-2*F(12);
dh(24,10)=2*E(12);
dh(24,11)=2*F(12);
dh(29,12)=-2*E(15);
dh(29,13)=-2*F(15);
dh(30,12)=2*E(15);
dh(30,13)=2*F(15);
dh(35,14)=-2*E(18);
dh(35,15)=-2*F(18);
dh(36,14)=2*E(18);
dh(36,15)=2*F(18);
dh(37,16)=-2*E(19);
dh(37,17)=-2*F(19);
dh(38,16)=2*E(19);
dh(38,17)=2*F(19);
dh(47,18)=-2*E(24);
dh(47,19)=-2*F(24);
dh(48,18)=2*E(24);
dh(48,19)=2*F(24);
dh(49,20)=-2*E(25);
dh(49,21)=-2*F(25);
dh(50,20)=2*E(25);
dh(50,21)=2*F(25);
dh(51,22)=-2*E(26);
dh(51,23)=-2*F(26);
dh(52,22)=2*E(26);
dh(52,23)=2*F(26);
dh(53,24)=-2*E(27);
dh(53,25)=-2*F(27);
dh(54,24)=2*E(27);
dh(54,25)=2*F(27);
dh(61,26)=-2*E(31);
dh(61,27)=-2*F(31);
dh(62,26)=2*E(31);
dh(62,27)=2*F(31);
dh(63,28)=-2*E(32);
dh(63,29)=-2*F(32);
dh(64,28)=2*E(32);
dh(64,29)=2*F(32);
dh(67,30)=-2*E(34);
dh(67,31)=-2*F(34);
dh(68,30)=2*E(34);
dh(68,31)=2*F(34);
```

```
dh(71,32)=-2*E(36);
dh(71,33)=-2*F(36);
dh(72,32)=2*E(36);
dh(72,33)=2*F(36);
dh(79,34)=-2*E(40);
dh(79,35)=-2*F(40);
dh(80,34)=2*E(40);
dh(80,35)=2*F(40);
dh(83,36)=-2*E(42);
dh(83,37)=-2*F(42);
dh(84,36)=2*E(42);
dh(84,37)=2*F(42);
dh(91,38)=-2*E(46);
dh(91,39)=-2*F(46);
dh(92,38)=2*E(46);
dh(92,39)=2*F(46);
dh(97,40)=-2*E(49);
dh(97,41)=-2*F(49);
dh(98,40)=2*E(49);
dh(98,41)=2*F(49);
dh(107,42)=-2*E(54);
dh(107,43)=-2*F(54);
dh(108,42)=2*E(54);
dh(108,43)=2*F(54);
dh(109,44)=-2*E(55);
dh(109,45)=-2*F(55);
dh(110,44)=2*E(55);
dh(110,45)=2*F(55);
dh(111,46)=-2*E(56);
dh(111,47)=-2*F(56);
dh(112,46)=2*E(56);
dh(112,47)=2*F(56);
dh(117,48)=-2*E(59);
dh(117,49)=-2*F(59);
dh(118,48)=2*E(59);
dh(118,49)=2*F(59);
dh(121,50)=-2*E(61);
dh(121,51)=-2*F(61);
dh(122,50)=2*E(61);
dh(122,51)=2*F(61);
dh(123,52)=-2*E(62);
dh(123,53)=-2*F(62);
dh(124,52)=2*E(62);
dh(124,53)=2*F(62);
dh(129,54)=-2*E(65);
dh(129,55)=-2*F(65);
dh(130,54)=2*E(65);
dh(130,55)=2*F(65);
dh(131,56)=-2*E(66);
dh(131,57)=-2*F(66);
dh(132,56)=2*E(66);
dh(132,57)=2*F(66);
dh(137,58)=-2*E(69);
dh(137,59)=-2*F(69);
```

```
dh(138,58)=2*E(69);
dh(138,59)=2*F(69);
dh(139,60)=-2*E(70);
dh(139,61)=-2*F(70);
dh(140,60)=2*E(70);
dh(140,61)=2*F(70);
dh(143,62)=-2*E(72);
dh(143,63)=-2*F(72);
dh(144,62)=2*E(72);
dh(144,63)=2*F(72);
dh(145,64)=-2*E(73);
dh(145,65)=-2*F(73);
dh(146,64)=2*E(73);
dh(146,65)=2*F(73);
dh(147,66)=-2*E(74);
dh(147,67)=-2*F(74);
dh(148,66)=2*E(74);
dh(148,67)=2*F(74);
dh(151,68)=-2*E(76);
dh(151,69)=-2*F(76);
dh(152,68)=2*E(76);
dh(152,69)=2*F(76);
dh(153,70)=-2*E(77);
dh(153,71)=-2*F(77);
dh(154,70)=2*E(77);
dh(154,71)=2*F(77);
dh(159,72)=-2*E(80);
dh(159,73)=-2*F(80);
dh(160,72)=2*E(80);
dh(160,73)=2*F(80);
dh(169,74)=-2*E(85);
dh(169,75)=-2*F(85);
dh(170,74)=2*E(85);
dh(170,75)=2*F(85);
dh(173,76)=-2*E(87);
dh(173,77)=-2*F(87);
dh(174,76)=2*E(87);
dh(174,77)=2*F(87);
dh(177,78)=-2*E(89);
dh(177,79)=-2*F(89);
dh(178,78)=2*E(89);
dh(178,79)=2*F(89);
dh(179,80)=-2*E(90);
dh(179,81)=-2*F(90);
dh(180,80)=2*E(90);
dh(180,81)=2*F(90);
dh(181,82)=-2*E(91);
dh(181,83)=-2*F(91);
dh(182,82)=2*E(91);
dh(182,83)=2*F(91);
dh(18384)=-2*E(92);
dh(183,85)=-2*F(92);
dh(184,84)=2*E(92);
dh(184,85)=2*F(92);
```

```
dh(197,86)=-2*E(99);
dh(197,87)=-2*F(99);
dh(198,86)=2*E(99);
dh(198,87)=2*F(99);
dh(199,88)=-2*E(100);
dh(199,89)=-2*F(100);
dh(200,88)=2*E(100);
dh(200,89)=2*F(100);
dh(205,90)=-2*E(103);
dh(205,91)=-2*F(103);
dh(206,90)=2*E(103);
dh(206,91)=2*F(103);
dh(207,92)=-2*E(104);
dh(207,93)=-2*F(104);
dh(208,92)=2*E(104);
dh(208,93)=2*F(104);
dh(209,94)=-2*E(105);
dh(209,95)=-2*F(105);
dh(210,94)=2*E(105);
dh(210,95)=2*F(105);
dh(213,96)=-2*E(107);
dh(213,97)=-2*F(107);
dh(214,96)=2*E(107);
dh(214,97)=2*F(107);
dh(219,98)=-2*E(110);
dh(219,99)=-2*F(110);
dh(220,98)=2*E(110);
dh(220,99)=2*F(110);
dh(221,100)=-2*E(111);
dh(221,101)=-2*F(111);
dh(222,100)=2*E(111);
dh(222,101)=2*F(111);
dh(223,102)=-2*E(112);
dh(223,103)=-2*F(112);
dh(224,102)=2*E(112);
dh(224,103)=2*F(112);
dh(225,104)=-2*E(113);
dh(225,105)=-2*F(113);
dh(226,104)=2*E(113);
dh(226,105)=2*F(113);
dh(231,106)=-2*E(116);
dh(231,107)=-2*F(116);
dh(232,106)=2*E(116);
dh(232,107)=2*F(116);
```

B.30 Function that computes the Hessian of the constraint functions for the 118-bus system

```
function d2ht_lami=d2ht_lami_118bus(lam_i)

d2ht_lami=2*zeros(107);

d2ht_lami(1,1)=lam_i(2)-lam_i(1);
d2ht_lami(2,2)=lam_i(8)-lam_i(7);
```

```
d2ht_lami(3,3)=lam_i(8)-lam_i(7);
d2ht_lami(4,4)=lam_i(12)-lam_i(11);
d2ht_lami(5,5)=lam_i(12)-lam_i(11);
d2ht_lami(6,6)=lam_i(16)-lam_i(15);
d2ht_lami(7,7)=lam_i(16)-lam_i(15);
d2ht_lami(8,8)=lam_i(20)-lam_i(19);
d2ht_lami(9,9)=lam_i(20)-lam_i(19);
d2ht_lami(10,10)=lam_i(24)-lam_i(23);
d2ht_lami(11,11)=lam_i(24)-lam_i(23);
d2ht_lami(12,12)=lam_i(30)-lam_i(29);
d2ht_lami(13,13)=lam_i(30)-lam_i(29);
d2ht_lami(14,14)=lam_i(36)-lam_i(35);
d2ht_lami(15,15)=lam_i(36)-lam_i(35);
d2ht_lami(16,16)=lam_i(38)-lam_i(37);
d2ht_lami(17,17)=lam_i(38)-lam_i(37);
d2ht_lami(18,18)=lam_i(48)-lam_i(47);
d2ht_lami(19,19)=lam_i(48)-lam_i(47);
d2ht_lami(20,20)=lam_i(50)-lam_i(49);
d2ht_lami(21,21)=lam_i(50)-lam_i(49);
d2ht_lami(22,22)=lam_i(52)-lam_i(51);
d2ht_lami(23,23)=lam_i(52)-lam_i(51);
d2ht_lami(24,24)=lam_i(54)-lam_i(53);
d2ht_lami(25,25)=lam_i(54)-lam_i(53);
d2ht_lami(26,26)=lam_i(62)-lam_i(61);
d2ht_lami(27,27)=lam_i(62)-lam_i(61);
d2ht_lami(28,28)=lam_i(64)-lam_i(63);
d2ht_lami(29,29)=lam_i(64)-lam_i(63);
d2ht_lami(30,30)=lam_i(68)-lam_i(67);
d2ht_lami(31,31)=lam_i(68)-lam_i(67);
d2ht_lami(32,32)=lam_i(72)-lam_i(71);
d2ht_lami(33,33)=lam_i(72)-lam_i(71);
d2ht_lami(34,34)=lam_i(80)-lam_i(79);
d2ht_lami(35,35)=lam_i(80)-lam_i(79);
d2ht_lami(36,36)=lam_i(84)-lam_i(83);
d2ht_lami(37,37)=lam_i(84)-lam_i(83);
d2ht_lami(38,38)=lam_i(92)-lam_i(91);
d2ht_lami(39,39)=lam_i(92)-lam_i(91);
d2ht_lami(40,40)=lam_i(98)-lam_i(97);
d2ht_lami(41,41)=lam_i(98)-lam_i(97);
d2ht_lami(42,42)=lam_i(108)-lam_i(107);
d2ht_lami(43,43)=lam_i(108)-lam_i(107);
d2ht_lami(44,44)=lam_i(110)-lam_i(109);
d2ht_lami(45,45)=lam_i(110)-lam_i(109);
d2ht_lami(46,46)=lam_i(112)-lam_i(111);
d2ht_lami(47,47)=lam_i(112)-lam_i(111);
d2ht_lami(48,48)=lam_i(118)-lam_i(117);
d2ht_lami(49,49)=lam_i(118)-lam_i(117);
d2ht_lami(50,50)=lam_i(122)-lam_i(121);
d2ht_lami(51,51)=lam_i(122)-lam_i(121);
d2ht_lami(52,52)=lam_i(124)-lam_i(123);
d2ht_lami(53,53)=lam_i(124)-lam_i(123);
d2ht_lami(54,54)=lam_i(130)-lam_i(129);
d2ht_lami(55,55)=lam_i(130)-lam_i(129);
d2ht_lami(56,56)=lam_i(132)-lam_i(131);
```

```
d2ht_lami(57,57)=lam_i(132)-lam_i(131);
d2ht_lami(58,58)=lam_i(138)-lam_i(137);
d2ht_lami(59,59)=lam_i(138)-lam_i(137);
d2ht_lami(60,60)=lam_i(140)-lam_i(139);
d2ht_lami(61,61)=lam_i(140)-lam_i(139);
d2ht_lami(62,62)=lam_i(144)-lam_i(143);
d2ht_lami(63,63)=lam_i(144)-lam_i(143);
d2ht_lami(64,64)=lam_i(146)-lam_i(145);
d2ht_lami(65,65)=lam_i(146)-lam_i(145);
d2ht_lami(66,66)=lam_i(148)-lam_i(147);
d2ht_lami(67,67)=lam_i(148)-lam_i(147);
d2ht_lami(68,68)=lam_i(152)-lam_i(151);
d2ht_lami(69,69)=lam_i(152)-lam_i(151);
d2ht_lami(70,70)=lam_i(154)-lam_i(153);
d2ht_lami(71,71)=lam_i(154)-lam_i(153);
d2ht_lami(72,72)=lam_i(160)-lam_i(159);
d2ht_lami(73,73)=lam_i(160)-lam_i(159);
d2ht_lami(74,74)=lam_i(170)-lam_i(169);
d2ht_lami(75,75)=lam_i(170)-lam_i(169);
d2ht_lami(76,76)=lam_i(174)-lam_i(173);
d2ht_lami(77,77)=lam_i(174)-lam_i(173);
d2ht_lami(78,78)=lam_i(178)-lam_i(177);
d2ht_lami(79,79)=lam_i(178)-lam_i(177);
d2ht_lami(80,80)=lam_i(180)-lam_i(179);
d2ht_lami(81,81)=lam_i(180)-lam_i(179);
d2ht_lami(82,82)=lam_i(182)-lam_i(181);
d2ht_lami(83,83)=lam_i(182)-lam_i(181);
d2ht_lami(84,84)=lam_i(184)-lam_i(183);
d2ht_lami(85,85)=lam_i(184)-lam_i(183);
d2ht_lami(86,86)=lam_i(198)-lam_i(197);
d2ht_lami(87,87)=lam_i(198)-lam_i(197);
d2ht_lami(88,88)=lam_i(200)-lam_i(199);
d2ht_lami(89,89)=lam_i(200)-lam_i(199);
d2ht_lami(90,90)=lam_i(206)-lam_i(205);
d2ht_lami(91,91)=lam_i(206)-lam_i(205);
d2ht_lami(92,92)=lam_i(208)-lam_i(207);
d2ht_lami(93,93)=lam_i(208)-lam_i(207);
d2ht_lami(94,94)=lam_i(210)-lam_i(209);
d2ht_lami(95,95)=lam_i(210)-lam_i(209);
d2ht_lami(96,96)=lam_i(214)-lam_i(213);
d2ht_lami(97,97)=lam_i(214)-lam_i(213);
d2ht_lami(98,98)=lam_i(220)-lam_i(219);
d2ht_lami(99,99)=lam_i(220)-lam_i(219);
d2ht_lami(100,100)=lam_i(222)-lam_i(221);
d2ht_lami(101,101)=lam_i(222)-lam_i(221);
d2ht_lami(102,102)=lam_i(224)-lam_i(223);
d2ht_lami(103,103)=lam_i(224)-lam_i(223);
d2ht_lami(104,104)=lam_i(226)-lam_i(225);
d2ht_lami(105,105)=lam_i(226)-lam_i(225);
d2ht_lami(106,106)=lam_i(232)-lam_i(231);
d2ht_lami(107,107)=lam_i(232)-lam_i(231);
```

B.31 Function that computes the Jacobian and Hessian of the Lagrangian of the VVO problem for the 118-bus system

```
function [f, h, dh, gL, g2L]=f_118bus(x, lam_i)

% Define objective function, its gradient and hessian [f, df, d2f]:

%[~, f, df, d2f, ~]=loss_func_sym_118bus();
[f, ~]=loss_func();
df=df_118bus();
d2f=d2f_118bus();

% Define inequality constraints, the Jacobian and hessian [h, dht_lam_i,
d2ht_lam_i]:
%[h, dh, dht_lam_i, d2ht_lam_i, ~, ~]=h_gradh_hessh_118bus_sym(x, lam_i);
h=h_118bus(x);
dh=dh_118bus();
dht_lam_i=dht_lami_118bus(lam_i);
d2ht_lam_i=d2ht_lami_118bus(lam_i);

% Define gradient and Hessian of Lagrangian, gL, g2L:
% gL=-df+dht_lam_i;
% g2L=d2f-d2ht_lam_i;

gL=-df+dht_lam_i;
g2L=d2f+d2ht_lam_i;
```

B.32 MATLAB script that runs the PDIPM-VVO algorithm for the 118_bus system

```
clear

close all
clc

% File name: one_hundred_eighteen_bus_system_pdipm_vvo_test.m

% List of functions needed to run this program:
% (further details are provided for each of these functions
%  where they are first called in the program):

% 1.  [G, B]=ybus(R, X, Cf)
% 2.  [R, X, Cf]=computeRX(from, to, r, x)
% 3.  x0=define_x0()
% 4.  dF=dF(x, Vgref)
% 5.  J=jacobian(x)
% 6.  [V, output]=NR_load_flow(@dF, @jacobian, x0, Vgref)
% 7.  h=h_118bus(x)
% 8.  [f, df, d2f]=loss_func_118bus()
% 9.  [h, dh, dht_lam_i, d2ht_lam_i]=h_gradh_hessh_118bus(x, lam_i)
% 10. [f, h, dh, gL, g2L]=f_118bus(x, lam_i)
% 11. update_generator_voltages(x)
% 12. [f, df]=loss_func()
```

```
% 13. df=df_118bus()
% 14. d2f=d2f_118bus()
% 15. h=h_118bus(x)
% 16. dh=dh_118bus()
% 17. dht_lami=dht_lami_118bus(lam_i)
% 18. d2ht_lami=d2ht_lami_118bus(lam_i)
% 19. x=update_control_vector()
% 20. x0=define_updated_x0(x)
% 21. Vgref=updated_Vgref1(x)
% 22. [X,adx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu,fval, cond, output]=...
%         ipm_118bus(@f_118bus,x0,s0,mu0, sigma, @h_118bus, @dF, @jacobian,...
%         Vgref, maxIter)
%
% Bus Data:
% Volt/VAR optimization for the IEEE 118-bus system:
%
% Number of buses       : 118;
% Number of lines       : 186;
% Number of generators  : 54;
% Number of loads       : 99

global bus_data Cf E F G B bus_type nbus V1_angle_ratio PQs

% bus_data is matrix in which each row applies to a bus, and specifies:
% (1) bus type (1=slack bus, 2=PV bus, 3=PQ bus)
% (2) Real component of bus voltage (E)
% (3) Imaginary component of bus voltage (F)
% (4) Generated real power (Pg)
% (5) Generated reactive power (Qg)
% (6) Real power demand (Pd)
% (7) Reactive power demand (Qd)

% Cf is the bus connectivity matrix; Cf(i,j)=1 when buses i and j are
% connected, zero otherwise

% E and F are vectors containing real and imaginary components of the
% bus voltages (in rectangular coordinates)

% G and B are conductance and susceptance matrices respectively;
% i.e. Y=G+jB, where Y is the bus admittance matrix

% bus_type is simply the first column of the bus_data matrix

% nbus is the number of buses in the system

% Values are all in per-unit

%=====================================================================
%     Bus type    V_real     V_imag     Pg      Qg      Pd          Qd
bus_data=[1       .8963     .5175       0       0       0           0  ;...
          3       .971         0        0       0      .2          .09  ;...
          3       .968         0        0       0      .39         .1   ;...
          2       .998         0        0       0      .39         .12  ;...
          3      1.002         0        0       0       0           0   ;...
```

```
2       .99         0        0       0      .52         .22  ;...
3       .989        0        0       0      .19         .02  ;...
2      1.015        0        0       0      .28          0   ;...
3      1.043        0        0       0       0           0   ;...
2      1.05         0      4.5       0       0           0   ;...
3       .985        0        0       0      .7          .23  ;...
2       .99         0       .85      0      .47         .1   ;...
3       .968        0        0       0      .34         .16  ;...
3       .984        0        0       0      .14         .01   ;...
2       .97         0        0       0      .9          .3   ;...
3       .984        0        0       0      .25         .1   ;...
3       .995        0        0       0      .11         .03  ;...
2       .973        0        0       0      .6          .34  ;...
2       .963        0        0       0      .45         .25  ;...
3       .958        0        0       0      .18         .03  ;...
3       .959        0        0       0      .14         .08  ;...
3       .97         0        0       0      .1          .05  ;...
3         1         0        0       0      .07         .03  ;...
2       .992        0        0       0      .13          0   ;...
2      1.05         0      2.2       0       0           0   ;...
2      1.015        0     3.14       0       0           0   ;...
2       .968        0        0       0      .71         .13  ;...
3       .962        0        0       0      .17         .07  ;...
3       .963        0        0       0      .24         .04  ;...
3       .968        0        0       0       0          0   ;...
2       .967        0       .07      0      .43         .27   ;...
2       .964        0        0       0      .59         .23   ;...
3       .972        0        0       0      .23         .09   ;...
2       .986        0        0       0      .59         .26   ;...
3       .981        0        0       0      .33         .09   ;...
2       .98         0        0       0      .31         .17   ;...
3       .992        0        0       0       0          0   ;...
3       .962        0        0       0       0          0   ;...
3       .97         0        0       0      .27         .11   ;...
2       .97         0        0       0      .66         .23   ;...
3       .967        0        0       0      .37         .1   ;...
2       .985        0        0       0      .96         .23   ;...
3       .978        0        0       0      .18         .07   ;...
3       .985        0        0       0      .16         .08   ;...
3       .987        0        0       0      .53         .22   ;...
2      1.005        0       .19      0      .28         .1   ;...
3      1.017        0        0       0      .34          0   ;...
3      1.021        0        0       0      .2          .11   ;...
2      1.025        0     2.04       0      .87         .3   ;...
3      1.001        0        0       0      .17         .04   ;...
3       .967        0        0       0      .17         .08   ;...
3       .957        0        0       0      .18         .05   ;...
3       .946        0        0       0      .23         .11   ;...
2       .955        0       .48      0     1.13         .32   ;...
2       .952        0        0       0      .63         .22   ;...
2       .954        0        0       0      .84         .18   ;...
3       .971        0        0       0      .12         .03   ;...
3       .959        0        0       0      .12         .03   ;...
2       .985        0     1.55       0     2.77        1.13   ;...
```

```
3       .993        0        0       0      .78         .03   ;...
2       .995        0      1.6       0       0           0    ;...
2       .998        0        0       0      .77         .14   ;...
3       .969        0        0       0       0           0    ;...
3       .984        0        0       0       0           0    ;...
2      1.005        0     3.91       0       0           0    ;...
2       1.05        0     3.92       0      .39         .18   ;...
3       1.02        0        0       0      .28         .07   ;...
3      1.003        0        0       0       0           0    ;...
2       .955        0        0       0      .51         .27   ;...
2       .984        0        0       0      .66         .2    ;...
3       .987        0        0       0       0           0    ;...
2        .98        0        0       0      .12          0    ;...
2       .991        0        0       0      .06          0    ;...
2       .958        0        0       0      .68         .27   ;...
3       .967        0        0       0      .47         .11   ;...
2       .943        0        0       0      .68         .36   ;...
2      1.006        0        0       0      .61         .28   ;...
3      1.003        0        0       0      .71         .26   ;...
3      1.009        0        0       0      .39         .32   ;...
2       1.04        0     4.77       0      1.3         .26   ;...
3       .997        0        0       0       0           0    ;...
3       .989        0        0       0      .54         .27   ;...
3       .985        0        0       0      .2          .1    ;...
3        .98        0        0       0      .11         .07   ;...
2       .985        0        0       0      .24         .15   ;...
3       .987        0        0       0      .21         .1    ;...
2      1.015        0      .04       0       0           0    ;...
3       .987        0        0       0      .48         .1    ;...
2      1.005        0     6.07       0       0           0    ;...
2       .985        0        0       0     1.63         .42   ;...
2        .98        0        0       0      .1          0    ;...
2       .993        0        0       0      .65         .1    ;...
3       .987        0        0       0      .12         .07   ;...
3       .991        0        0       0      .3          .16   ;...
3       .981        0        0       0      .42         .31   ;...
3       .993        0        0       0      .38         .15   ;...
3      1.011        0        0       0      .15         .09   ;...
3      1.024        0        0       0      .34         .08   ;...
2       1.01        0        0       0      .42          0    ;...
2      1.017        0     2.52       0      .37         .18   ;...
3       .993        0        0       0      .22         .15   ;...
3       .991        0        0       0      .05         .03   ;...
2      1.001        0       .4       0      .23         .16   ;...
2       .971        0        0       0      .38         .25   ;...
2       .965        0        0       0      .31         .26   ;...
3       .962        0        0       0      .43         .16   ;...
2       .952        0        0       0      .5          .12   ;...
3       .967        0        0       0      .02         .01   ;...
3       .967        0        0       0      .08         .03   ;...
2       .973        0        0       0      .39         .3    ;...
2        .98        0      .36       0       0           0    ;...
2       .975        0        0       0      .68         .13   ;...
2       .993        0        0       0      .06          0    ;...
```

```
          3       .96          0        0       0      .08          .03   ;...
          3       .96          0        0       0      .22          .07   ;...
          2      1.005         0        0       0     1.84           0    ;...
          3       .974         0        0       0      .2           .08   ;...
          3       .949         0        0       0      .33          .15];
%=========================================================================

bus_type=bus_data(:,1);
nbus=length(bus_type);

% Bus 44 consistently exhibits low voltage (<0.95)
% Adjusting initial voltage and reactive power injection
% seemed to result in some slight improvement
% bus_data(44,2)=.99;
% bus_data(44,end)=-.35;


% Uniformly distributed generation of 0.68 seems to result in slight
% reduction in losses and improvement in voltage magnitude, but voltage
% phase angles are too high (most beyond 45 degrees maximum specification)
for i=2:nbus
    if (bus_type(i)==2)
        bus_data(i,4)=.68;
    end
end
% figure(4), plot(bus_data(:,4)), title('Uniformly distributed power generation');
% xlabel('bus number'), ylabel('Amplitude')

% Line Data:
% Corresponding elements of the from_bus and to_bus vectors are the bus
% pairs of connected buses; a line or branch (i,j) exists between bus
% pairs (from_bus(i), to_bus(j));
% r and x are vectors of line resistance and reactance values,
% corresponding to the lines specified by (from_bus(i), to_bus(j))

from1=[1 1 4 3 5 6 8 8 9 4];
to1=[2 3 5 5 6 7 9 5 10 11];
r1=[.0303 .0129 .0018 .0241 .0119 .0046 .0024 0 .0026 .0209];
x1=[.0999 .0424 .008 .108 .054 .0208 .0305 .0267 .0322 .0688];

from2=[5 11 2 3 7 11 12 13 14 12];
to2=[11 12 12 12 12 13 14 15 15 16];
r2=[.0203 .006 .0187 .0484 .0086 .0223 .0215 .0744 .0595 .0212];
x2=[.0682 .0196 .0616 .16 .034 .0731 .0707 .2444 .195 .0834];

from3=[15 16 17 18 19 15 20 21 22 23];
to3=[17 17 18 19 20 19 21 22 23 24];
r3=[.0132 .0454 .0123 .0112 .0252 .012 .0183 .0209 .0342 .0135];
x3=[.0437 .1801 .0505 .0493 .117 .0394 .0849 .097 .159 .0492];

from4=[23 26 25 27 28 30 8 17 29 23];
to4=[25 25 27 28 29 17 30 31 31 32];
r4=[.0156 0 .0318 .0191 .0237 0 .0043 .0474 .0108 .0317];
x4=[.08 .0382 .163 .0855 .0943 .0388 .0504 .1563 .0331 .1153];
```

```
from5=[31 27 15 19 35 35 33 34 34 37];
to5=[32 32 33 34 36 37 37 36 37 39];
r5=[.0298 .0229 .038 .0752 .0022 .011 .0415 .0087 .0026 .0321];
x5=[.0985 .0755 .1244 .247 .0102 .0497 .142 .0268 .0094 .106];

from6=[37 30 39 40 40 41 43 34 44 45];
to6=[40 38 40 41 42 42 44 43 45 46];
r6=[.0593 .0046 .0184 .0145 .0555 .041 .0608 .0413 .0224 .04];
x6=[.168 .054 .0605 .0487 .183 .135 .2454 .1681 .0901 .1356];

from7=[46 46 47 42 42 45 48 49 49 51];
to7=[47 48 49 49 49 49 49 50 51 52];
r7=[.038 .0601 .0191 .0715 .0715 .0684 .0179 .0267 .0486 .0203];
x7=[.127 .189 .0625 .323 .323 .186 .0505 .0752 .137 .0588];

from8=[52 53 49 49 54 54 55 56 50 56];
to8=[53 54 54 54 55 56 56 57 57 58];
r8=[.0405 .0263 .073 .0869 .0169 .0027 .0049 .0343 .0474 .0343];
x8=[.1635 .122 .289 .291 .0707 .0095 .0151 .0966 .134 .0966];

from9=[51 54 56 56 55 59 59 60 60 61];
to9=[58 59 59 59 59 60 61 61 62 62];
r9=[.0255 .0503 .0825 .0803 .0474 .0317 .0328 .0026 .0123 .0082];
x9=[.0719 .2293 .251 .239 .2158 .145 .15 .0135 .0561 .0376];

from10=[63 63 64 38 64 49 49 62 62 65];
to10=[59 64 61 65 65 66 66 66 67 66];
r10=[0 .0017 0 .009 .0027 .018 .018 .0482 .0258 0];
x10=[.0386 .02 .0268 .0986 .0302 .0919 .0919 .218 .117 .037];

from11=[66 47 49 68 69 24 70 24 71 71];
to11=[67 69 69 69 70 70 71 72 72 73];
r11=[.0224 .0844 .0985 0 .03 .0022 .0088 .0488 .0446 .0087];
x11=[.1015 .2778 .324 .037 .127 .4115 .0355 .196 .18 .0454];

from12=[70 70 69 74 76 69 75 77 78 77];
to12=[74 75 75 75 77 77 77 78 79 80];
r12=[.0401 .0428 .0405 .0123 .0444 .0309 .0601 .0038 .0055 .017];
x12=[.1323 .141 .122 .0406 .148 .101 .1999 .0124 .0244 .0485];

from13=[77 79 68 81 77 82 83 83 84 85];
to13=[80 80 81 80 82 83 84 85 85 86];
r13=[.0294 .0156 .0018 0 .0298 .0112 .0625 .043 .0302 .035];
x13=[.105 .0704 .0202 .037 .0853 .0366 .132 .148 .0641 .123];

from14=[86 85 85 88 89 89 90 89 89 91];
to14=[87 88 89 89 90 90 91 92 92 92];
r14=[.0283 .02 .0239 .0139 .0518 .0238 .0254 .0099 .0393 .0387];
x14=[.2074 .102 .173 .0712 .188 .0997 .0836 .0505 .1581 .1272];

from15=[92 92 93 94 80 82 94 80 80 80];
to15=[93 94 94 95 96 96 96 97 98 99];
r15=[.0258 .0481 .0223 .0132 .0356 .0162 .0269 .0183 .0238 .0454];
```

```
x15=[.0848 .158 .0732 .0434 .182 .053 .0869 .0934 .108 .206];

from16=[92 94 95 96 98 99 100 92 101 100];
to16=[100 100 96 97 100 100 101 102 102 103];
r16=[.0648 .0178 .0171 .0173 .0397 .018 .0277 .0123 .0246 .016];
x16=[.295 .058 .0547 .0885 .179 .0813 .1262 .0559 .112 .0525];

from17=[100 103 103 100 104 105 105 105 106 108];
to17=[104 104 105 106 105 106 107 108 107 109];
r17=[.0451 .0466 .0535 .0605 .0099 .014 .053 .0261 .053 .0105];
x17=[.204 .1584 .1625 .229 .0378 .0547 .183 .0703 .183 .0288];

from18=[103 109 110 110 17 32 32 27 114 68];
to18=[110 110 111 112 113 113 114 115 115 116];
r18=[.0391 .0278 .022 .0247 .0091 .0615 .0135 .0164 .0023 .0003];
x18=[.1813 .0762 .0755 .064 .0301 .203 .0612 .0741 .0104 .0041];

from19=[12 75 76 38 26 65];
to19=[117 118 118 37 30 68];
r19=[.0329 .0145 .0164 0 .008 .0014];
x19=[.14 .0481 .0544 .0375 .086 .016];

% Combine the vectors:
from_bus=[from1 from2 from3 from4 from5 from6 from7 from8 from9 from10 ...
    from11 from12 from13 from14 from15 from16 from17 from18 from19]';

to_bus=[to1 to2 to3 to4 to5 to6 to7 to8 to9 to10 to11 to12 to13 to14 ...
    to15 to16 to17 to18 to19]';

r=[r1 r2 r3 r4 r5 r6 r7 r8 r9 r10 r11 r12 r13 r14 r15 r16 r17 r18 r19]';

x=[x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16 x17 x18 x19]';

% Compute the admittance (in the form G+jB) and bus connectivity (Cf)
% matrices
% Function compute_Ybus() calls function computeRX(), which computes
% the resistance (R) and reactance (X) vectors, needed by the
% function compute_Ybus(), as well as matrix Cf

[R, X, Cf]=computeRX(from_bus, to_bus, r, x);

% Lines 42-49, 49-54, 49-66, 56-59, 77-80, 89-90 and 89-92 are double
% (parallel) lines; calculate the effective impedande and update the
% R, X vectors:
R(42,49)=.5*.0715;
R(49,42)=R(42,49);
X(42,49)=.5*.323;
X(49,42)=X(42,49);

R(49,54)=.073*.0869/(.073+.0869);
R(54,49)=R(49,54);
X(49,54)=.289*.291/(.289+.291);
X(54,49)=X(49,54);
```

```
R(49,66)=.5*.018;
R(66,49)=R(49,66);
X(49,66)=.5*.0919;
X(66,49)=X(49,66);

R(56,59)=.0803*.0825/(.0803+.0825);
R(59,56)=R(56,59);
X(56,59)=.239*.251/(.239+.251);
X(59,56)=X(56,59);

R(77,80)=.017*.0294/(.018+.0294);
R(80,77)=R(77,80);
X(77,80)=.0485*.105/(.0485+.105);
X(80,77)=X(77,80);

R(89,90)=.0518*.0238/(.0518+.0238);
R(90,89)=R(89,90);
X(89,90)=.188*.0997/(.188+.0997);
X(90,89)=X(89,90);

R(89,92)=.0099*.0393/(.0099+.0393);
R(92,89)=R(89,92);
X(89,92)=.0505*.1581/(.0505+.1581);
X(92,89)=X(89,92);

% Compute the admittance and bus connectivity matrices:

[G, B]=ybus(R, X, Cf);

% Define some parameters:

% Extract the vectors of real and imaginary bus voltage components
% from the bus_data matrix

E=bus_data(:,2);
F=bus_data(:,3);

% [E1, F1]=set_slack_bus_voltage();
% E(1)=E1(8);
% F(1)=F1(8);

% To maintain slack-bus voltage angle:
V1_angle_ratio=F(1)/E(1);

% Define initial input to Newton-Raphson load flow algorithm
% (initial bus voltages,in rectangular form):

x0=define_x0();

% Define the reference voltage vector for the generator voltages;
% this is required for the Volt/VAR optimization problem when running
% the load flow algorithm at each Newton method iteration

Vgref=Vgref_0();
```

```
% Perform Newton-Raphson load flow
% Newton-Raphson algorithm implemented in rectangular coordinates of
% bus voltages; assumes that bus 1 is the (only) slack bus. Calls functions
% dF() of residues (of the real and reactive power/voltage balance
% equations), as well as jacobian(), which computes the Jacobian of the
% residues, needed to compute the Newton step once the algorithm
% (hopefully) converges, the bus voltages are output in polar form
% (i.e. magnitude and phase angle)

% tic
% [~, output]=NR_load_flow(@dF, @jacobian_118bus, x0, Vgref);
% toc
%
% v=[output.V(:,1) output.V(:,2)]
%
% % Compute system losses after the load flow algorithm has terminated
%
% [losses, ~]=loss_func();


% Perform Volt/VAR optimization:
% Interior-Point Method (IPM)-based Volt/VAR optimization (VVO); applies
% the Newton method to compute the search direction for the primal-dual
% system of the VVO problem derived on the basis of the perturbed KKT
% (first-order) optimality conditions.
%
% Computation of the Newton step requires calculating the first- and
% second-order partial derivatives of the objective and constraint
% functions. The IPM algorithm only considers the inequality constraints
% (in this implementation only the bus voltage magnitude constraints),
% the equality constraints (real and reactive power balance equations)
% are handled by the (Newton-Raphson) load flow algorithm. Therefore, at
% each iteration of the Newton step of the IPM algorithm, the load flow
% algorithm is executed once the primal and dual variables have been
% updated.
%
% The IPM algorithm needs the following functions to execute:
%
% 1. f_118bus(), which computes the gradient and hessian of the Lagrangian
%      function of the VVO problem, which are required to compute the
%      Newton step;
% 2.  loss_func(), df_118bus(), and d2f_118bus(), which are called by
%     f_118bus(), to compute the objective, its gradient and hessian;
% 3.  h_118bus(), dh_118bus(), dht_lami_118bus(), and d2ht_lami_118bus(),
%     which are also called by f_118bus, to compute the constraint
%     functions, the Jacobian and hessian thereof as well;
% 4.  dF(), which computes the residues needed by the Newton-Raphson
%     load flow algorithm;
% 5.  jacobian(), which computes the Jacobian of the residues, also needed
%     by the Newton-Raphson load flow algorithm;
% 6.  A number of utility functions called by the functions stated above,
%     including update_generator_voltages(), update_control_vector(),
%     define_updated_x0(), updated_Vgref()
```

```
%
%
% The IPM algorithm also requires a number of parameters, such as the
% barrier parameter (mu), the centering parameter (zeta), and the
% choice of initial primal and dual variables.

% Initialize some input parameters:
xc=update_control_vector();
h0=h_118bus(x0);
s0=abs(h0);
s0(s0<0.01)=.01;
% choice of initial values of slack variables seems to significantly
% affect converge of the algorithm. s0 1.15*s0 gives better results,
% with a slight loss reduction (1.6%); requires limiting number of% Newton
% iterations to 8. But it causes divergence when the number of iterations
% is increased a lot (to say, 20)
s0=1.15*s0;
% mu0=9;
% sigma=.065;
mu0=10;
sigma=.15;
maxIt=7;

s=s0;
mu=mu0;
niq=length(h0);
e=ones(niq,1);
S=diag(s);
Si=inv(S);
lam_i=mu*Si*e;
%[h, dh, dht_lam_i, d2ht_lam_i, hh, nonzeroh]=h_gradh_hessh_118bus_sym(x, lam_i);

% Run the IPM algorithm on the VVO problem:

tic
[X,adx,s,ds,lam_i,dlam_i, alpha_p, alpha_d, mu,fval, cond, output]=...
    ipm_118bus5(@f_118bus,x0,s0,mu0, sigma, @h_118bus, @dF, @jacobian_118bus,
Vgref, maxIt, V1_angle_ratio);
toc

% Output some computation results:

X
mu
cond
output
V=output.V

% Compute the loss reduction:

initial_loss_pu=-cond.loss(1)
final_loss_pu=-cond.loss(end)
loss_reduction_percentage=100*(cond.loss(1)-cond.loss(end))/cond.loss(1)
```

```
% Plot the loss reduction vs. the iteration number:


figure(1), plot(cond.V1(:,1)), hold on, plot(cond.V1(:,2),'r'),hold on,
plot(cond.V1(:,2)./cond.V1(:,1),'k--')
legend('E','F','F/E');
title('Slack-bus voltage (real(E) and imaginary(F) components)');
xlabel('Iteration number')
ylabel('V_{slack-pu}')
V1_mag=sqrt(cond.V1(:,1).^2+cond.V1(:,2).^2);
V1_angle=180/pi*atan(cond.V1(:,2)./cond.V1(:,1));
V1=[V1_mag V1_angle]

figure(2)
Vinit=cond.Vinit;
Vfinal=V(:,1);
plot(Vinit,'b-s'), hold on, plot(Vfinal,'r-s'),legend('V_{Initial}','V_{Final}')
grid
title('Voltage profile, final compared with initial');
xlabel('Bus number')
ylabel('V_{bus-pu}')

% extract generator bus voltages:
Vall=[Vinit Vfinal];

gen_bus_idx=find(bus_type==1 | bus_type==2);
ng=length(gen_bus_idx);
Vgen=zeros(ng,3);

for i=1:ng
    Vgen(i,:)=[gen_bus_idx(i) Vall(gen_bus_idx(i), :)];
end

% xlswrite('one_eighteen_bus_system_voltage1.xlsx', Vgen)

figure(3)
plot(cond.iter,-cond.loss, 'r',cond.iter,-cond.loss, 'b*');
grid
title('IEEE 118-bus system real power loss vs. iteration number', 'FontSize', 10.0)
xlabel('Iteration number')
ylabel('P_{loss} (p.u.)')
loss_label=['Percentage loss reduction = ' num2str(loss_reduction_percentage) '%'];

ax=gca;
y_lims=ax.YLim;
text(2,y_lims(2)-.025, loss_label);

loss=-cond.loss;
% Adding 2e-2 to P1 makes slack-bus active power
% and power loss coincide; this (value of 2e-2)
% seems to only represent a discrepancy due to
% differences in scale of the two quantities
P1=.02+cond.PQs(1,:)';
Q1=cond.PQs(2,:)';
```

```
Q4=cond.PQs(3,:)';
% 4.8e-1 added to Qsum to shift it up, improves comparison with
% real power loss reduction
Qsum=.48+cond.Qsum';

x1=0:length(loss)-1;
x2=0:length(P1)-1;

figure(4)
subplot(221)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('Slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid

subplot(222)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
grid

subplot(223)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q4);
title('Bus-4 generator reactive power and power loss')
legend('P_{loss}','Q_{g4}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-4 generator reactive power');
hLine2.LineStyle='--';
grid

subplot(224)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Qsum);
title('Total generated reactive power and power loss')
legend('P_{loss}','Q_{gen}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Total generated reactive power');
hLine2.LineStyle='--';
grid

figure(5)
subplot(211)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('118-bus system: slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
```

```
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid

ax=gca;
y_lims=ax.YLim;
text(.7,y_lims(2)-.025, loss_label);

subplot(212)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Qsum);
title('118-bus system: total generated reactive power and power loss')
legend('P_{loss}','Q_{gen}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Total generated reactive power');
hLine2.LineStyle='--';
grid


% figure(5)
% [hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
% title('Fig.5: Slack-bus active power and power loss')
% legend('P_{loss}','P_{slack}')
% xlabel('Iteration number')
% ylabel(hAx(1),'Real power loss');
% ylabel(hAx(2),'Slack-bus real power');
% hLine2.LineStyle='-.';
% grid


figure(6)
subplot(211)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
grid

x3=[0:length(V1_mag)-1]';

subplot(212)
[hAx, ~, hLine2]=plotyy(x2,Q1,x3,V1_mag);
title('Slack-bus reactive power and Slack-bus voltage')
legend('Q_{slack}','V_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Slack-bus voltage');
hLine2.LineStyle='--';
grid
```

```
figure(7)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Qsum);
title('Total generated reactive power and power loss')
legend('P_{loss}','Q_{gen}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Total generated reactive power');
hLine2.LineStyle='--';
grid
```

APPENDIX C: SOFTWARE PROGRAMS FOR CHAPTER 6

C.1 MATLAB script that runs the PSO-VVO algorithm for the 3-bus system

```
% File name: pso_vvo_3bus_system.m

% load 3-bus system data:
three_bus_system_data;
global PQs Qsum Vgens

% initialize PSO parameters:
```

```
c1=2.05;
c2=2.05;
% c1=2;
% c2=2;
p=10;       % swarm size
d=3;        % problem dimension
N=200;       % maximum number of iterations
wmin=.4;    % minimum inertia weight
wmax=.9;    % maximum inertia weight

% generate initial position vector:
e10=.95+.15*rand(p,1);
e30=.95+.15*rand(p,1);
f30=real(sqrt(1-(.95+.15*rand(p,1)).^2));
x=[e10 e30 f30];

% generate initial velocity vector (set to zero):
v=zeros(p,d);

fprev=inf*ones(p,1);        % initialize fitness function values (to infinity)
pbest=x;                    % initialize each particle's past best position (equal
to initial position)
Gbest=[];                   % keep track of global best position
fbest=[];                   % keep track of fitness value of global best position

% evaluate fitness function for each particle to determine personal and global best
positions:
[fprev,pbest,gbest,best_idx]=PSO_compute_pbest_gbest(x,fprev,pbest);

% run (Newton-Raphson) load flow:
% compute initial voltage vector for the load flow algorithm:
x_loadflow=define_updated_x0(gbest);

% compute voltage reference vector for the load flow algorithm:
Vgref_loadflow=updated_Vgref(gbest);

% run the load flow algorithm:
[V, output]=NR_load_flow(@dF, @jacobian, x_loadflow, Vgref_loadflow);
E=real(V);
F=imag(V);

% Initialize slack-bus active, and generator reactive power outputs
% matrix to empty matrix:
PQs=[];

% Compute slack-bus active, and generator reactive power outputs
compute_PQ();

% Vector to hold generator bus voltages:
Vgens=[];

% Extract generator bus voltages;
get_Vgen();
```

```
% Initial load flow voltage results:
Vinit=output.V(:,1);


% update control vector (gbest):
gbest=update_control_vector()';

% recompute objective function:
f_best=PSO_objective_evaluation(gbest);

% keep track of global best position and its correspoding fitness/objective
function value:
Gbest=[Gbest;gbest];
fbest=[fbest;f_best];

% keep tract of change in global best position and its corresponding
fitness/objective value:
dGbest=[];
dfbest=[];

% compute termination conditions based on change in global best position, and its
associated fitness value:
objective_norm_change=(abs(f_best)/(1+abs(f_best))<1e-5);
global_position_change_norm=(norm(gbest)<1e-5);
objective_change=abs(f_best)>=0;

% initialize iteration counter:
t=1;

tic     % time the PSO loop
% loop until termination conditions are satisfied:
while ((~objective_norm_change || objective_change) && t<=N)

    % compute the particle velocity:

    % based on inertia weight:
    %[V,X]=PSO_X_update2(V,X,pbest,gbest,c1,c2,r1,r2,wmin,wmax,t,N);

    % based on constriction coefficient:
    [v,x]=PSO_X_update_cc1(v,x,pbest,gbest,c1,c2);

    % compute the objective function value for each particle to
    % update the personal and global best positions:
    [fprev,pbest,gbest,best_idx]=PSO_compute_pbest_gbest(x,fprev,pbest);

    % re-run (Newton-Raphson) load flow:
    % compute initial voltage vector for the load flow algorithm:
    x_loadflow=define_updated_x0(gbest);

    % compute voltage reference vector for the load flow algorithm:
    Vgref_loadflow=updated_Vgref(gbest);

    % run the load flow algorithm:
    [V, output]=NR_load_flow(@dF, @jacobian, x_loadflow, Vgref_loadflow);
```

```
    E=real(V);
    F=imag(V);

    % Compute slack-bus active, and generator reactive power outputs
    compute_PQ();

    % Extract generator bus voltages;
    get_Vgen();

    % update control vector (gbest):
    gbest=update_control_vector()';

    % recompute objective function:
    f_best=PSO_objective_evaluation(gbest);

    % keep track of the global best position and objective function values,
    % as well as the change in these quantities between iterations:
    Gbest=[Gbest;gbest];
    fbest=[fbest; f_best];
    delta_fbest=abs(fbest(t+1)-fbest(t))/(1+abs(fbest(t+1)));
    dGbest=[dGbest; norm(Gbest(t+1,:)-Gbest(t,:))];
    dfbest=[dfbest; delta_fbest];

    % check to see if convergence criteria are satisfied by the global best
    % position:
    objective_norm_change=(abs(fbest(t+1)-fbest(t))/(1+abs(fbest(t+1)))<1e-4);
    global_position_change_norm=(norm(Gbest(t+1,:)-Gbest(t,:))<1e-4);
    objective_change=(fbest(t+1)-fbest(1))>=0;

    % increment the iteration counter, before the loop runs again:
    t=t+1;

end
toc

x_opt=Gbest(end,:)
f_opt=[fbest(1);fbest(end)]
num_iterations=t
if (num_iterations<201 || fbest(end)<fbest(1))
    percentage_loss_reduction=100*(fbest(1)-fbest(end))/fbest(1)
else
    disp('Failed to achieve power loss reduction within set maximum number of
iterations')
end
[Va, Vm]=cart2pol(real(V), imag(V));
Vpolar=[Vm 180*Va/pi];

    plot(Gbest(:,1),Gbest(:,2),Gbest(:,1),Gbest(:,2),'r*')
    grid
    for i=1:length(Gbest(:,1))
        text(Gbest(i,1), Gbest(i,2), ['iter ' num2str(i)])
    end
    figure(2)
    subplot(311)
```

```
    plot(dGbest)
    hold on
    plot(dGbest,'r*')
    title('Change in global best position')
    grid
    hold off
    subplot(313)
    plot(fbest), grid
    title('Fitness value of global best position')
    subplot(312)
    plot(dfbest), grid
    title('Change in fitness value of global best position')

figure(3)
Vfinal=output.V(:,1);
plot(Vinit,'b-s'), hold on, plot(Vfinal,'r-s'),legend('V_{Initial}','V_{Final}'),
hold off;
grid
title('Voltage profile, final compared with initial');
xlabel('Bus number')
ylabel('V_{bus-pu}')

xlswrite('three_bus_system_voltage2.xlsx', [Vinit Vfinal]);

loss=fbest;
P1=PQs(1,:)';
Q1=PQs(2,:)';
Q3=PQs(3,:)';
Qsum=.0015+Qsum';
Vg1=Vgens(1,:)';
Vg3=Vgens(2,:)';

x1=0:length(loss)-1;
x2=0:length(P1)-1;

figure(4)
subplot(221)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('Slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid

subplot(222)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
```

```
grid

subplot(223)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q3);
title('Bus-3 generator reactive power and power loss')
legend('P_{loss}','Q_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-3 generator reactive power');
hLine2.LineStyle='--';
grid

subplot(224)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Q3);
title('Generator reactive powers')
legend('Q_{slack}','Q_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Generator reactive powers');
hLine2.LineStyle='--';
grid


figure(5)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('Slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid

figure(6)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
grid

figure(7)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q3);
title('Bus-3 generator reactive power and power loss')
legend('P_{loss}','Q_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-3 generator reactive power');
hLine2.LineStyle='--';
grid

figure(8)
```

```
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Q3);
title('Generator reactive powers')
legend('Q_{slack}','Q_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Bus-3 generator reactive power');
hLine2.LineStyle='--';
grid

figure(9)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Qsum);
title('Total generated reactive power and power loss')
legend('P_{loss}','Q_{gen}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Total generated reactive power');
hLine2.LineStyle='--';
grid

figure(10)
subplot(121)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Vg1);
title('Slack-bus reactive power and voltage magnitude')
legend('Q_{slack}','V_{g1}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Slack-bus voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(122)
[hAx, ~, hLine2]=plotyy(x2,Q3,x2,Vg3);
title('Bus-3 generator reactive power and voltage magnitude')
legend('Q_{g3}','V_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-3 generator reactive power');
ylabel(hAx(2),'Bus-3 voltage magnitude');
hLine2.LineStyle='--';
grid
```

C.2 MATLAB script that runs the PSO-VVO algorithm for the 6-bus system

```
% File name: pso_vvo_6bus_system.m

% load 6-bus system data:
six_bus_system_data;
global PQs Qsum Vgens

% initialize PSO parameters:
c1=2.05;
c2=2.05;
% c1=2;
% c2=2;
```

```
p=10;       % swarm size
d=5;        % problem dimension
N=200;       % maximum number of iterations
wmin=.4;    % minimum inertia weight
wmax=.9;    % maximum inertia weight

% generate initial position vector:
e10=.95+.15*rand(p,1);
e20=.95+.15*rand(p,1);
f20=real(sqrt(1-(.95+.15*rand(p,1)).^2));
e30=.95+.15*rand(p,1);
f30=real(sqrt(1-(.95+.15*rand(p,1)).^2));

x=[e10 e20 f20 e30 f30];

% generate initial velocity vector (set to zero):
v=zeros(p,d);

fprev=inf*ones(p,1);        % initialize fitness function values (to infinity)
pbest=x;                    % initialize each particle's past best position (equal
to initial position)
Gbest=[];                   % keep track of global best position
fbest=[];                   % keep track of fitness value of global best position

% evaluate fitness function for each particle to determine personal and global best
positions:
[fprev,pbest,gbest,best_idx]=PSO_compute_pbest_gbest(x,fprev,pbest);

% run (Newton-Raphson) load flow:
% compute initial voltage vector for the load flow algorithm:
x_loadflow=define_updated_x0(gbest);

% compute voltage reference vector for the load flow algorithm:
Vgref_loadflow=updated_Vgref(gbest);

% run the load flow algorithm:
[V, output]=NR_load_flow(@dF, @jacobian, x_loadflow, Vgref_loadflow);
E=real(V);
F=imag(V);

% Initialize slack-bus active, and generator reactive power outputs
% matrix to empty matrix:
PQs=[];

% Compute slack-bus active, and generator reactive power outputs
compute_PQ();

% Vector to hold generator bus voltages:
Vgens=[];

% Extract generator bus voltages;
get_Vgen();

% Initial load flow voltage results:
```

```
Vinit=output.V(:,1);

% update control vector (gbest):
gbest=update_control_vector()';

% recompute objective function:
f_best=PSO_objective_evaluation(gbest);

Gbest=[Gbest;gbest];
fbest=[fbest;f_best];

% keep tract of change in global best position and its corresponding fitness value:
dGbest=[];
dfbest=[];

% compute termination conditions based on change in global best position, and its
associated fitness value:
objective_norm_change=(abs(f_best)/(1+abs(f_best))<1e-5);
global_position_change_norm=(norm(gbest)<1e-5);
objective_change=abs(f_best)>=0;

% initialize iteration counter:
t=1;

tic
% loop until termination conditions are satisfied:
while ((~objective_norm_change || objective_change) && t<=N)
    r1=rand(p,1);
    r2=rand(p,1);

    % compute the particle velocity:
    %[V,X]=PSO_X_update2(V,X,pbest,gbest,c1,c2,r1,r2,wmin,wmax,t,N);
    [v,x]=PSO_X_update_cc1(v,x,pbest,gbest,c1,c2);

    % compute the objective function value for each particle to
    % update the personal and global best positions:
    [fprev,pbest,gbest,best_idx]=PSO_compute_pbest_gbest(x,fprev,pbest);

    % run (Newton-Raphson) load flow:
    % compute initial voltage vector for the load flow algorithm:
    x_loadflow=define_updated_x0(gbest);

    % compute voltage reference vector for the load flow algorithm:
    Vgref_loadflow=updated_Vgref(gbest);

    % run the load flow algorithm:
    [V, output]=NR_load_flow(@dF, @jacobian, x_loadflow, Vgref_loadflow);
    E=real(V);
    F=imag(V);

    % Compute slack-bus active, and generator reactive power outputs
    compute_PQ();

    % Extract generator bus voltages;
```

```
    get_Vgen();

    % update control vector (gbest):
    gbest=update_control_vector()';

    % recompute objective function:
    f_best=PSO_objective_evaluation(gbest);

    % keep track of the global best position and objective function values,
    % as well as the change in these quantities between iterations:
    Gbest=[Gbest;gbest];
    fbest=[fbest; f_best];
    delta_fbest=abs(fbest(t+1)-fbest(t))/(1+abs(fbest(t+1)));
    dGbest=[dGbest; norm(Gbest(t+1,:)-Gbest(t,:))];
    dfbest=[dfbest; delta_fbest];

    % check to see if convergence criteria are satisfied by the global best
    % position:
    objective_norm_change=(abs(fbest(t+1)-fbest(t))/(1+abs(fbest(t+1)))<1e-4);
    global_position_change_norm=(norm(Gbest(t+1,:)-Gbest(t,:))<1e-4);
    objective_change=(fbest(t+1)-fbest(1))>=0;

    % increment the iteration counter, before the loop runs again:
    t=t+1;

end
x_opt=Gbest(end,:)
f_opt=[fbest(1);fbest(end)]
num_iterations=t
if (num_iterations<201 || fbest(end)<fbest(1))
    percentage_loss=100*(fbest(1)-fbest(end))/fbest(1)
else
    disp('Failed to achieve power loss reduction within set maximum number of
iterations')
end

[Va, Vm]=cart2pol(real(V), imag(V));
Vpolar=[Vm 180*Va/pi];

toc
    plot(Gbest(:,1),Gbest(:,2),Gbest(:,1),Gbest(:,2),'r*')
    grid
    for i=1:length(Gbest(:,1))
        text(Gbest(i,1), Gbest(i,2), ['iter ' num2str(i)])
    end
    figure(2)
    subplot(311)
    plot(dGbest)
    hold on
    plot(dGbest,'r*')
    title('Change in global best position')
    grid
    hold off
    subplot(313)
```

```
    plot(fbest), grid
    title('Fitness value of global best position')
    subplot(312)
    plot(dfbest), grid
    title('Change in fitness value of global best position')

figure(3)
Vfinal=output.V(:,1);
plot(Vinit,'b-s'), hold on, plot(Vfinal,'r-s'),legend('V_{Initial}','V_{Final}'),
hold off;
grid
title('Voltage profile, final compared with initial');
xlabel('Bus number')
ylabel('V_{bus-pu}')

xlswrite('six_bus_system_voltage2.xlsx', [Vinit Vfinal]);

loss=fbest;
P1=PQs(1,:)';
Q1=PQs(2,:)';
Q2=PQs(3,:)';
Q3=PQs(4,:)';
Qsum=Qsum';
Vg1=Vgens(1,:)';
Vg2=Vgens(2,:)';
Vg3=Vgens(3,:)';

x1=0:length(loss)-1;
x2=0:length(P1)-1;

figure(4)
subplot(221)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('Slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid

subplot(222)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
grid

subplot(223)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q2);
title('Bus-2 generator reactive power and power loss')
```

```
legend('P_{loss}','Q_{g2}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-2 generator reactive power');
hLine2.LineStyle='--';
grid

subplot(224)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Q3);
title('Generator reactive powers')
legend('Q_{slack}','Q_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Generator reactive powers');
hLine2.LineStyle='--';
grid

figure(5)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('Slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid

figure(6)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
grid

figure(7)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q2);
title('Bus-2 generator reactive power and power loss')
legend('P_{loss}','Q_{g2}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-2 generator reactive power');
hLine2.LineStyle='--';
grid

figure(8)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q3);
title('Bus-3 generator reactive power and power loss')
legend('P_{loss}','Q_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-3 generator reactive power');
```

```
hLine2.LineStyle='--';
grid

figure(9)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Qsum);
title('Total generated reactive power and power loss')
legend('P_{loss}','Q_{gen}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Total generated reactive power');
hLine2.LineStyle='--';
grid

figure(10)
plot(x2,Q1,x2,Q2,'r--',x2,Q3,'m-.')
title('Generator reactive powers')
legend('Q_{slack}','Q_{g2}','Q_{g3}')
grid

figure(11)
subplot(221)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Vg1);
title('Slack-bus reactive power and voltage magnitude')
legend('Q_{slack}','V_{g1}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Slack-bus voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(222)
[hAx, ~, hLine2]=plotyy(x2,Q2,x2,Vg2);
title('Bus-2 generator reactive power and voltage magnitude')
legend('Q_{g2}','V_{g2}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-2 generator reactive power');
ylabel(hAx(2),'Bus-2 voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(223)
[hAx, ~, hLine2]=plotyy(x2,Q3,x2,Vg3);
title('Bus-3 generator reactive power and voltage magnitude')
legend('Q_{g3}','V_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-3 generator reactive power');
ylabel(hAx(2),'Bus-3 voltage magnitude');
hLine2.LineStyle='--';
grid
```

C.3 MATLAB script that runs the PSO-VVO algorithm for the 14-bus system

```
% File name: pso_vvo_14bus_system.m
```

```
% load 14-bus system data:
fourteen_bus_system_data;
global PQs Qsum Vgens

% initialize PSO parameters:
c1=2.05;
c2=2.05;
% c1=2;
% c2=2;
p=20;       % swarm size
d=9;        % problem dimension
N=200;       % maximum number of iterations
wmin=.4;    % minimum inertia weight
wmax=.9;    % maximum inertia weight

% generate initial position vector:
ng=length(find(bus_type==2));   % number of PV buses
x=zeros(p,d);                   % initialize matrix to store initial generator
voltage vectors
x(:,1)=.95+.15*rand(p,1);       % first element is slack-bus real component of
voltage
k=2;
for i=2:ng+1                      % loop over PV buses
    x(:,k)=.95+.15*rand(p,1);
    x(:,k+1)=(-1)^randi(ng)*real(sqrt(1-x(:,k).^2));
    %x(:,k+1)=(-1)^i*real(sqrt(1-(.95+.15*rand(p,1)).^2));
    k=k+2;
end

% generate initial velocity vector (set to zero):
v=zeros(p,d);
%v=.2*rand(p,d);
X0=x;
fprev=inf*ones(p,1);        % initialize fitness function values (to infinity)
pbest=x;                    % initialize each particle's past best position (equal
to initial position)
Gbest=[];                   % keep track of global best position
fbest=[];                   % keep track of fitness value of global best position

% evaluate fitness function for each particle to determine personal and
% global best positions:
[fprev,pbest,gbest,best_idx]=PSO_compute_pbest_gbest_14bus2(x,fprev,pbest);

% run (Newton-Raphson) load flow:
% compute initial voltage vector for the load flow algorithm:
x_loadflow=define_updated_x0(gbest);

% compute voltage reference vector for the load flow algorithm:
Vgref_loadflow=updated_Vgref(gbest);

% run the load flow algorithm:
[V, output]=NR_load_flow(@dF, @jacobian, x_loadflow, Vgref_loadflow);
E=real(V);
```

```
F=imag(V);


% Initialize slack-bus active, and generator reactive power outputs
% matrix to empty matrix:
PQs=[];

% Compute slack-bus active, and generator reactive power outputs
compute_PQ();

% Vector to hold generator bus voltages:
Vgens=[];

% Extract generator bus voltages;
get_Vgen();

% Initial load flow voltage results:
Vinit=output.V(:,1);

% update control vector (gbest):
gbest_before_update_control_vector=gbest;
gbest=update_control_vector()';
gbest_after_update_control_vector=gbest;
% recompute objective function:
f_best=PSO_objective_evaluation_14bus2(gbest);

Gbest=[Gbest;gbest];
fbest=[fbest;f_best];

% keep tract of change in global best position and its corresponding
% fitness value:
dGbest=[];
dfbest=[];

% compute termination conditions based on change in global best position, and its
associated fitness value:
objective_norm_change=(abs(f_best)/(1+abs(f_best))<1e-5);
global_position_change_norm=(norm(gbest)<1e-5);
objective_change=abs(f_best)>=0;

% initialize iteration counter:
t=1;

tic
% loop until termination conditions are satisfied:
while ((~objective_norm_change || objective_change) && t<=N)

    % compute the particle velocity:
    %[V,X]=PSO_X_update2(V,X,pbest,gbest,c1,c2,r1,r2,wmin,wmax,t,N);
    [v,x]=PSO_X_update_cc_14bus(v,x,pbest,gbest,c1,c2);

    % compute the objective function value for each particle to
    % update the personal and global best positions:
    [fprev,pbest,gbest,best_idx]=PSO_compute_pbest_gbest_14bus2(x,fprev,pbest);
```

```
    % run (Newton-Raphson) load flow:
    % compute initial voltage vector for the load flow algorithm:
    x_loadflow=define_updated_x0(gbest);

    % compute voltage reference vector for the load flow algorithm:
    Vgref_loadflow=updated_Vgref(gbest);

    % run the load flow algorithm:
    [V, output]=NR_load_flow(@dF, @jacobian, x_loadflow, Vgref_loadflow);
    E=real(V);
    F=imag(V);

    % Compute slack-bus active, and generator reactive power outputs
    compute_PQ();

    % Extract generator bus voltages;
    get_Vgen();

    % update control vector (gbest):
    gbest=update_control_vector()';

    % recompute objective function:
    f_best=PSO_objective_evaluation_14bus2(gbest);

    % keep track of the global best position and objective function values,
    % as well as the change in these quantities between iterations:
    Gbest=[Gbest;gbest];
    fbest=[fbest; f_best];
    delta_fbest=abs(fbest(t+1)-fbest(t))/(1+abs(fbest(t+1)));
    dGbest=[dGbest; norm(Gbest(t+1,:)-Gbest(t,:))];
    dfbest=[dfbest; delta_fbest];

    % check to see if convergence criteria are satisfied by the global best
    % position:
    objective_norm_change=(abs(fbest(t+1)-fbest(t))/(1+abs(fbest(t+1)))<1e-4);
    global_position_change_norm=(norm(Gbest(t+1,:)-Gbest(t,:))<1e-4);
    objective_change=(fbest(t+1)-fbest(1))>=0;

    % increment the iteration counter, before the loop runs again:
    t=t+1;

end
x_opt=Gbest(end,:)
f_opt=[fbest(1);fbest(end)]
num_iterations=t
if (num_iterations<201 || fbest(end)<fbest(1))
    percentage_loss=100*(fbest(1)-fbest(end))/fbest(1)
else
    disp('Failed to achieve power loss reduction within set maximum number of
iterations')
end
[Va, Vm]=cart2pol(real(V), imag(V));
Vpolar=[Vm 180*Va/pi];
```

```
toc
    plot(Gbest(:,1),Gbest(:,2),Gbest(:,1),Gbest(:,2),'r*')
    grid
    for i=1:length(Gbest(:,1))
        text(Gbest(i,1), Gbest(i,2), ['iter ' num2str(i)])
    end
    figure(2)
    subplot(311)
    plot(dGbest)
    hold on
    plot(dGbest,'r*')
    title('Change in global best position')
    grid
    hold off
    subplot(313)
    plot(fbest), grid
    title('Fitness value of global best position')
    subplot(312)
    plot(dfbest), grid
    title('Change in fitness value of global best position')

figure(3)
Vfinal=output.V(:,1);
plot(Vinit,'b-s'), hold on, plot(Vfinal,'r-s'),legend('V_{Initial}','V_{Final}'),
hold off;
grid
title('Voltage profile, final compared with initial');
xlabel('Bus number')
ylabel('V_{bus-pu}')

% xlswrite('fourteen_bus_system_voltage2.xlsx', [Vinit Vfinal]);

loss=fbest;
P1=PQs(1,:)';
Q1=PQs(2,:)';
Q2=PQs(3,:)';
Q3=PQs(4,:)';
Q6=PQs(5,:)';
Q8=PQs(6,:)';
Qsum=Qsum';
Vg1=Vgens(1,:)';
Vg2=Vgens(2,:)';
Vg3=Vgens(3,:)';
Vg6=Vgens(4,:)';
Vg8=Vgens(5,:)';

x1=0:length(loss)-1;
x2=0:length(P1)-1;

figure(4)
subplot(221)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('Slack-bus active power and power loss')
```

```
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid

subplot(222)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
grid

subplot(223)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q3);
title('Bus-3 generator reactive power and power loss')
legend('P_{loss}','Q_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-3 generator reactive power');
hLine2.LineStyle='--';
grid

subplot(224)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Q2);
title('Generator reactive powers')
legend('Q_{slack}','Q_{g2}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Generator reactive powers');
hLine2.LineStyle='--';
grid


figure(5)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('Slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid

figure(6)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
```

```
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
grid

figure(7)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q2);
title('Bus-2 generator reactive power and power loss')
legend('P_{loss}','Q_{g2}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-2 generator reactive power');
hLine2.LineStyle='--';
grid

figure(8)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q3);
title('Bus-3 generator reactive power and power loss')
legend('P_{loss}','Q_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-3 generator reactive power');
hLine2.LineStyle='--';
grid

figure(9)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q6);
title('Bus-6 generator reactive power and power loss')
legend('P_{loss}','Q_{g6}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-6 generator reactive power');
hLine2.LineStyle='--';
grid

figure(10)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q8);
title('Bus-6 generator reactive power and power loss')
legend('P_{loss}','Q_{g8}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-8 generator reactive power');
hLine2.LineStyle='--';
grid

figure(11)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Qsum);
title('Total generated reactive power and power loss')
legend('P_{loss}','Q_{gen}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Total generated reactive power');
hLine2.LineStyle='--';
grid
```

```
figure(12)
plot(x2,Q1,x2,Q2,'r--',x2,Q3,'m-.',x2,Q6,'c-+',x2,Q8,'k-*')
title('Generator reactive powers')
legend('Q_{slack}','Q_{g2}','Q_{g3}','Q_{g6}','Q_{g8}')
grid

figure(13)
subplot(321)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Vg1);
title('Slack-bus reactive power and voltage magnitude')
legend('Q_{slack}','V_{g1}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Slack-bus voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(322)
[hAx, ~, hLine2]=plotyy(x2,Q2,x2,Vg2);
title('Bus-2 generator reactive power and voltage magnitude')
legend('Q_{g2}','V_{g2}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-2 generator reactive power');
ylabel(hAx(2),'Bus-2 voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(323)
[hAx, ~, hLine2]=plotyy(x2,Q3,x2,Vg3);
title('Bus-3 generator reactive power and voltage magnitude')
legend('Q_{g3}','V_{g3}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-3 generator reactive power');
ylabel(hAx(2),'Bus-3 voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(324)
[hAx, ~, hLine2]=plotyy(x2,Q6,x2,Vg6);
title('Bus-6 generator reactive power and voltage magnitude')
legend('Q_{g6}','V_{g6}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-6 generator reactive power');
ylabel(hAx(2),'Bus-6 voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(325)
[hAx, ~, hLine2]=plotyy(x2,Q8,x2,Vg8);
title('Bus-8 generator reactive power and voltage magnitude')
legend('Q_{g8}','V_{g8}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-8 generator reactive power');
ylabel(hAx(2),'Bus-8 voltage magnitude');
```

```
hLine2.LineStyle='--';
grid
```

C.4 MATLAB script that runs the PSO-VVO algorithm for the 30-bus system

```
% File name: pso_vvo_30bus_system.m

% load 30-bus system data:
thirty_bus_system_data;
global PQs Qsum Vgens

% initialize PSO parameters:
c1=2.05;
c2=2.05;
% c1=2;
% c2=2;
p=20;        % swarm size
d=11;         % problem dimension
N=200;        % maximum number of iterations
wmin=.4;     % minimum inertia weight
wmax=.9;     % maximum inertia weight

% generate initial position vector:
ng=length(find(bus_type==2));    % number of PV buses
x=zeros(p,d);                    % initialize matrix to store initial generator
voltage vectors
x(:,1)=.95+.15*rand(p,1);        % first element is slack-bus real component of
voltage
k=2;
for i=2:ng+1                       % loop over PV buses
    x(:,k)=.95+.15*rand(p,1);
    %x(:,k+1)=(-1)^i*real(sqrt(1-x(:,k).^2));
    x(:,k+1)=(-1)^i*real(sqrt(1-(.95+.15*rand(p,1)).^2));
    k=k+2;
end

% generate initial velocity vector (set to zero):
v=zeros(p,d);
%v=.2*rand(p,d);
X0=x;
fprev=inf*ones(p,1);         % initialize fitness function values (to infinity)
pbest=x;                     % initialize each particle's past best position (equal
to initial position)
Gbest=[];                    % keep track of global best position
fbest=[];                    % keep track of fitness value of global best position

% evaluate fitness function for each particle to determine personal and global best
positions:
[fprev,pbest,gbest,best_idx]=PSO_compute_pbest_gbest_30bus1(x,fprev,pbest);

% run (Newton-Raphson) load flow:
% compute initial voltage vector for the load flow algorithm:
x_loadflow=define_updated_x0(gbest);
```

```
% compute voltage reference vector for the load flow algorithm:
Vgref_loadflow=updated_Vgref(gbest);

% run the load flow algorithm:
[V, output]=NR_load_flow(@dF, @jacobian, x_loadflow, Vgref_loadflow);
E=real(V);
F=imag(V);


% Initialize slack-bus active, and generator reactive power outputs
% matrix to empty matrix:
PQs=[];

% Initial load flow voltage results:
Vinit=output.V(:,1);

% update control vector (gbest):
gbest_before_update_control_vector=gbest;
gbest=update_control_vector()';
gbest_after_update_control_vector=gbest;

% recompute objective function:
f_best=PSO_objective_evaluation_30bus1(gbest);

Gbest=[Gbest;gbest];
fbest=[fbest;f_best];


% Compute slack-bus active, and generator reactive power outputs
compute_PQ();

% Vector to hold generator bus voltages:
Vgens=[];

% Extract generator bus voltages;
get_Vgen();


% keep tract of change in global best position and its corresponding fitness value:
dGbest=[];
dfbest=[];

% keep track of velocity update:
vupdate=[];

% compute termination conditions based on change in global best position, and its
associated fitness value:
objective_norm_change=(abs(f_best)/(1+abs(f_best))<1e-5);
global_position_change_norm=(norm(gbest)<1e-5);
objective_change=abs(f_best)>=0;

% initialize iteration counter:
t=1;
```

```
tic
% loop until termination conditions are satisfied:
while ((~objective_norm_change || objective_change) && t<=N)

    % compute the particle velocity:
    %[v,x]=PSO_X_update3(v,x,pbest,gbest,c1,c2,wmin,wmax,t,N);
    [v,x]=PSO_X_update_cc_30bus(v,x,pbest,gbest,c1,c2);

    % keep track of velocity update:
    vupdate=[vupdate; v];

    % compute the objective function value for each particle to
    % update the personal and global best positions:
    [fprev,pbest,gbest,best_idx]=PSO_compute_pbest_gbest_30bus1(x,fprev,pbest);

    % run (Newton-Raphson) load flow:
    % compute initial voltage vector for the load flow algorithm:
    x_loadflow=define_updated_x0(gbest);

    % compute voltage reference vector for the load flow algorithm:
    Vgref_loadflow=updated_Vgref(gbest);

    % run the load flow algorithm:
    [V, output]=NR_load_flow(@dF, @jacobian, x_loadflow, Vgref_loadflow);
    E=real(V);
    F=imag(V);



    % update control vector (gbest):
    gbest=update_control_vector()';

    % recompute objective function:
    f_best=PSO_objective_evaluation_30bus1(gbest);

    % keep track of the global best position and objective function values,
    % as well as the change in these quantities between iterations:
    Gbest=[Gbest;gbest];
    fbest=[fbest; f_best];
    delta_fbest=abs(fbest(t+1)-fbest(t))/(1+abs(fbest(t+1)));
    dGbest=[dGbest; norm(Gbest(t+1,:)-Gbest(t,:))];
    dfbest=[dfbest; delta_fbest];

     % Compute slack-bus active, and generator reactive power outputs
    compute_PQ();

    % Extract generator bus voltages;
    get_Vgen();


    % check to see if convergence criteria are satisfied by the global best
    % position:
    objective_norm_change=(abs(fbest(t+1)-fbest(t))/(1+abs(fbest(t+1)))<1e-4);
```

```
    global_position_change_norm=(norm(Gbest(t+1,:)-Gbest(t,:))<1e-4);
    objective_change=(fbest(t+1)-fbest(1))>=0;

    % increment the iteration counter, before the loop runs again:
    t=t+1;

end
x_opt=Gbest(end,:)
f_opt=[fbest(1);fbest(end)]
num_iterations=t
if (num_iterations<N+1 || fbest(end)<fbest(1))
    percentage_loss=100*(fbest(1)-fbest(end))/fbest(1)
else
    disp('Failed to achieve power loss reduction within set maximum number of
iterations')
end

[Va, Vm]=cart2pol(real(V), imag(V));
Vpolar=[Vm 180*Va/pi];

toc
    plot(Gbest(:,1),Gbest(:,2),Gbest(:,1),Gbest(:,2),'r*')
    grid
    for i=1:length(Gbest(:,1))
        text(Gbest(i,1), Gbest(i,2), ['iter ' num2str(i)])
    end
    figure(2)
    subplot(311)
    plot(dGbest)
    hold on
    plot(dGbest,'r*')
    title('Change in global best position')
    grid
    hold off
    subplot(313)
    plot(fbest), grid
    title('Fitness value of global best position')
    subplot(312)
    plot(dfbest), grid
    title('Change in fitness value of global best position')

figure(3)
Vfinal=output.V(:,1);
plot(Vinit,'b-s'), hold on, plot(Vfinal,'r-s'),legend('V_{Initial}','V_{Final}'),
hold off;
grid
title('Voltage profile, final compared with initial');
xlabel('Bus number')
ylabel('V_{bus-pu}')

% xlswrite('thirty_bus_system_voltage2.xlsx', [Vinit Vfinal]);

loss=fbest;
% Adding 2.2e-3 to P1 makes slack-bus active power
```

```
% and power loss coincide; this (value of 2.2e-3)
% seems to only represent a discrepancy due to
% differences in scale of the two quantities
P1=2.2e-3+PQs(1,:)';
Q1=PQs(2,:)';
Q2=PQs(3,:)';
Q5=PQs(4,:)';
Q8=PQs(5,:)';
Q11=PQs(6,:)';
Q13=PQs(7,:)';
Qsum=Qsum';
Vg1=Vgens(1,:)';
Vg2=Vgens(2,:)';
Vg5=Vgens(3,:)';
Vg8=Vgens(4,:)';
Vg11=Vgens(5,:)';
Vg13=Vgens(6,:)';

x1=0:length(loss)-1;
x2=0:length(P1)-1;

figure(4)
subplot(221)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('Slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid

subplot(222)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
grid

subplot(223)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q5);
title('Bus-5 generator reactive power and power loss')
legend('P_{loss}','Q_{g5}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-5 generator reactive power');
hLine2.LineStyle='--';
grid

subplot(224)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Q5);
```

```
title('Generator reactive powers')
legend('Q_{slack}','Q_{g5}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Generator reactive powers');
hLine2.LineStyle='--';
grid

figure(5)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('Slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid

figure(6)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
grid

figure(7)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q2);
title('Bus-2 generator reactive power and power loss')
legend('P_{loss}','Q_{g2}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-2 generator reactive power');
hLine2.LineStyle='--';
grid

figure(8)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q5);
title('Bus-5 generator reactive power and power loss')
legend('P_{loss}','Q_{g5}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-5 generator reactive power');
hLine2.LineStyle='--';
grid

figure(9)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q8);
title('Bus-8 generator reactive power and power loss')
legend('P_{loss}','Q_{g8}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
```

```
ylabel(hAx(2),'Bus-8 generator reactive power');
hLine2.LineStyle='--';
grid

figure(10)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q11);
title('Bus-11 generator reactive power and power loss')
legend('P_{loss}','Q_{g11}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-11 generator reactive power');
hLine2.LineStyle='--';
grid

figure(11)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q13);
title('Bus-11 generator reactive power and power loss')
legend('P_{loss}','Q_{g13}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-13 generator reactive power');
hLine2.LineStyle='--';
grid

figure(12)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Qsum);
title('Total generated reactive power and power loss')
legend('P_{loss}','Q_{gen}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Total generated reactive power');
hLine2.LineStyle='--';
grid

figure(13)
plot(x2,Q1,x2,Q2,'r--',x2,Q5,'m-.',x2,Q8,'c-+',x2,Q11,'k-*',x2,Q13,'g-o')
title('Generator reactive powers')
legend('Q_{slack}','Q_{g2}','Q_{g5}','Q_{g8}','Q_{g11}','Q_{g13}')
grid

figure(14)
subplot(211)
[hAx, ~, hLine2]=plotyy(x2,Q1,x2,Vg1);
title('Slack-bus reactive power and voltage magnitude')
legend('Q_{slack}','V_{g1}')
xlabel('Iteration number')
ylabel(hAx(1),'Slack-bus reactive power');
ylabel(hAx(2),'Slack-bus voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(212)
[hAx, ~, hLine2]=plotyy(x2,Q2,x2,Vg2);
title('Bus-2 generator reactive power and voltage magnitude')
```

```
legend('Q_{g2}','V_{g2}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-2 generator reactive power');
ylabel(hAx(2),'Bus-2 voltage magnitude');
hLine2.LineStyle='--';
grid

figure(15)
subplot(211)
[hAx, ~, hLine2]=plotyy(x2,Q5,x2,Vg5);
title('Bus-5 generator reactive power and voltage magnitude')
legend('Q_{g5}','V_{g5}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-5 generator reactive power');
ylabel(hAx(2),'Bus-5 voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(212)
[hAx, ~, hLine2]=plotyy(x2,Q8,x2,Vg8);
title('Bus-8 generator reactive power and voltage magnitude')
legend('Q_{g8}','V_{g8}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-8 generator reactive power');
ylabel(hAx(2),'Bus-8 voltage magnitude');
hLine2.LineStyle='--';
grid

figure(16)
subplot(211)
[hAx, ~, hLine2]=plotyy(x2,Q11,x2,Vg11);
title('Bus-11 generator reactive power and voltage magnitude')
legend('Q_{g11}','V_{g11}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-11 generator reactive power');
ylabel(hAx(2),'Bus-11 voltage magnitude');
hLine2.LineStyle='--';
grid

subplot(212)
[hAx, ~, hLine2]=plotyy(x2,Q13,x2,Vg13);
title('Bus-13 generator reactive power and voltage magnitude')
legend('Q_{g13}','V_{g13}')
xlabel('Iteration number')
ylabel(hAx(1),'Bus-13 generator reactive power');
ylabel(hAx(2),'Bus-13 voltage magnitude');
hLine2.LineStyle='--';
grid
```

C.5 MATLAB script that runs the PSO-VVO algorithm for the 118-bus system

```
% File name: pso_vvo_118bus_system.m
```

```
% load 118-bus system data:
one_hundred_eighteen_bus_system_data;
global PQs Qsum

% initialize PSO parameters:
c1=2.05;
c2=2.05;
% c1=2;
% c2=2;
p=30;       % swarm size
d=107;        % problem dimension
N=200;       % maximum number of iterations
wmin=.4;    % minimum inertia weight
wmax=.9;    % maximum inertia weight

% generate initial position vector:
ng=length(find(bus_type==2));   % number of PV buses
x=zeros(p,d);                   % initialize matrix to store initial generator
voltage vectors
x(:,1)=.95+.15*rand(p,1);       % first element is slack-bus real component of
voltage
k=2;
for i=2:ng+1                      % loop over PV buses
    x(:,k)=.95+.15*rand(p,1);
    %x(:,k+1)=(-1)^i*real(sqrt(1-x(:,k).^2));
    x(:,k+1)=(-1)^i*real(sqrt(1-(.95+.15*rand(p,1)).^2));
    %x(:,k+1)=real(sqrt(1-(.95+.15*rand(p,1)).^2));
    k=k+2;
end

% generate initial velocity vector (set to zero):
v=zeros(p,d);
%v=.2*rand(p,d);
X0=x;
fprev=inf*ones(p,1);        % initialize fitness function values (to infinity)
pbest=x;                    % initialize each particle's past best position (equal
to initial position)
Gbest=[];                   % keep track of global best position
fbest=[];                   % keep track of fitness value of global best position

% evaluate fitness function for each particle to determine personal and global best
positions:
[fprev,pbest,gbest,best_idx]=PSO_compute_pbest_gbest_118bus(x,fprev,pbest);

% run (Newton-Raphson) load flow:
% compute initial voltage vector for the load flow algorithm:
x_loadflow=define_updated_x0(gbest);

% compute voltage reference vector for the load flow algorithm:
Vgref_loadflow=updated_Vgref(gbest);

% run the load flow algorithm:
[V, output]=NR_load_flow(@dF, @jacobian, x_loadflow, Vgref_loadflow);
E=real(V);
```

```
F=imag(V);

% Initialize slack-bus active, and generator reactive power matrix to
% empty array:

PQs=[];

% Compute slack-bus active, and generator reactive power outputs
compute_PQ();

% Initial load flow voltage results:
Vinit=output.V(:,1);

% update control vector (gbest):
gbest_before_update_control_vector=gbest;
gbest=update_control_vector()';
gbest_after_update_control_vector=gbest;

% recompute objective function:
f_best=PSO_objective_evaluation_118bus(gbest);

Gbest=[Gbest;gbest];
fbest=[fbest;f_best];

% keep tract of change in global best position and its corresponding fitness value:
dGbest=[];
dfbest=[];

% compute termination conditions based on change in global best position, and its
associated fitness value:
objective_norm_change=(abs(f_best)/(1+abs(f_best))<1e-3);
global_position_change_norm=(norm(gbest)<1e-3);
objective_change=abs(f_best)>=0;

% initialize iteration counter:
t=1;

tic
% loop until termination conditions are satisfied:
while ((~objective_norm_change || objective_change) && t<=N)

    % compute the particle velocity:
    %[V,X]=PSO_X_update2(V,X,pbest,gbest,c1,c2,r1,r2,wmin,wmax,t,N);
    [v,x]=PSO_X_update_cc_118bus(v,x,pbest,gbest,c1,c2);

    % compute the objective function value for each particle to
    % update the personal and global best positions:
    [fprev,pbest,gbest,best_idx]=PSO_compute_pbest_gbest_118bus(x,fprev,pbest);

    % run (Newton-Raphson) load flow:
    % compute initial voltage vector for the load flow algorithm:
    x_loadflow=define_updated_x0(gbest);

    % compute voltage reference vector for the load flow algorithm:
```

```
    Vgref_loadflow=updated_Vgref(gbest);

    % run the load flow algorithm:
    [V, output]=NR_load_flow(@dF, @jacobian, x_loadflow, Vgref_loadflow);
    E=real(V);
    F=imag(V);

    % Compute slack-bus active, and generator reactive power outputs
    compute_PQ();

    % update control vector (gbest):
    gbest=update_control_vector()';

    % recompute objective function:
    f_best=PSO_objective_evaluation_118bus(gbest);

    % keep track of the global best position and objective function values,
    % as well as the change in these quantities between iterations:
    Gbest=[Gbest;gbest];
    fbest=[fbest; f_best];
    delta_fbest=abs(fbest(t+1)-fbest(t))/(1+abs(fbest(t+1)));
    dGbest=[dGbest; norm(Gbest(t+1,:)-Gbest(t,:))];
    dfbest=[dfbest; delta_fbest];

    % check to see if convergence criteria are satisfied by the global best
    % position:
    objective_norm_change=(abs(fbest(t+1)-fbest(t))/(1+abs(fbest(t+1)))<0.01);
    global_position_change_norm=(norm(Gbest(t+1,:)-Gbest(t,:))<1e-4);
    objective_change=(fbest(t+1)-fbest(1))>=0;

    % increment the iteration counter, before the loop runs again:
    t=t+1;

end
x_opt=Gbest(end,:);
f_opt=[fbest(1);fbest(end)]
num_iterations=t
if (num_iterations<201 || fbest(end)<fbest(1))
    percentage_loss=100*(fbest(1)-fbest(end))/fbest(1)
else
    disp('Failed to achieve power loss reduction within set maximum number of
iterations')
end

[Va, Vm]=cart2pol(real(V), imag(V));
Vpolar=[Vm 180*Va/pi];

toc
    plot(Gbest(:,1),Gbest(:,2),Gbest(:,1),Gbest(:,2),'r*')
    grid
    for i=1:length(Gbest(:,1))
        text(Gbest(i,1), Gbest(i,2), ['iter ' num2str(i)])
    end
    figure(2)
```

```
    subplot(311)
    plot(dGbest)
    hold on
    plot(dGbest,'r*')
    title('Change in global best position')
    grid
    hold off
    subplot(313)
    plot(fbest), grid
    title('Fitness value of global best position')
    subplot(312)
    plot(dfbest), grid
    title('Change in fitness value of global best position')

figure(3)
Vfinal=output.V(:,1);
plot(Vinit,'b-s'), hold on, plot(Vfinal,'r-s'),legend('V_{Initial}','V_{Final}'),
hold off;
grid
title('Voltage profile, final compared with initial');
xlabel('Bus number')
ylabel('V_{bus-pu}')

% xlswrite('one_hundred_eighteen_bus_system_voltage2.xlsx', [Vinit Vfinal]);

loss=fbest;
% Adding 2e-2 to P1 makes slack-bus active power
% and power loss coincide; this (value of 2e-2)
% seems to only represent a discrepancy due to
% differences in scale of the two quantities
P1=.02+PQs(1,:)';
Q1=PQs(2,:)';
Q4=PQs(3,:)';
% 4.8e-1 added to Qsum to shift it up, improves comparison with
% real power loss reduction
Qsum=.48+Qsum';

x1=0:length(loss)-1;
x2=0:length(P1)-1;

figure(4)
subplot(221)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('Slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='--';
grid

subplot(222)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
```

```
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
grid

subplot(223)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q4);
title('Bus-4 generator reactive power and power loss')
legend('P_{loss}','Q_{g4}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Bus-4 generator reactive power');
hLine2.LineStyle='--';
grid

subplot(224)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Qsum);
title('Total generated reactive power and power loss')
legend('P_{loss}','Q_{gen}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Total generated reactive power');
hLine2.LineStyle='--';
grid

figure(5)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,P1);
title('Slack-bus active power and power loss')
legend('P_{loss}','P_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus real power');
hLine2.LineStyle='-.';
grid

figure(6)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Q1);
title('Slack-bus reactive power and power loss')
legend('P_{loss}','Q_{slack}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Slack-bus reactive power');
hLine2.LineStyle='--';
grid

figure(7)
[hAx, ~, hLine2]=plotyy(x1,loss,x2,Qsum);
title('Total generated reactive power and power loss')
legend('P_{loss}','Q_{gen}')
xlabel('Iteration number')
ylabel(hAx(1),'Real power loss');
ylabel(hAx(2),'Total generated reactive power');
```

```
hLine2.LineStyle='--';
grid
```

C.6 Function that computes the personal and global best positions for the PSO algorithm; applies to all case studies

```
function [fprev,pbest,gbest,best_idx]=PSO_compute_pbest_gbest(X,fprev,pbest)
global E F bus_type nbus

Vmsq=.95^2;
VMsq=1.1^2;
tol=1e-4;

% obtain swarm size and problem dimension:
[p,~]=size(X);

% initialize a vector to store the objective function/fitness value of each
particle:
f=zeros(p,1);

% evaluate objective and constraint functions;
% for any bound constraint exceeding the limit,
% set it to the limit value
gen_buses=[find(bus_type==1); find(bus_type==2)];
for i=1:p
    if (-(X(i,1)^2+F(1)^2)+Vmsq>tol)
        X(i,1)=sqrt(Vmsq);
        E(1)=X(i,1);
        F(1)=0;
    end

    if (X(i,1)^2+F(1)^2-VMsq>tol)
        X(i,1)=sqrt(VMsq);
        E(1)=X(i,1);
        F(1)=0;
    end

    k=2;
    for j=2:length(gen_buses)
        if (-(X(i,k)^2+X(i,k+1)^2)+Vmsq>tol)
            X(i,k)=sqrt(Vmsq);
            X(i,k+1)=0;
            E(gen_buses(j))=X(i,k);
            F(gen_buses(j))=X(i,k+1);
            k=k+2;
        else
            if (X(i,k)^2+X(i,k+1)^2-VMsq>tol)
                X(i,k)=sqrt(VMsq);
                X(i,k+1)=0;
                E(2)=X(i,k);
                F(2)=X(i,k+1);
```

```
                k=k+2;
            end
        end
    end
    k=0;

    ngen_buses=1:nbus;
    ngen_buses(gen_buses)=[];

    for jj=1:length(ngen_buses)
        if (-(E(ngen_buses(jj))^2+F(ngen_buses(jj))^2)+Vmsq>tol)
            E(ngen_buses(jj))=sqrt(Vmsq);
            F(ngen_buses(jj))=0;
        end

        if (E(ngen_buses(jj))^2+F(ngen_buses(jj))^2-VMsq>tol)
            E(ngen_buses(jj))=sqrt(VMsq);
            F(ngen_buses(jj))=0;
        end
    end

    f(i)=-loss_func2();

end

% update pbest:
for k=1:p
     if(f(k)<fprev(k))
        pbest(k,:)=X(k,:);
     end
end

    % save current objective function values for comparison later:
    fprev=f;

    % determine gbest:
    best_idx=find(f==min(f));
    best_idx=best_idx(1);
    gbest=X(best_idx,:);
```

C.7 Function that computes the fitness value of an individual particle for the PSO algorithm; applies to all case studies

```
function f_best=PSO_objective_evaluation(gbest)
global E F bus_type nbus

Vmsq=.95^2;
VMsq=1.1^2;
tol=1e-4;

% evaluate objective and constraint functions;
% for any bound constraint exceeding the limit,
```

```
% set it to the limit value
gen_buses=[find(bus_type==1); find(bus_type==2)];

if (-(gbest(1)^2+F(1)^2)+Vmsq>tol)
    E(1)=sqrt(Vmsq);
    F(1)=0;
end

if (gbest(1)^2+F(1)^2-VMsq>tol)
    E(1)=sqrt(VMsq);
    F(1)=0;
end

k=2;
for j=2:length(gen_buses)
    if (-(gbest(k)^2+gbest(k+1)^2)+Vmsq>tol)
        E(gen_buses(j))=sqrt(Vmsq);
        F(gen_buses(j))=0;
        k=k+2;
    end

    if (gbest(k)^2+gbest(k+1)^2-VMsq>tol)
        E(gen_buses(j))=sqrt(VMsq);
        F(gen_buses(j))=0;
        k=k+2;
    end
end

ngen_buses=1:nbus;
ngen_buses(gen_buses)=[];

for jj=1:length(ngen_buses)
    if (-(E(ngen_buses(jj))^2+F(ngen_buses(jj))^2)+Vmsq>tol)
        E(ngen_buses(jj))=sqrt(Vmsq);
        F(ngen_buses(jj))=0;
    end

    if (E(ngen_buses(jj))^2+F(ngen_buses(jj))^2-VMsq>tol)
        E(ngen_buses(jj))=sqrt(VMsq);
        F(ngen_buses(jj))=0;
    end
end
f_best=-loss_func();
```

C.8 Function that computes the velocity update and adjusts the particle position for the PSO algorithm; applies to all case studies

```
function [V,X]=PSO_X_update_cc1(V,X,pbest,gbest,c1,c2)
    [~,n]=size(X);

    % define constriction coefficient:
    phi=c1+c2;
```

```
    xi=2/(phi-2+sqrt(phi^2-4*phi));

    Gbest=ones(size(X));
    for i=1:n
        Gbest(:,i)=gbest(i)*Gbest(:,i);
    end

    % compute particle velocity:
    V=xi*(V+c1*rand(size(X)).*(pbest-X)+c2*rand(size(X)).*(Gbest-X));
    %V=xi*(V+c1*rand*(pbest-X)+c2*rand*(Gbest-X));
    X=X+V;
```

APPENDIX D: UTILITY FUNCTIONS USED IN CHAPTER 5

D.1 Function to compute the impedance (Y) matrix for an arbitrary power system

```
function [G, B, Cf]=compute_Ybus(from, to, r, x)
[R, X, Cf]=computeRX(from, to, r, x);
[~,n]=size(R);
Y=zeros(n);
for i=1:n
    for j=1:n
        if (i==j)
            for k=1:n
                if(Cf(i,k)==1 && i~=k)
                    Y(i,j)=Y(i,j)+1/(R(i,k)+1i*X(i,k));
                end
            end
        else
            if (Cf(i,j)==1 && i~=j)
                Y(i,j)=-1/(R(i,j)+1i*X(i,j));
                Y(j,i)=Y(i,j);
            end
        end
    end
end
G=real(Y);
B=imag(Y);
```

D.2 Function to compute the generator active and reactive power outputs once the load flow computation has converged

```
function compute_PQ()
global E F G B bus_type nbus Cf PQs Qsum

% Define some parameters:
Vsq=E.^2+F.^2;

gen_buses=find(bus_type==2);
```

```
ng=length(gen_buses)+1;
PQ=zeros(ng+1,1);
PQ(1)=G(1,1)*Vsq(1);
PQ(2)=-B(1,1)*Vsq(1);
for j=1:nbus
        if (Cf(1,j)==1 && j~=1)
            PQ(1)=PQ(1)+E(1)*(G(1,j)*E(j)-B(1,j)*F(j))+...
                F(1)*(G(1,j)*F(j)+B(1,j)*E(j));
            PQ(2)=PQ(2)+F(1)*(G(1,j)*E(j)-B(1,j)*F(j))-...
                E(1)*(G(1,j)*F(j)+B(1,j)*E(j));
        end
end

k=1;

for i=3:ng+1
    l=gen_buses(k);
    PQ(i)=-B(l,l)*Vsq(l);
    for j=1:nbus
            if (Cf(l,j)==1 && l~=j)
                PQ(i)=PQ(i)+F(l)*(G(l,j)*E(j)-B(l,j)*F(j))-...
                    E(l)*(G(l,j)*F(j)+B(l,j)*E(j));
            end
    end
    k=k+1;
end

PQs=[PQs PQ];
Qsum=sum(PQs(2:end,:));
```

D.3 Function to define the initial guess for the Newton-Raphson load flow computation

```
function x0=define_x0()
global nbus E F
x0=zeros(2*(nbus-1),1);
k=1;
for i=2:nbus
    x0(k)=E(i);
    x0(k+1)=F(i);
    k=k+2;
end
```

D.4 Function to define the generator voltage magnitude set-points for the Newton-Raphson load flow computation

```
function Vgref=Vgref_0()
global E F nbus bus_type

Vgref=zeros(nbus,1);
```

```
Vgref(1)=sqrt(E(1)^2+F(1)^2);

for i=2:nbus
    if (bus_type(i)==2)
        Vgref(i)=sqrt(E(i)^2+F(i)^2);
    end
end
```

D.5 Function to compute the real power loss magnitude for an arbitrary power system

```
function ploss=loss_func()
global E F G Cf
[n, ~]=size(G);
ploss=0;
for i=1:n-1
    for j=i+1:n
        if (Cf(i,j)==1)
            ploss=ploss+G(i,j)*((E(i)-E(j))^2+(F(i)-F(j))^2);
        end
    end
end
```